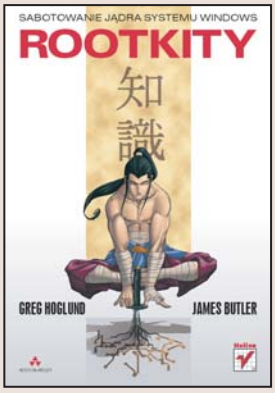
Rootkity. Sabotowanie jądra systemu Windows



Autorzy: Greg Hoglund, Jamie Butler
Tłumaczenie: Wojciech Moch
ISBN: 83-246-0257-7
Tytuł oryginału: Rootkits: Subverting the Windows Kernel
Format: B5, stron: 312

Chcesz ochronić swój system? Poznaj jedno z najpoważniejszych zagrożeń

- Sposób działania rootkitów
- Pisanie rootkitów i narzędzi chroniących przed nimi
- Wykrywanie rootkitów

Rootkit to zestaw programów i kodów pozwalający hakerowi na niewykrywalny dostęp do komputera, a tym samym na korzystanie z cudzego systemu operacyjnego. Narzędzie takie można stworzyć, znając luki w zabezpieczeniach jądra systemu operacyjnego i dysponując odpowiednimi umiejętnościami. Ale można również uchronić się przed jego działaniem, co dla osoby odpowiedzialnej za bezpieczeństwo komputera jest zdecydowanie ważniejsze.

Dzięki książce „Rootkity. Sabotowanie jądra systemu Windows" poznasz swojego przeciwnika i nauczysz się z nim walczyć, tworząc własny arsenał. Greg Hoglund i James Butler przedstawiają sposoby, z jakich korzystają hakerzy, aby włamywać się do komputerów i używać ich bez wiedzy właścicieli. Książka opisuje szczegóły sabotowania jądra systemów Windows XP i Windows 2000 oraz koncepcje, które można zastosować w praktycznie każdym systemie operacyjnym — od Windows Server 2003, poprzez Linuksa, aż po inne systemy uniksowe. Czytając tę książkę, poznasz techniki programowania rootkitów oraz tworzenia systemów obronnych.

- Zasada działania rootkitów
- Sposoby wprowadzania kodu do jądra systemu
- Tworzenie rootkitów
- Manipulowanie obiektami jądra systemu
- Uzyskiwanie bezpośredniego dostępu do sterowników sprzętu
- Wykorzystywanie w rootkitach połączeń sieciowych
- Wykrywanie rootkitów w systemie

Jeśli zajmujesz się bezpieczeństwem komputerów i danych, koniecznie przeczytaj tę książkę

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 1.
Nie zostawić śladu

Subtelny i dyskretny, ekspert nie zostawia śladów;
bosko tajemniczy, jest niesłyszalny.
Jest zatem władcą losu swego przeciwnika.

— Sun Tzu

Wiele książek omawia sposoby penetracji systemów komputerowych i oprogramowania. Liczni autorzy zajmowali się się już kwestiami uruchamiania złośliwych skryptów, tworzenia przepełnień bufora i ciekawych skryptów powłoki. Do najważniejszych przykładów takich książek zaliczyć można: *Exploiting Software*[1], *The Shellcoder's Handbook*[2] i *Hacking Exposed*[3].

Ta książka jest inna. Zamiast zajmować się w niej technikami ataków, opisywać będziemy sposoby atakujących na pozostanie w naszych systemach **po** udanym włamaniu. Ten temat omawiany jest w bardzo niewielu publikacjach, z wyjątkiem tych zajmujących się analizą przejętych systemów. W przypadku tych ostatnich najczęściej opisywane są metody defensywne, czyli wykrywanie obecności włamywacza i wsteczna inżynieria złośliwego kodu. W tej książce zajmiemy się znacznie bardziej agresywnymi zachowaniami. Omawiać będziemy sposoby penetrowania systemu komputerowego uniemożliwiające wykrycie tego zdarzenia. W końcu udana penetracja systemu nie może zostać wykryta.

W niniejszym rozdziale wprowadzimy Czytelnika w świat technologii rootkitów i ogólnych zasad ich działania. Rootkity stanowią tylko jeden z elementów spektrum zagrożeń dla naszego komputera, ale są one niezbędnym elementem wielu typów ataków.

Rootkity nie są złośliwe same z siebie, ale mogą być używane w ramach złośliwych programów. Poznanie technologii rootkitów jest niezbędne każdemu, kto chce ochronić swój system przed nowoczesnymi technikami ataków.

---

[1] G. Hoglund i G. McGraw, *Exploiting Software: How to Break Code*, Boston, Addison-Wesley, 2004. Proszę zajrzeć również na stronę *www.exploitingsoftware.com*.

[2] J. Koziol, D. Lichtfield, D. Aitel, C. Anley, S. Eren, N. Mehta i R. Hassell, *The Shellcoder's Handbook*, Nowy Jork, John Wiley & Sons, 2004.

[3] S. McClure, J. Scambray i G. Kurtz, *Hacking Exposed*, Nowy Jork, McGraw-Hill, 2003.

Zrozumienie motywów atakującego

Tylne drzwi do komputera stanową tajny sposób na dostęp do tej maszyny. W wielu hollywoodzkich filmach tylne drzwi przedstawiane są jako tajne hasło lub sposób na uzyskanie dostępu do bardzo ściśle chronionego systemu komputerowego. Nie są to jednak tylko wymysły filmowców. Tylne drzwi są niezwykle poważnym zagrożeniem dla naszych komputerów i mogą być wykorzystywane do kradzieży danych, monitorowania użytkowników i przeprowadzania ataków w ramach sieci komputerowych.

Atakujący może mieć wiele powodów do pozostawienia w naszym komputerze tylnego wejścia. Włamanie do systemu to naprawdę ciężka praca, dlatego po przeprowadzeniu udanego włamania każdy atakujący będzie chciał utrzymać zajęte pozycje. Może też chcieć wykorzystać przechwycony komputer do przeprowadzenia ataków na kolejne komputery w danej sieci.

Głównym powodem, dla którego atakujący starają się włamać do naszych komputerów, jest chęć zbierania informacji. W tym celu atakujący będzie monitorował naciśnięcia klawiszy, obserwował nasze zachowania, przeglądał pakiety przesyłane przez sieć i w ten sposób potajemnie **wydobywał dane** z naszego systemu. Te wszystkie operacje wymagają jednak wcześniejszego przygotowania tylnego wejścia do tego systemu. Dlatego właśnie atakujący będzie chciał pozostawić w przechwyconym systemie oprogramowanie pozwalające mu na zbieranie informacji.

Atakujący może też włamać się do naszego komputera w celu jego zniszczenia i w związku z tym pozostawić w komputerze tak zwaną **bombę logiczną**, która ma za zadanie zniszczyć komputer w określonym czasie. Taka bomba w czasie oczekiwania musi jak najlepiej ukrywać się w systemie. Jest to zatem przypadek, w którym atakujący nie musi tworzyć w systemie tylnego wejścia, ale i tak pozostawia ukrywające się w nim oprogramowanie.

Znaczenie niewidzialności

Jeżeli pozostawiony program tworzący tylne wejścia ma uniknąć wykrycia, to musi korzystać z technik niewidzialności. Niestety, wiele z dostępnych publicznie „hakerskich" programów tego typu nie ukrywa się zbyt dobrze. W takich programach źle działać może wiele elementów, co najczęściej wynika z tego, że ich twórcy starają się w nie wbudować wszystko, włącznie z kuchennym zlewem. Przyjrzyjmy się na przykład programom Back Orfice lub NetBus. Każdy z nich oferuje imponującą listę funkcji, wśród których znajdziemy również tak niedorzeczne jak wysuwanie tacki napędu CD-ROM. Taka funkcja może się przydać przy robieniu komuś złośliwości w biurze, ale z całą pewnością nie przyda się w czasie planowania profesjonalnego ataku[4].

Jeżeli atakujący nie będzie dostatecznie ostrożny, to może niechcący ujawnić swoją obecność w sieci i w ten sposób narazić na szwank całość operacji. Z tego powodu

[4] W tym przypadku **profesjonalny** oznacza pewnego rodzaju usankcjonowane działanie wykonywane na przykład przez służby bezpieczeństwa.

profesjonalnie przeprowadzony atak wymaga bardzo specjalizowanego i automatycznie działających programów typu „tylne wejście", czyli wykonujących wyłącznie jedną rzecz. Dzięki temu można mieć pewność, że uzyskamy właściwe wyniki.

Jeżeli osoby obsługujące komputery zaczęłyby podejrzewać, że ktoś włamał się do sieci ich komputerów, to mogłyby rozpocząć intensywne przeszukiwanie sieci, szukając w niej nietypowych działań lub programów typu „tylne wejście"[5]. Najlepszym sposobem przeciwdziałania takim śledztwom jest skuteczne ukrywanie się. Jeżeli nikt nie będzie podejrzewał ataku, to najprawdopodobniej nie zacznie też przeszukiwać systemu. Atakujący mają do dyspozycji wiele metod ukrywania się. Niektórzy mogą ograniczać się do absolutnego minimum i po prostu jak najbardziej ograniczać generowany ruch w sieci oraz rezygnować z zapisywania plików na dysku. Inni mogą natomiast używać w swoich działaniach plików, ale jednocześnie mogą stosować najróżniejsze techniki utrudniające przeprowadzenie śledztwa w komputerze. Jeżeli techniki ukrywania się w systemie stosowane będą prawidłowo, to na zainfekowanym komputerze praktycznie nigdy nie zostanie przeprowadzone żadne śledztwo, ponieważ takie włamanie nie zostanie nigdy zauważone. Nawet jeżeli takie śledztwo zostanie przeprowadzone z powodu różnorakich podejrzeń, to dobre techniki ukrywania rootkitu mogą spowodować, że nawet dokładne śledztwo nie wykryje włamania.

Kiedy nie trzeba się ukrywać?

Czasami włamywacz wcale nie musi się ukrywać w systemie. Na przykład w sytuacji, gdy chce dostać się do danego komputera i korzystać z niego tylko tak długo, aż uda się mu wykraść pewne dane, takie jak zbiór adresów e-mail, to najprawdopodobniej nie będzie się przejmować tym, czy takie włamanie zostanie kiedykolwiek zauważone.

Nie ma potrzeby ukrywania się również w sytuacji, gdy chcemy po prostu zakłócić działanie zdalnego komputera. Taki komputer może na przykład sterować wrogim systemem przeciwlotniczym. Nie trzeba się w tej sytuacji ukrywać, ponieważ zniszczenie systemu jest już całkowicie wystarczające do osiągnięcia naszych celów. W większości przypadków zniszczenie systemu będzie i tak bardzo oczywistą (i niezwykle kłopotliwą) informacją dla naszej ofiary. Jeżeli to takie ataki chciałbyś przeprowadzać, drogi Czytelniku, to niestety, nie jest to książka dla Ciebie.

Skoro znamy już choćby najważniejsze motywy atakujących, możemy w pozostałej części rozdziału bardzo ogólnie pomówić o samych rootkitach, w tym również o podstawach zasad ich działania.

Czym jest rootkit?

Termin **rootkit** istnieje już od mniej więcej dziesięciu lat. Rootkit jest „zestawem" (ang. *kit*) składającym się z niewielkich, ale bardzo użytecznych programów pozwalających

[5] Jako dobrą książkę omawiającą podobne śledztwa można polecić pozycję D. Farmera i W. Venema, *Fornesic Discovery* (Boston: Addison-Wesley, 2004).

atakującemu na uzyskanie dostępu administratora (ang. *root*), czyli najbardziej uprzywilejowanego użytkownika w systemie. Innymi słowy, **rootkit jest zestawem programów i kodu pozwalającym na trwałą i niewykrywalną obecność w obcym komputerze**.

W podanej definicji rootkitu najważniejsze jest słowo „niewykrywalna". Większość technologii i rozwiązań stosowanych w rootkitach przeznaczonych jest do ukrywania jego kodu i danych w systemie. Na przykład wiele rootkitów jest w stanie ukryć określone pliki i katalogi. Inne funkcje rootkitów obejmują najczęściej zdalny dostęp do komputera oraz podsłuchiwanie jego użytkowników, czyli na przykład przeglądanie pakietów przesyłanych w sieci. Połączenie tych wszystkich funkcji może stanowić nokautujący cios dla wszelkich systemów zabezpieczających.

Rootkity same z siebie nie są „złe" i nie zawsze wykorzystywane są przez „przestępców". Bardzo ważne jest, żeby przyswoić sobie fakt, że rootkit jest jedynie technologią. Dobre lub złe intencje jej użycia mogą mieć tylko wykorzystujący ją ludzie. Istnieje wiele pożytecznych programów komercyjnych pozwalających na zdalne administrowanie systemem, a nawet funkcje podsłuchiwania. W niektórych z tych programów stosowane są nawet funkcje ukrywające, a zatem pod wieloma względami są one podobne do rootkitów. W ramach dyskusji prawniczych termin „rootkit" może być stosowany do opisywania usankcjonowanego programu typu „tylne wejście", który legalnie instalowany jest na danym komputerze na wniosek organów państwowych sądu (takie sprawy rozpatrywać będziemy w punkcie „Uprawnione użycie rootkitów"). Wielkie korporacje również mogą stosować rootkity do monitorowania i wymuszania prawidłowego wykorzystania komputerów pracowników.

Postaramy się przedstawić umiejętności i techniki naszego przeciwnika, przedstawiając rootkity z perspektywy atakującego. W ten sposób znacznie podniesiemy nasze umiejętności w zabezpieczaniu się przed takimi zagrożeniami. Niniejsza książka pomoże również wszystkim prawym twórcom rootkitów, przedstawiając techniki, z których będą mogli oni skorzystać.

Dlaczego istnieją rootkity?

Rootkity to względnie nowy wynalazek, choć szpiegostwo jest równie stare, jak i sama wojna. Rootkity powstały z tych samych powodów co pluskwy do podsłuchów. Jedni ludzie chcą zobaczyć lub kontrolować to, co robią inni ludzie. Ze względu na ciągle rosnącą ilość przetwarzanych danych, komputery stały się naturalnym celem takich ataków.

Rootkity przydają się tylko wtedy, gdy chcemy utrzymać nasz ciężko uzyskany dostęp do systemu. Jeżeli chodzi nam tylko o wykradzenie pewnych danych i porzucenie komputera, to naprawdę instalowanie rootkitu nie ma sensu. Co więcej, zainstalowanie rootkitu sprawia, że zwiększamy prawdopodobieństwo wykrycia naszego włamania. Jeżeli uda się nam wykraść pewne dane i dokładnie wyczyścić system, to najprawdopodobniej nie zostawimy żadnych śladów naszej aktywności.

Rootkity spełniają zatem dwie podstawowe funkcje: umożliwiają zdalną kontrolę nad komputerem i podsłuchiwanie innych programów.

Zdalna kontrola

Zdalna kontrola nad komputerem może obejmować kontrolę nad plikami, wymuszanie ponownego uruchomienia komputera lub „niebieskich ekranów śmierci", a nawet umożliwienie dostępu do wiesza poleceń (czyli *cmd.exe* lub */bin/sh*). Na rysunku 1.1 przedstawiony został przykład menu poleceń rootkitu. Takie menu daje nam przegląd funkcji oferowanych przez rootkit.

Rysunek 1.1. *Menu rootkitu*

```
Win2K Rootkit by the team rootkit.com
Version 0.4 alpha
----------------------------------------
command       description
ps            show process list
help          this data
buffertest    debug output
hidedir       hide prefixed file or directory
hideproc      hide prefixed processes
debugint      (BSOD)fire int3
sniffkeys     toggle keyboard sniffer
echo <string> echo the given string

*"(BSOD)" means Blue Screen of Death
  if a kernel debugger is not present!
*"prefixed" means the process or filename
  starts with the letters '_root_'.
*"sniffer" means listening or monitoring software.
```

Podsłuchiwanie oprogramowania

Podsłuchiwanie oprogramowania pozwala na obserwowanie operacji wykonywanych przez użytkowników komputera. Oznacza to przeglądanie pakietów sieciowych, przechwytywanie naciśnięć klawiszy i czytanie wysyłanych e-maili. Atakujący może wykorzystywać te techniki do przechwytywania haseł i zaszyfrowanych plików, a nawet kluczy szyfrujących.

Uprawnione użycie rootkitów

Jak już wspominaliśmy wcześniej, rootkity mogą być też używane jak najbardziej legalnie. Na przykład mogą być stosowane przez agencje chroniące prawo w celu zbierania informacji na podobnej zasadzie, na jakiej stosuje się podsłuchy. Takiego rozwiązania można użyć w związku z każdym przestępstwem, w którym wykorzystywany jest komputer. Chodzi tu o włamania komputerowe, tworzenie i dystrybucja dziecięcej pornografii, piractwo oprogramowania lub muzyki.

Broń elektroniczna

Co prawda rootkity są często stosowane jako broń elektroniczna, jednak nie jest to ich podstawowe zastosowanie.

Wojny toczą się na wielu frontach, przy czym front ekonomiczny z całą pewnością jest jednym z ważniejszych. Od zakończenia drugiej wojny światowej, przez cały czas „zimnej wojny", Związek Radziecki tworzył wiele grup zajmujących się zbieraniem informacji na temat technologii rozwijanych w Stanach Zjednoczonych[6].

Po wykryciu kilku takich grup w USA podsunięto im fałszywe plany, oprogramowanie i materiały. W jednym z opublikowanych wypadków to właśnie „złośliwe modyfikacje oprogramowania" zostały uznane za przyczynę wybuchu gazociągu na Syberii[7]. Eksplozja ta została sfotografowana przez satelity, a eksperci określali ją jako „najbardziej przerażający wybuch nienuklearny, jaki kiedykolwiek został zaobserwowany z przestrzeni kosmicznej"[8].

Rootkity wykorzystywane są również w działaniach wojennych. Potencjały militarne różnych państw w znaczącym stopniu bazują na sprzęcie komputerowym. Jeżeli te komputery przestaną działać, to z pewnością wpłynie to w obozie przeciwnika na cykl podejmowania decyzji i skuteczność działania. Zaletą wykorzystywania ataków komputerowych (w stosunku do ataków konwencjonalnych) jest ich mniejszy koszt, pełne bezpieczeństwo żołnierzy i to, że nie powodują one prawie żadnych dodatkowych zniszczeń, a w większości przypadków nie są przyczyną absolutnie żadnych trwałych uszkodzeń. Na przykład, jeżeli zbombardowalibyśmy wszystkie elektrownie w pewnym kraju, to ich odbudowa pochłonęłaby ogromną ilość środków. Jeżeli jednak udałoby się nam programowo unieszkodliwić sieć dystrybucji energii, to w tym kraju nadal brakowałoby energii, ale powstałe szkody nie byłyby trwałe, a ich usunięcie nie byłoby kosztowne.

Jak długo istnieją rootkity?

Jak już mówiliśmy wcześniej, rootkity to względnie nowe rozwiązanie. Większość metod stosowanych w nowoczesnych rootkitach nie zmieniła się od lat 80. Na przykład modyfikowanie tablic systemowych, pamięci i logiki programów. Pod koniec lat 80. techniki te wykorzystywane były przez wirusy, które ukrywały się przed skanerami antywirusowymi. W tamtych czasach wirusy rozprzestrzeniały się za pośrednictwem dyskietek i BBS-ów.

Później Microsoft wprowadził system Windows NT, w którym model pamięci został zmodyfikowany tak, że normalne programy użytkownika nie mogły już modyfikować tablic systemowych. Powstał wtedy pewien zastój w rozwoju technologii wirusów, ponieważ żaden z ich twórców nie korzystał z jądra nowego systemu.

---

[6] G. Weiss „The Farewell Dossier" w *Studies in Intelligence* (Waszyngton: CIA, Center for the Study of Intelligence, 1996), dostępny na stronie *www.cia.gov/csi/studies/96unclass/farewell.thml*.

[7] Co sugeruje, że eksplozja została spowodowana przez sabotaż oprogramowania.

[8] D. Hoffman, „Zimna wojna zaogniła się jeszcze bardziej w momencie, gdy w wyniku sabotażu w powietrze wyleciał sowiecki rurociąg.", *Sydney Morning Herald*, 28 lutego 2004.

Później coraz większą popularność zaczął zdobywać internet, w którym dominowały systemy uniksowe. Większość komputerów podłączonych do internetu działała pod kontrolą tych systemów, w związku z czym wirusy były prawdziwą rzadkością. Mniej więcej w tym samym czasie zaczęły powstawać robaki sieciowe. Wraz z powstaniem pierwszego robaka o nazwie Morris Worm świat komputerów stanął przed zagrożeniem wykorzystywania błędów w oprogramowaniu (ang. *software exploits*)[9]. Na początku lat 90. wielu hakerów poznało metody wyszukiwania i wykorzystywania błędów przepełnienia bufora, które są „bombą nuklearną" wszystkich programów. Mimo to środowisko twórców wirusów przez prawie dekadę nie chciało zastosować tej techniki.

Na początku lat 90. haker mógł się włamać do systemu, ustanowić w nim przyczółek, a następnie wykorzystać zdobyty właśnie komputer do zaatakowania kolejnego komputera. Po udanym włamaniu do komputera musiał jednak w jakiś sposób zapewnić sobie stały dostęp do niego. Tak właśnie narodziły się rootkity. Początkowo rootkity były prostymi programami tworzącymi w komputerze „tylne wejście" dla swoich twórców, ale prawie wcale nieukrywającymi się w systemie. W niektórych przypadkach zastępowały one najważniejsze pliki systemowe ich zmodyfikowanymi wersjami, które ukrywały pewne pliku lub procesy. Proszę sobie wyobrazić program `ls`, który przeznaczony jest do wypisywania plików i katalogów. Rootkit pierwszej generacji najprawdopodobniej podmieniłby ten program wersją, która ukrywałaby plik o nazwie *dane_hakera*. Dzięki temu haker mógłby spokojnie zapisywać na dysku swojej ofiary wszystkie zgromadzone dane, zapisując je do tego pliku. Zmodyfikowana wersja programu `ls` dość dobrze ukrywałaby obecność tego pliku na dysku.

Administratorzy systemów zareagowali w tamtych czasach programami takimi jak *Tripwire*[10], które wykrywały ewentualne podmiany określonych plików. Kontynuując nasz poprzedni przykład, programy zabezpieczające, takie jak *Tripwire*, mogłyby sprawdzić zawartość pliku `ls` i stwierdzić, że został on zmodyfikowany, co równałoby się z wykryciem trojana.

Naturalną odpowiedzią atakujących było przeniesienie się do jądra systemu. Pierwsze rootkity atakujące jądro zostały napisane dla systemów uniksowych. Po zainfekowaniu jądra systemu takie rootkity mogły sabotować działanie dowolnego programu zabezpieczającego działającego na danym komputerze. Innymi słowy, pliki trojanów przestały być już potrzebne. Bezpieczeństwo rootkitu mogło być zapewnione przez odpowiednie zmodyfikowanie jądra systemu. Technika ta nie różniła się właściwie niczym od technik stosowanych w wirusach pod koniec lat 80. w celu ukrywania się przez oprogramowaniem antywirusowym.

---

[9] Robert Morris przygotował pierwszego udokumentowanego robaka internetowego. Więcej informacji na temat tego robaka znaleźć można w książce K. Hafnera i J. Markoffa, *Cyberpunkt: Outlaws and hackers on the Computer Frontier*, (Nowy Jork, Simon & Schuster, 1991).

[10] *www.tripwire.org*.

Jak działają rootkity?

Rootkity w swoim działaniu wykorzystują prostą koncepcję **modyfikacji**. Mówiąc ogólnie, oprogramowanie przygotowane jest do tego, żeby podejmować określone decyzje na podstawie określonych danych. Rootkit jest w stanie zlokalizować i zmodyfikować to oprogramowanie tak, żeby podejmowało niewłaściwe decyzje.

Tego rodzaju modyfikacje można wprowadzać w wielu miejscach w programach. Niektóre z nich omówimy w kolejnych punktach.

Poprawianie

Kod wykonywalny (czasami nazywany jest **binarką**) składa się z serii instrukcji zapisanych w postaci ciągu bajtów. Takie bajty muszą być ułożone w ściśle określonym porządku, który dla komputera ma jakiekolwiek znaczenie. Jeżeli takie bajty zostaną zmodyfikowane, to jednocześnie zmodyfikowana zostanie logika oprogramowania. Taka technika czasami nazywana jest **poprawianiem** (ang. *patching*), a podobna jest do nakładania kolorowych łatek na starą kołdrę. Oprogramowania nie można nazwać sprytnym. Wykonuje ono tylko kolejne rozkazy i nie zajmie się niczym innym. To właśnie dlatego tak doskonale sprawdzają się tego rodzaju modyfikacje. Co więcej, ich wprowadzanie wcale nie jest tak bardzo skomplikowane. Poprawianie bajtów kodu jest jedną z najważniejszych technik stosowanych w tak zwanych „krakach", które usuwają zabezpieczenia z oprogramowania. Innym zastosowaniem takich poprawek jest wprowadzanie możliwości oszukiwania w grach komputerowych (na przykład nieograniczone złoto, zdrowie lub inne dodatki).

Jaja wielkanocne

Zmiany w logice oprogramowania mogą być również „wbudowane". Takie „tylne wejścia " w programie może umieszczać jego twórca. Nie jest to oczywiście opisywane w dokumentacji programu, a zatem takie funkcje zostają w utajone. Czasami prezentowane podejście nazywane jest **jajami wielkanocnymi** (ang. *easter eggs*) i przez twórców programów używane jest do pozostawienia swojego specjalnego podpisu. Jest to swego rodzaju specjalny znak mówiący, że to ten programista napisał dany program. Wczesne wersje bardzo szeroko wykorzystywanego programu Microsoft Excel zawierały takie jajo wielkanocne, które pozwalało zagrać w trójwymiarową „strzelankę" podobną do starej gry Doom[11] wyświetlaną w jednej z komórek arkusza kalkulacyjnego.

Spyware

Czasami jeden program specjalnie infekuje inny program, wprowadzając do niego elementy szpiegujące (tzw. spyware). Niektóre programy tego typu zapamiętują strony WWW, jakie odwiedzają użytkownicy komputera. Takie programy szpiegujące mogą

[11] *The Easter Eggs and Curios Database*, *www.eggheaven2000.com.*

być bardzo trudne do wykrycia, podobnie jak i rootkity. Niektóre z tych programów podłączają się do przeglądarek stron WWW lub powłoki systemowej, przez co ich usunięcie jest niezwykle trudne. Później mogą zamienić życie użytkownika w piekło, cały czas umieszczając na pulpicie skróty do ofert nowych kredytów lub sprzedaży Viagry. Użytkownik takiego komputera jest zatem ciągle uświadamiany, że jego przeglądarka stron WWW jest całkowicie niezabezpieczona[12].

Modyfikacje kodów źródłowych

Czasami oprogramowanie modyfikowane jest dosłownie już u samych źródeł. Programista może wstawić do kodu źródłowego programu wiersze złośliwego kodu. Ta możliwość powoduje, że niektóre organizacje wojskowe starają się unikać stosowania pakietów o otwartych źródłach, takich jak Linux. Takie projekty umożliwiają niemal każdemu (w tym sensie, że „każdy" oznacza osoby, których nie znamy) dodanie do źródeł nowego kodu. Oczywiście najważniejsze części kodu takich projektów jak BIND, Apache lub Sendmail sprawdzane są przez wszystkich uczestników takich projektów. Z drugiej strony, czy na pewno ktokolwiek przegląda kod projektów wiersz po wierszu? (Jeżeli tak jest, to raczej nie udaje się zbyt skutecznie wyszukiwać dziur w zabezpieczeniach.) Proszę sobie wyobrazić, że „tylne wejście" może zostać wprowadzone jako poprawka jakiegoś błędu w programie. Na przykład złośliwy programista może celowo narazić program na powstawanie błędów przepełnienia bufora. A ponieważ będzie to ukryte w poprawce pewnego błędu, to wykrycie takiego zagrożenia nie będzie łatwe. Co więcej, taki złośliwy programista może twierdzić, że tego „błędu" nie wprowadził celowo!

Tak, teraz zapewne usłyszymy: „Pewnie! Ja całkowicie ufam wszystkim tym, którzy tworzyli wykorzystywane przeze mnie oprogramowanie, ponieważ każdy z nich jest co najwyżej o trzy podania ręki od Linusa Torvaldsa[13], a jemu ufam całkowicie!". No dobrze, ale czy równym zaufaniem możemy obdarzyć administratorów systemów, na których działają serwery kontroli źródeł albo administratorów dystrybucji tych kodów źródłowych? A to tylko kilka przykładów miejsc, w których atakujący mogą uzyskać dostęp do kodów źródłowych. Najlepszym przykładem, jak wielkie może to być zagrożenie, jest pamiętne włamanie z 2003 roku na główne serwery FTP projektu GNU (*gnu.org*), będące źródłem wszystkich systemów opartych na Linuksie[14]. Modyfikacje kodów źródłowych mogą się przenosić do setek różnych programów i w związku z tym mogą być bardzo trudne do zlokalizowania. W ten sposób mogą zostać zaatakowane nawet kody źródłowe narzędzi stosowanych przez profesjonalistów[15].

---

[12] Wiele przeglądarek WWW staje się łupem tego rodzaju szpiegów, przy czym największym celem twórców takiego oprogramowania jest, oczywiście, Internet Explorer.

[13] Linus Torvalds jest „ojcem" Linuksa.

[14] CERT Advisory CA-2003-21, dokument dostępny jest na stronie *www.cert.org/advisories/CA-2003-21.html*.

[15] Na przykład na stronę *monkey.org* należącą do D. Songa włamano się w maju 2002 roku i wstawiono poprawki do źródeł przechowywanych na tej stronie programów *Dsniff*, *Fragroute* i *Fragrouter*. Proszę zobaczyć też dokument „Download Sites Hacked, Source Code Backdoored" dostępny na stronie *www.securityfocus.com/news/462*.

Legalność modyfikowania oprogramowania

Niektóre formy modyfikowania oprogramowania są oczywiście nielegalne. Jeżeli, na przykład, użyjemy jednego programu do zmodyfikowania innego w celu usunięcia z niego mechanizmów zapewniających prawa własności, to najprawdopodobniej naruszymy prawo (zależy do od konkretnego ustawodawstwa). Dotyczy to wszystkich „krakerów", które można swobodnie pobierać z internetu. Konkretny przykład: jeżeli pobierzemy z sieci wersję próbną programu, która przestaje działać po 15 dniach, a następnie zastosujemy wobec tego programu „krak", który umożliwi jego dalszą pracę, tak jakby został legalnie zarejestrowany, taka bezpośrednia modyfikacja kodu i logiki programu z całą pewnością będzie nielegalna.

Czym nie jest rootkit?

No dobrze, do tej pory opisaliśmy w szczegółach czym jest rootkit i omówiliśmy mniej więcej technologię umożliwiającą tworzenie rootkitów. Dowiedzieliśmy się, jak potężnym narzędziem rootkit staje się w rękach sprawnego hakera. Ale w arsenale takiego hakera znajduje się wiele narzędzi, wśród których rootkit jest tylko jednym elementem kolekcji. Należałoby zatem powiedzieć, czym rootkit **nie** jest.

Rootkit nie jest exploitem

Rootkity można stosować w połączeniu z exploitami (programami wykorzystującymi luki w innych programach), ale sam rootkit jest tylko prostym zestawem programów narzędziowych. Programy te mogą korzystać z nieudokumentowanych funkcji i metod, ale najczęściej nie korzystają one z błędów programowych (takich jak błędy przepełniania bufora).

Rootkit jest najczęściej instalowany w systemie po udanym użyciu exploitu. Wielu hakerów ma przygotowanych ogromne ilości eksploitów, ale w swoim „narzędziowniku" przechowują zwykle tylko jeden lub dwa rootkity. Niezależnie od tego, jakiego exploita użyje włamywacz, żeby dostać się do naszego systemu, później może zainstalować właściwy rootkit.

Co prawda sam rootkit nie jest eksploitem, lecz może zawierać w sobie pewne części wykorzystujące błędy w innych programach. Rootkit najczęściej wymaga dostępu do jądra systemu i dlatego zawiera jeden lub kilka programów uruchamianych razem z systemem. Istnieje tylko kilka sposobów na umieszczenie swojego kodu w jądrze systemu (na przykład jako sterownik urządzenia), a wiele z nich można wykryć za pomocą odpowiednich procedur.

Jedną z nowszych metod instalowania rootkitu w systemie jest wykorzystanie błędów w oprogramowaniu. Wiele takich błędów pozwala na wykonanie dowolnego kodu lub zainstalowanie oprogramowania firm trzecich. Proszę sobie wyobrazić, że w jądrze nastąpiło przepełnienie bufora (są udokumentowane błędy dające takie efekty), które

pozwalają na wykonanie dowolnego kodu w trybie jądra. Błędy przepełnienia bufora w jądrze mogą powstawać w niemal każdym sterowniku (na przykład w sterowniku drukarki). W czasie uruchamiania systemu program ładujący rootkit może zatem wykorzystać takie błędy w oprogramowaniu do załadowania rootkitu. Taki program ładujący nie korzysta z żadnych udokumentowanych metod ładowania i rejestrowania sterownika urządzenia, ani żadnej innej metody instalowania rootkitu. Wykorzystuje po prostu błąd przepełnienia bufora do zainstalowania części rooktitu działającej w trybie jądra.

Wykorzystanie błędu przepełnienia bufora jest mechanizmem pozwalającym na załadowanie kodu do jądra. Większość osób potraktuje to jak błąd oprogramowania, ale twórca rootkitu będzie traktował go jak nieudokumentowaną funkcję umożliwiającą załadowanie kodu do jądra. Ze względu na brak dokumentacji takiego rozwiązania to „wejście do jądra" raczej nie zostanie objęte żadnym dochodzeniem wewnątrzsystemowym. Co więcej, nie będzie ono zabezpieczane przez programy typu firewall działające na danym komputerze. Wykrycie takiego włamania może się udać wyłącznie komuś, kto ma doskonale opanowane mechanizmy wstecznej inżynierii kodu.

Rootkit nie jest wirusem

Program wirusa jest właściwie samorozprzestrzeniającym się automatem. I to jest pierwsza różnica: rootkit nie tworzy swoich kopii i nie ma swojej własnej „woli". Rootkit jest całkowicie pod kontrolą osoby, która przeprowadziła atak, a wirus działa „na własną rękę".

W większości przypadków zastosowanie przez włamywacza wirusa było by bardzo głupie i niebezpieczne, ponieważ włamywacz przede wszystkim musi się ukrywać. Po pierwsze, przygotowanie i rozprzestrzenianie wirusa może być nielegalne, a po drugie, większość wirusów i robaków to bardzo „hałaśliwe" i samodzielne programy. Rootkit pozwala atakującemu na pełną kontrolę nad swoim programem. W przypadku usankcjonowanego włamania (na przykład na wniosek sądu) atakujący musi mieć pewność, że zaatakuje tylko określone komputery, żaby nie przekroczyć przyznanych uprawnień i nie zagrozić powodzeniu przedsięwzięcia. Takie operacje wymagają bardzo ścisłej kontroli, więc wykorzystanie wirusa całkowicie nie wchodzi tu w grę.

Możliwe jest takie przygotowanie wirusa lub robaka, żeby rozprzestrzeniał się z wykorzystaniem błędów w oprogramowaniu, które nie są wykrywane przez systemy wykrywania włamań (chodzi tu głównie o błędy typu *zero-day*[16]). Taki robak rozprzestrzeniałby się powoli i byłby bardzo trudny do wykrycia. Mógłby zostać przetestowany w doskonale przygotowanym środowisku laboratoryjnym na modelu środowiska docelowego. Można by w nim zapisać ograniczenie „działania w pewnym obszarze", tak aby nie wymknął się poza pewne ściśle określone granice. W końcu mógłby zawierać też pewien licznik, który po pewnym czasie całkowicie wyłączyłby takiego robaka, przez co nie sprawiałby on żadnych problemów po zakończeniu swojej misji. W dalszej części tego rozdziału rozmawiać będziemy jeszcze na temat systemów wykrywania włamań.

[16] Taką nazwą oznacza się świeżo wykryte błędy, dla których nie ma jeszcze poprawek.

Problem z wirusami

Co prawda rootkit nie jest wirusem, ale techniki stosowane w rootkitach mogą być również wykorzystywane w wirusach. Jeżeli wirus zostałby połączony z rootkitem, to powstałaby wyjątkowo niebezpieczna technologia.

Świat już wielokrotnie się przekonał, do czego zdolne są wirusy. Niektóre z nich rozprzestrzeniły się w ciągu kilku godzin na wiele milionów komputerów.

Najpopularniejszy system operacyjny, Microsoft Windows, ma już długą historię błędów umożliwiających wirusom infekowanie milionów komputerów w internecie. Większość złośliwych hakerów nie opisze wykrytych przez siebie błędów do twórcy oprogramowania. Innymi słowy, jeżeli taki haker znajdzie w systemie Microsoft Windows błąd pozwalający na wykonanie dowolnego kodu, to na pewno firma Microsoft się o tym nie dowie. Błąd tego rodzaju znaleziony w domyślnej instalacji systemu jest jak „klucz do bram królestwa", a poinformowanie o tym błędzie producenta jest równoznaczne z oddaniem tego klucza.

Poznanie technologii rootkitów jest bardzo istotne dla prawidłowej ochrony przed wirusami. Programiści tworzący wirusy już od lat korzystają z tej technologii do usprawnienia swoich produktów. To naprawdę niebezpieczny trend. Zostały już opublikowane algorytmy rozprzestrzeniania[17] wirusów, które pozwalają na spenetrowanie setek, a może i tysięcy komputerów w ciągu godziny. Powstały już techniki pozwalające na zniszczenie systemów komputerowych, a nawet sprzętu. Co więcej, wcale nie ubywa błędów w systemach Windows pozwalających na ich zdalne wykorzystanie. Wirusy korzystające z technologii rootkitów będą coraz trudniejsze do wykrycia i coraz trudniej będzie się przed nimi zabezpieczyć.

Rootkity i błędy w oprogramowaniu

Wykorzystywanie błędów w oprogramowaniu jest bardzo ważnym zagadnieniem, ściśle powiązanym z rootkitami. (Nie będziemy tu jednak opisywać metod złamania oprogramowania i wykorzystania takich błędów. Każdy, kto jest zainteresowany tym zagadnieniem, powinien zapoznać się z książką *Exploiting Software*[18]).

Co prawda rootkit nie jest eksploitem, ale może stać się częścią narzędzia przygotowanego do wykorzystywania błędów w oprogramowaniu (wirusa lub programu typu spyware).

Zagrożenie ze strony rootkitów wzrasta tym bardziej, że cały czas rośnie liczba dostępnych eksploitów. Jeżeli powiemy, że w każdej chwili dostępnych jest przynajmniej sto dziur w najnowszej wersji systemu Microsoft Windows, które można w każdej chwili

[17] N. Weaver, *Warhol Worms: The Potential for Very Fast Internet Plagues*, dostępne na stronie *www.cs.berkeley.edu/~nweaver/warhol.html*.

[18] G. Hoglund i G. McGraw, *Exploiting Software*.

wykorzystać[19]. W większości takie błędy są Microsoftowi znane i powoli łatane przez dział zapewniania jakości w ramach systemu śledzenia błędów[20]. Takie błędy są czasami poprawiane **po cichu**[21].

Niektóre z tak niebezpiecznych błędów w programach znajdowane są przez niezależnych badaczy, którzy niestety nie informują o nich producentów oprogramowania. Takie błędy są śmiertelnymi pułapkami, ponieważ o ich istnieniu nie wie nikt z wyjątkiem atakującego. Oznacza to, że praktycznie nie mamy przed nimi żadnej ochrony (nie istnieje żadna poprawka).

Wiele spośród publicznie eksploitów ujawnionych rok temu, dzisiaj nadal jest bardzo szeroko wykorzystywanych. Nawet jeżeli istnieją poprawki dla tych błędów, to większość administratorów nie wykorzystuje ich do poprawienia swoich systemów we właściwym czasie. Jest to szczególnie niebezpieczne, ponieważ programy wykorzystujące opublikowane błędy pojawiają się już w kilka dni po opublikowaniu informacji o takich błędach wraz z odpowiednimi poprawkami.

Microsoft oczywiście bardzo poważnie traktuje błędy w oprogramowaniu, ale przygotowanie odpowiednich poprawek do tak dużego systemu operacyjnego może zająć dość dużo czasu.

Jeżeli jakiś badacz prześle do Microsoftu informację o błędzie, to najczęściej firma prosi go, aby nie publikował tej informacji do czasu przygotowania poprawki, ponieważ jej przygotowanie jest bardzo drogie i zajmuje sporo czasu. Niektóre z błędów nie są jeszcze poprawione nawet w kilka miesięcy po ich wykryciu.

Można twierdzić, że takie ukrywanie błędów sprawia, że Microsoft nie spieszy się z tworzeniem dla nich poprawek. Dopóki nikt nie wie o istnieniu błędu, nie ma potrzeby, żeby szybko tworzyć taką poprawkę. Firma eEye stara się zatem odpowiednio zapobiegać takim niepożądanym tendencjom i podaje do oficjalnej wiadomości informację o istnieniu błędu, ale nie podaje jego szczegółów.

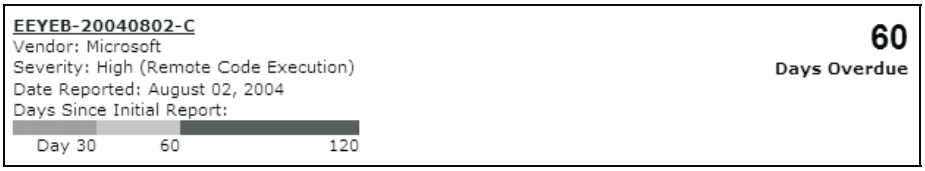
Rysunek 1.2 przedstawia część strony WWW firmy eEye[22], na której zobaczyć można typowy komunikat o wykrytym błędzie. Podano w nim datę przekazania informacji o błędzie do producenta, a także „opóźnienie", z jakim producent przygotował poprawkę tego błędu. Takie opóźnienie wyliczane jest na podstawie założenia, że producent powinien przygotować poprawkę w czasie 60 dni. Jak już wielokrotnie mieliśmy okazję się przekonać, dużym producentom oprogramowania przygotowanie takiej poprawki

[19] Nie możemy dostarczyć żadnego dowodu na takie stwierdzenie, ale zostało ono wywiedzione z naszej wiedzy na temat tego problemu.

[20] Większość innych producentów oprogramowania stosuje podobne metody śledzenia i poprawiania błędów w swoich produktach.

[21] „Ciche poprawianie" oznacza, że błąd jest poprawiony przez aktualizację oprogramowania, ale producent nigdy nie podaje do wiadomości klientów informacji o istnieniu takiego błędu. Taki błąd uznawany jest za „tajny" i nikt nie ma prawa na jego temat rozmawiać. Niestety, jest to praktyka szeroko stosowana przez wielu dużych producentów oprogramowania.

[22] *www.eEye.com*.



Rysunek 1.2. *Metoda stosowana przez firmę eEye przy „wstępnym" publikowaniu błędów*

zajmuje zwykle znacznie dłużej niż wspomniane 60 dni. Wygląda na to, że do tej pory poprawki takich błędów powstawały nawet w ciągu kilku dni, ale tylko w sytuacji, gdy w internecie szalał robak wykorzystujący określony błąd.

Języki z bezpiecznymi typami

Języki programowania posiadające **bezpieczne typy danych** są zdecydowanie lepiej zabezpieczone przed pewnymi rodzajami błędów, takimi jak przepełnienia bufora.

Bez stosowania zabezpieczeń typów dane programu są tylko jednym wielkim oceanem bitów. Program może pobrać dowolną ilość takich bitów i interpretować je w całkowicie dowolny sposób, niezależnie od ich oryginalnego przeznaczenia. Na przykład, jeżeli w pamięci umieścimy ciąg znaków `GARY`, to później może on zostać wykorzystany jako 32-bitowa liczba całkowita (o wartości `0x47415259` lub dziesiętnie `1.195.463.257` — naprawdę duża liczba), a nie jako tekst. Jeżeli w programie może dojść do niepraWidłowej interpretacji danych podanych przez użytkownika, to bardzo prawdopodobne jest powstanie niebezpiecznych błędów.

Z drugiej strony, program napisany w języku z bezpiecznymi typami danych (takim jak Java lub C#[23]) nigdy nie pozwoli na przekształcenia ciągu znaków `GARY` w liczbę całkowitą. W takich językach ciągi znaków zawsze traktowane są jako tekst.

Eksploity nadal są wielkim problemem

Potrzeba zabezpieczania oprogramowania znana była już od długiego czasu, a mimo to eksploity różnych programów nadal stanowią poważny problem. Niestety, wszystkie te problemy mają swoje źródło w samym oprogramowaniu. Trzeba tu jasno powiedzieć, że większość programów wcale nie jest bezpieczna. Firmy takie jak Microsoft coraz bardziej starają się projektować bezpieczne oprogramowanie, ale kod aktualnie używanych systemów operacyjnych nadal tworzony jest w językach C lub C++, których natura powoduje wprowadzanie do kodu wielu dziur w zabezpieczeniach. To właśnie te języki stanowią główne źródło problemów znanych jako **błędy przepełnienia bufora**. Takie błędy są najpoważniejszą słabością tworzonych dzisiaj programów i umożliwiły przygotowanie tysięcy najróżniejszych eksploitów. Trzeba jednak pamiętać, że są to tylko błędy, czyli coś, co można poprawić[24].

[23] Języka C# nie należy mylić z językami C lub C++.

[24] Co prawda błędy przepełnienia bufora nie ograniczają się wyłącznie do języków C i C++, to jednak właśnie w tych językach najtrudniej jest zapewnić stosowanie praktyk bezpiecznych technik programowania. Języki te nie mają bezpiecznych typów danych (będzie o tym mowa w dalszej części rozdziału), korzystają z funkcji wbudowanych, które mogą spowodować przepełnienie bufora, a na dodatek są bardzo uciążliwe w debugowaniu.

Kiedyś błędy przepełnienia bufora stracą na znaczeniu, choć na pewno nie nastąpi to w najbliższej przyszłości. Co prawda zdyscyplinowany programista może napisać kod, który nie będzie zawierał tego rodzaju błędów (i to niezależnie od języka programowania; bezpieczne mogą być nawet programu tworzone w asemblerze), ale niestety, większość programistów nie wykazuje się aż taką skrupulatnością. Aktualnie w czasie produkcji oprogramowania firmy starają się narzucić programistom praktyki bezpiecznego tworzenia kodu oraz stosować automatyczne narzędzia do przeglądania kodu i wyszukiwania ewentualnych usterek. Takimi narzędziami posługuje się na przykład Microsoft[25].

Narzędzia do automatycznego przeglądania kodu mogą wychwycić część błędów, ale z pewnością nie są w stanie znaleźć wszystkich. Większość programów komputerowych jest na tyle skomplikowana, że ich automatyczne, dokładne przetestowanie może być naprawdę trudne. Niektóre programy mogą mieć zbyt wiele stanów, żeby można je wszystkie sprawdzić[26]. Co ciekawe, niektóre programy mogą mieć więcej różnych stanów niż jest cząsteczek we wszechświecie[27]. Ze względu na tak ogromną złożoność programów bardzo trudno jest choćby ogólnie określić stopień bezpieczeństwa programu komputerowego.

Zastosowanie języków programowania z bezpiecznymi typami (takich jak Java lub C#) niemal całkowicie wyeliminuje ryzyko występowania błędów przepełnienia bufora. Co prawda języki z bezpiecznymi typami danych nie gwarantują całkowitego bezpieczeństwa programów, ale zdecydowanie zmniejszają ryzyko powstawania błędów przepełnienia bufora, błędów konwersji znaku oraz błędów przepełnienia liczb całkowitych (więcej na ten temat w ramce ze strony ???). Niestety, takie języki nie mogą się równać z językami C i C++ pod względem wydajności i dlatego systemy Microsoft Windows nawet w swoich najnowszych wersjach nadal tworzone są w językach C i C++. Twórcy systemów osadzonych (ang. *embedded systems*) zaczynają już coraz częściej stosować języki z bezpiecznymi typami, ale i ten proces postępuje bardzo powoli. Co więcej, miliony istniejących już systemów raczej nie zostaną wymienione w najbliższym czasie. To wszystko oznacza, że tradycyjne już eksploity oprogramowania będą nam towarzyszyły jeszcze przez długie lata.

---

[25] Na przykład narzędzia PREfix i PREfast zostały przygotowane i opublikowane przez Jona Pincusa, pracującego w centrum badawczym Microsoftu. Proszę przejrzeć stronę *http://research.microsoft.com/users/jpincus/*.

[26] „Stan" można traktować jak wewnętrzną konfigurację oprogramowania. Za każdym razem, gdy program wykonuje jakąś operację, zmienia się jego stan. Oznacza to, że większość programów może przyjmować naprawdę gigantyczną liczbę różnych stanów.

[27] Proszę sobie wyobrazić liczbę możliwych permutacji pewnego ciągu liczb binarnych. Za przykład niech posłuży nam program o wielkości 160 MB, który swój stan przechowuje w 16 MB pamięci (mniej więcej 10% całkowitej wielkości programu). Taki program może mieć teoretycznie nawet 2^16777216 różnych stanów, co jest liczbą o wiele, wiele większą od liczby cząsteczek we wszechświecie (zwykle ocenia się ją na mniej więcej 10^80). [Dziękujemy Aaronowi Bornsteinowi za wyjaśnienie tego przykładu.]

Ofensywne technologie rootkitów

Dobry rootkit powinien umieć ominąć wszelkie zabezpieczenia systemowe, takie jak firewalle lub systemy wykrywania włamań (*Intrusion Detection Systems* — IDS). Istnieją dwa podstawowe typy systemów IDS: sieciowe (*Network-based IDS* — NIDS) i stanowiskowe (*host-based IDS* — HIDS). Czasami systemy HIDS przygotowywane są tak, żeby zatrzymywać ataki w czasie ich trwania. Są to systemy „obrony aktywnej", które niejednokrotnie określa się mianem stanowiskowych systemów przeciwdziałania włamaniom (*host-based Intrusion Prevention Systems* — HIPS). W ramach uproszczenia systemy takie również określać będziemy skrótem HIPS.

HIPS

Technologia systemów HIPS może zostać przygotowana samodzielnie, ale można też kupić ją w postaci gotowych programów. Oto przykładowe programy wykorzystujące technologię HIPS:

- Blink (eEye Digital Security, *www.eEye.com*),
- Integrity Protection Driver (IPD, Pedestal Software, *www.pedestal.com*),
- Entercept (*www.networkassociates.com*),
- Okena Storm Watch (teraz nazywa się Cisco Security Agent, *www.cisco.com*),
- LIDS (Linux Intrusion Detection System, *www.lids.org*),
- WatchGuard ServerLock (*www.watchguard.com*).

Największym zagrożeniem dla rootkitów są systemy HIPS. Takie systemy są w stanie wykryć rootkit w czasie jego instalowania, a także przechwycić połączenia, jakie rootkit utrzymuje w sieci. Wiele systemów HIPS wykorzystuje technologie obejmujące jądro systemu i w związku z tym może dokładnie monitorować wszystkie zachowania systemu operacyjnego. Można powiedzieć, że systemy HIPS są *antyrootkitami*. Oznacza to, że wykrywają one i blokują praktycznie wszystkie operacje, jakie rootkit może podejmować w systemie. Jeżeli chcemy zastosować rootkit w systemie operacyjnym chronionym przez system HIPS, to tak naprawdę mamy tylko dwa rozwiązania: w jakiś sposób obejść mechanizmy systemu HIPS albo wybrać mniej wymagający cel.

W rozdziale 10. omawiać będziemy sposoby tworzenia systemów HIPS. W rozdziale tym przedstawimy też przykłady kodu antyrootkitowego. Taki kod pozwala poznać metody obchodzenia zabezpieczeń tworzonych przez systemy HIPS, a poza tym ułatwi też tworzenie własnego systemu ochrony przed rootkitami.

NIDS

Sieciowe systemy IDS (NIDS) również stanowią zagrożenie dla rootkitów, ale dobrze zaprojektowany rootkit może bardzo skutecznie unikać wykrycia przez systemy NIDS. Teoretycznie analiza statystyczna jest w stanie wykryć utajone kanały komunikacyjne,

ale w rzeczywistości udaje się to niezwykle rzadko. Połączenia sieciowe tworzone przez rootkit najczęściej ukrywają się w bardzo niewinnie wyglądających pakietach, a wszystkie ważne dane są przed wysłaniem szyfrowane. Większość funkcjonujących systemów NIDS zajmuje się wyłącznie bardzo dużymi strumieniami danych (nawet do 300 MB/s), a więc te niewielkie ilości danych przesyłanych przez rootkit zwykle pozostają niezauważone. Systemy NIDS mają znacznie większą szansę wykrycia rootkitów wykorzystujących ogólnie znane eksploity[28].

Obchodzenie systemów IDS i IPS

W celu ominięcia firewalli i oprogramowania typu IDS i ISP można zastosować dwa rozwiązania: aktywne i pasywne. Jeżeli jednak rootkit ma być naprawdę skuteczny, to konieczne jest zastosowanie obu rozwiązań. Rozwiązania aktywne działają w czasie pracy rootkitu i mają na celu uniknięcie wykrycia. Na wypadek, gdyby w kimś narosły jakieś podejrzenia, rozwiązania pasywne mają na celu zmylenie ewentualnego dochodzenia.

Działania aktywne polegają na modyfikacjach systemowego sprzętu i jądra ukierunkowanych na przeszkadzaniu w pracy systemom wykrywania i zapobiegania włamaniom. Takie działania najczęściej potrzebne są do wyłączenia programów typu HIPS (takich jak Okena lub Entercept). Mówiąc ogólnie, działania aktywne stosowane są wobec programów działających w systemie i próbujących wykryć rootkit. Mogą też posłużyć rootkitowi do wyeliminowania zagrożenia wykrycia ze strony narzędzi administracyjnych. Bardziej złożone mechanizmy mogą spowodować wyłączenie dowolnego narzędzia skanującego lub zabezpieczającego. Jeden z takich mechanizmów może wyszukiwać w systemie skanerów antywirusowych i po znalezieniu — wyłączać.

Działania pasywne polegają na wprowadzaniu zamieszania w systemach przechowywania i przesyłu danych. Przykładem może być tu szyfrowanie danych przed ich zapisaniem w systemie plików. Bardziej zaawansowana metoda polega na przeniesieniu klucza rozszyfrowującego z systemu plików do pamięci nieulotnej, takiej jak RAMK lub EEPROM. Kolejnym działaniem pasywnym jest wykorzystywanie utajonych kanałów komunikacyjnych pozwalających na wysyłanie danych przez sieć.

Rootkit nie powinien zostać wykryty przez skaner antywirusowy. Takie skanery nie pracują wyłącznie w czasie pracy systemu, ale wykorzystywane są też do skanowania dysków w trybie „offline”. Dyski czasami są kontrolowane na obecność wirusów w specjalnych laboratoriach. W takich przypadkach rootkit musi być na tyle dobrze ukryty w systemie plików, żeby skaner mimo wszystko go nie wykrył.

[28] Używając publicznego eksploitu, atakujący może udawać zachowanie znanego już wcześniej robaka (na przykład robaka Blaster). Większość administratorów zajmujących się bezpieczeństwem systemów uzna takie zachowania za typowy atak robaka i przez to nie rozpozna właściwego ataku.

Pomijanie narzędzi wykrywających

W sytuacji idealnej rootkit nigdy nie zostanie wykryty przez narzędzia skanujące system. Problem ten jest jednak niezwykle trudny do rozwiązania. Do skanowania twardych dysków stosowane są bardzo zaawansowane narzędzia. Niektóre z nich, takie jak Encase[29], „szukają złych elementów" w sytuacjach, gdy w danym systemie podejrzewana jest obecność rootkitu lub wirusa. Inne narzędzia, takie jak Tripwire, „stoją na straży" i mają za zadanie ochronić system przed infekcją.

Wprawiony użytkownik programów pokroju Encase będzie szukał na dysku określonych wzorców bajtów. Takie narzędzia potrafią przeszukać cały dysk, a nie tylko zapisane na nim pliki. Skanowane są również pliki usunięte oraz przestrzeń niewykorzystana. W takiej sytuacji rootkit może uniknąć wykrycia tylko przez unikanie stosowania łatwo rozpoznawalnych wzorców danych. Bardzo przydatnym narzędziem jest tutaj steganografia. Można stosować też szyfrowanie, ale niestety, narzędzia mierzące stopień losowości danych są w stanie zlokalizować zaszyfrowane bloki. Poza tym w przypadku stosowania szyfrowania pewna część rootkitu (ta odpowiedzialna za rozszyfrowanie) musi być niezaszyfrowana. Ochronę takiego kodu można zapewnić przez stosowanie polimorficznych algorytmów rozszyfrowujących. Trzeba przy tym pamiętać, że skuteczność każdego narzędzia jest ściśle uzależniona od umiejętności osób tworzących i obsługujących to narzędzie. Jeżeli wymyślimy sposób ukrywania rootkitu, o którym nie pomyśleli oni, to najprawdopodobniej uda się nam uniknąć wykrycia.

Narzędzia przygotowujące kryptograficzne „odciski palca" wszystkich plików w systemie (na przykład tripwire) muszą sobie przygotować bazę danych takich „odcisków" na podstawie czystego systemu. Teoretycznie, jeżeli kopia czystego systemu (czyli kopia całego twardego dysku) zostanie wykonana jeszcze przed zainstalowaniem rootkitu, to będzie można wykonać analizę porównawczą przygotowanego obrazu dysku z jego aktualnym stanem. W ten sposób wykryte zostaną wszystkie zmiany w stosunku do pierwotnego obrazu dysku. Jedna z takich zmian na pewno spowodowałoby zainstalowanie rootkitu, ale na dysku na pewno będzie więcej modyfikacji. Każdy system operacyjny z czasem podlega różnym zmianom. Oznacza to, że nasz rootkit może uniknąć wykrycia, jeżeli tylko uda mu się ukryć w typowym „szumie" zmian systemu plików. Poza tym tego rodzaju narzędzia sprawdzają wyłącznie pliki, a niektóre z nich mogą kontrolować tylko **wybrane** pliki — najprawdopodobniej te, które zostały uznane za najistotniejsze. Programy te nie zajmują się danymi znajdującymi się w nietypowych miejscach (na przykład w uszkodzonych sektorach dysku). Co więcej, bardzo często ignorowane są tymczasowe pliki systemowe, co pozostawia nam wiele miejsc, w których można się bezpiecznie ukryć.

Jeżeli atakujący naprawdę obawia się, że administrator atakowanego systemu ma przygotowane „odciski palca" wszystkich możliwych plików i przez to rootkit zostanie łatwo wykryty, to powinien unikać stosowania systemu plików. Rootkit można w całości zainstalować w pamięci i w ogóle nie używać dysku twardego. Wadą takiego rozwiązania jest oczywiście fakt, że po ponownym uruchomieniu systemu rootkit całkowicie zniknie z pamięci ulotnej.

[29] *www.encase.com*

Można też zastosować naprawdę ekstremalne rozwiązania i próbować zainstalować rootkit w miejsce BIOS-u komputera albo innej dostępnej karcie pamięci Flash.

Wnioski

Rootkity pierwszej generacji były najnormalniejszymi programami. Dzisiejsze rootkity najczęściej są konstruowane w postaci sterowników urządzeń. W nadchodzących latach powstające rootkity mogą próbować modyfikować lub instalować się w mikrokodzie procesora. Niewykluczone, że będą one mogły istnieć wyłącznie w ramach mikroukładów składających się na dany komputer. Nie jest na przykład wykluczone, że twórcom rootkitów uda się tak zmodyfikować obraz układów FPGA (Field Programmable Gate Array), żeby utworzyć w nich dla siebie tylne wejście[30]. Oczywiście rootkity tego rodzaju tworzone będą w celu zaatakowania bardzo konkretnego celu. Bardziej rozpowszechnione będą oczywiście te rootkity, które wykorzystywać będą najogólniejsze usługi udostępniane przez system operacyjny.

Technologia rootkitów ukrywających się w ramach układów FPGA nie nadaje się do zastosowania w robakach internetowych. W tych szkodnikach całkowicie nie sprawdzają się ataki uzależnione od konkretnej konfiguracji sprzętowej. Robaki sieciowe najlepiej rozprzestrzeniają się w ogromnych i jednolitych środowiskach. Innymi słowy, robaki najlepiej czują się w sytuacji, gdy na wszystkich komputerach działa takie samo oprogramowanie. W świecie rootkitów ukierunkowanych na konkretne rozwiązania sprzętowe istnieje zbyt wiele drobnych różnic pomiędzy komputerami, które uniemożliwiają masowe atakowanie komputerów. Bardziej prawdopodobne jest to, że takie ataki będą przeprowadzane na konkretnie wybrane cele, które atakujący może szczegółowo przeanalizować i przygotować rootkit idealnie pasujący do wybranej ofiary.

Rootkity będą wykorzystywać eksploity do czasu, aż te nie zostaną całkowicie wyeliminowane. Współpraca rootkitu z eksploitami jest czym całkowicie naturalnym. Trzeba jednak zaznaczyć, że rootkity istniałyby nawet wtedy, gdybyśmy nigdy nie słyszeli o eksploitach.

W ciągu kilku następnych dziesięcioleci najprawdopodobniej dzisiejszy „król wszystkich eksploitów", czyli błędy przepełnienia bufora, zostanie całkowicie wyeliminowany. Zaawansowane języki z bezpiecznymi typami, zaawansowane kompilatory oraz technologie maszyn wirtualnych praktycznie całkowicie wyeliminują zagrożenie błędów przepełnienia bufora. Z całą pewnością będzie to dotkliwy cios dla wszystkich tych, którzy w ten sposób zdalnie próbują atakować systemy. Nie oznacza to jednak, że w ten sposób znikną również eksploity. Nowa kategoria eksploitów opierać się będzie zapewne na błędach logiki znajdujących się w programach, a nie na słabościach architektury.

[30] Zakładamy teraz, że w takich układach będzie dość miejsca (co oznacza odpowiednią liczbę bramek) na dopisanie do układu nowych funkcji. Producenci sprzętu próbują jednak oszczędzać na każdym elemencie komputera, więc układy FPGA są zwykle tylko tak duże jak zapisywana do nich aplikacja. Oznacza to, że w układzie może brakować miejsca na nowe funkcje, a zatem rootkit, instalując się w tak niewielkiej przestrzeni, będzie prawdopodobnie musiał usunąć z układu pewne funkcje.

Rootkity będą istniały niezależnie od tego, czy nadal będzie istniała możliwość zdalnego stosowania eksploitów. Można je umieszczać w systemach na wielu etapach, od momentu tworzenia systemu, aż po jego instalację. Jak długo będzie istniała ludzkość, znajdą się też osoby pragnące szpiegować innych. Oznacza to, że rootkity na zawsze będą częścią technologii komputerowych. Programy tworzące w systemie „tylne wejście" i technologie sabotażu systemów operacyjnych są po prostu ponadczasowe!