Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Marcel Kołodziejczyk
Nr albumu: 219533

Luki w bezpieczeństwie systemu operacyjnego Android

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dra Marcina Peczarskiego
Instytut Informatyki

Czerwiec 2013

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora (autorów) pracy

Streszczenie

krótkie streszczenie pracy

Słowa kluczowe

android, arm, atak, bezpieczeństwo, przepełnienie bufora, metasploit, exploit, shellcode

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna

D. Software
D.4. Operating Systems
D.4.6. Security and Privacy Protection

Tytuł pracy w języku angielskim

Vulnerabilities in Android operating system

Spis treści

Todo list

Wprowadzenie

Rozdział 1

Platforma sprzętowa i programowa

Android jest systemem operacyjnym i zestawem aplikacji dedykowanym przede wszystkim dla urządzeń przenośnych z ekranami dotykowymi, takimi jak np. smartphone, tablet. Jądro systemu, zostało oparte na jądrze Linuksa. System ten został zaprojektowany i stworzony głównie z myślą o urządzeniach wyposażonych w procesor w architekturze ARM, aczkolwiek podejmowane są prace nad dostosowaniem Androida do innych architektur, np. x86.

W rozdziale tym zostaną opisane podstawy architektury procesorów ARM. Następnie zostanie omówiona architektura oraz model bezpieczeństwa systemu Android.

1.1. Architektura procesorów ARM

ARM jest 32-bitową architekturą procesorów typu RISC. Główne jej cechy to:

- proste tryby adresowania
- instrukcje stałej długości co ułatwia adresowanie
- architektura typu *load/store* - operacje wykonywane są na rejestrach a nie bezpośrednio na pamięci
- duża liczba 32-bitowych rejestrów
- zredukowana liczba instrukcji

Z biegiem czasu ukazywały się kolejne wersje architektury ARM. Niniejsza praca bazuje na wersji 7 (ARMv7), które jest obecnie najbardziej powszechnie wykorzystywana w urządzeniach przenośnych.

1.1.1. Thumb-2

Instrukcje ARM są stałej, 32-bitowej długości. W celu zwiększenia gęstości kodu został wprowadzony drugi, uproszczony zestaw 16-bitowych instrukcji Thumb-2. Ponieważ w obydwu trybach adresy instrukcji muszą być odpowiednio wyrównane, ostatni bit adresu może być wykorzystany w celu zmiany trybu pracy procesora. Instrukcja skoku do adresu, którego ostatni bit jest zapalony, wymusza zmianę trybu na Thumb-2 i dalsze wykonywanie instrukcji spod adresu odpowiednio wyrównanego. Analogicznie instrukcja skoku do parzystego adresu powoduje przejście w 32-bitowy zestaw instrukcji ARM.

1.1.2. Rejestry

Z punktu widzenia programisty dostępnych jest szesnaście 32-bitowych rejestrów R0-R15. Trzy z nich mają dedykowane przeznaczenie:

- SP (Stack Pointer) - R13 - wskaźnik stosu
- LR (Link Register) - R14 - zawiera adres następnej instrukcji przy instrukcjach skoku wywołania podprogramu
- PC (Program Counter) - R15 - przechowuje adres następnej instrukcji

Dodatkowo występuje rejestr statusowy procesora CPSR (ang. Current Porcessor Status Register). Przechowuje on m. in. flagi Negative, Zero, Carry, oVerflow. Większość instrukcji może być wykonywanych warunkowo, w zależności od stanu tych flag rejestru statusu (APSR).

Pełna dokumentacja znajduje się w [12].

1.1.3. Standard wywołania procedur

Do wykonywania procedur służą następujące instrukcje:

- **B** - Branch
- **BL** - Branch with Link
- **BX** - Branch and Exchange
- **BLX** - Branch with Link and Exchange

Instrukcja *Branch* umożliwia wykonanie skoku o maksymalnie 32 MB w przód lub w tył od bieżącej instrukcji. *Branch with link* dodatkowo zachowuje adres powrotu w rejestrze LR (R14). Pozostałe dwie instrukcje jako argument przyjmują rejestr - skok jest wykonywany do adresu, jaki znajduje się w przekazanym rejestrze.

Architektura ARM określa następujące zasady wywoływania procedur:

- Do przekazywania argumentów i zwracania wyniku procedury używane są rejestry R0-R3. Kolejne parametry mogą być przekazywane na stosie.
- rejestry R4-R11 mogą być wykorzystywana do przechowywania zmiennych lokalnych
- Zawartość rejestrów R4-R12 powinna być zachowana w trakcie wykonania procedury. Zazwyczaj w prologu procedury rejestry te są odkładana na stos, aby przywrócić ich wartości w epilogu.
- Stos rośnie w kierunku mniejszych adresów pamięci.

Bardziej szczegółowe informacje można znaleźć w [13].

1.2. Architektura systemu Android

1.3. Model bezpieczeństwa Androida

Rozdział 2

Przykłady ataków

2.1. Klasyczny błąd przepełnienie bufora

2.2. Technika „heap spray”

2.2.1. NX bit

W celu ochrony przed tego typu takimi w wersji 6 architektury ARM wprowadzona została technologia NX bit (No Execute). Umożliwia ona systemowi operacyjnemu oznaczyć wybrane strony pamięci jako niewykonywalne. Gdy bit NX dla danej strony jest ustawiony, próba wykonania zawartości tej strony jako kodu kończy się wygenerowaniem wyjątku, zgłaszanego systemowi operacyjnemu, co powoduje przerwanie wykonywania programu. Bit NX powinien być ustawiony dla wszystkich stron procesu, z wyjątkiem programu i bibliotek oraz świadomie dozwolonych przez program wyjątków.

Technologia „NX bit” jest wspierana przez Androida od wersji 2.3.

wyko dzięki któremu system operacyjny

2.3. Technika „return to library” (Ret2Libc)

Rozdział 3

Tworzenie payloadów

Rozdział 4

Rozszerzenie Matesploita

4.1. CVE-2010-1119

4.2. CVE-2010-1807

Rozdział 5

Podsumowanie

Bibliografia

[1] Anthony Desnos, Geoffroy Gueguen *Android: From Reversing to Decompilation*, Black Hat, Abu Dhabi, 2011

[2] S. Höbarth, R. Mayrhofer, *A framework for on-device privilege escalation exploit execution on android*, IWSSI/SPMU 2011: 3rd International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use, colocated with Pervasive 2011, czerwiec 2011. dostępne na `http://www.medien.ifi.lmu.de/iwssi2011/`

[3] Gaurav Kumar, Aditya Gupta, *A Short Guide on ARM Exploitation*, `http://www.exploit-db.com/wp-content/themes/exploit/docs/24493.pdf`

[4] Yves Younan, Pieter Philippaerts, *Alphanumeric RISC ARM shellcode*, Phrack, 66, czerwiec 2009

[5] Joshua Hulse, *Buffer Overflows: Anatomy of an Exploit*, `http://packetstormsecurity.com/files/108549/Buffer-Overflows-Anatomy-Of-An-Exploit.html`

[6] Emanuele Acri, *Exploiting Arm Linux Systems*, `http://packetstormsecurity.com/files/98376/Exploiting-ARM-Linux-Systems.html`

[7] Collin Mulliner, Charlie Miller, *Fuzzing the Phone in your Phone*, Black Hat USA, 2009

[8] Jonathan Salwan, *How to Create a Shellcode on ARM Architecture*, `http://www.exploit-db.com/papers/15652/`

[9] Dustin „I)ruid" Trammel, *Metasploit Framework Telephony*, Black Hat USA, 2009

[10] Itzhak Avraham, *Non-Executable Stack ARM Exploitation*, Black Hat DC, 2011

[11] jip@soldierx.com, *Stack Smashing On A Modern Linux System*, `http://www.soldierx.com/tutorials/Stack-Smashing-Modern-Linux-System`

[12] ARM Ltd. *Arm architecture reference manual.*

[13] ARM Ltd. *Procedure call standard for the arm architecture.*

[14] Metasploit framework, `http://www.metasploit.com`

[15] Android project, `http://developer.android.com`

[16] The WebKit Open Source Project, `http://www.webkit.org`