WOJCIECH SMOL

Zacieranie śladów po włamaniu

Stopień trudności

Dla crackera włamującego się do cudzego systemu informatycznego najważniejsze jest... skuteczne zatarcie śladów po całej operacji. Nawet najbardziej spektakularne włamanie przy użyciu stworzonego przez siebie exploita nie przyniesie włamywaczowi wiele korzyści, jeśli nazajutrz pojawią się u niego agenci CBŚ.

Z ARTYKUŁU DOWIESZ SIĘ

- o metodach zacierania śladów stosowanych przez crackerów po wykonanym włamaniu,
- o narzędziach umożliwiających modyfikowanie logów systemowych,
- o słabościach procedur stosowanych w analizie powłamaniowej,
- o metodach wykrywania obecności nietypowych zabezpieczeń w systemach operacyjnych *nix,
- o metodach oszukiwania systemów HIDS.

CO POWINIENEŚ WIEDZIEĆ

- znać podstawy analizy powłamaniowej,
- znać podstawy administracji systemami operacyjnymi *nix,
- znać zasadę działania podstawowych usług znajdujących się w systemie *nix,
- znać podstawy języka programowania C.

Zapraszam do poznania kilku technik stosowanych przez crackerów w celu zmylenia śledczych przeprowadzających analizę powłamaniową.

Słabości analizy powłamaniowej

Analiza powłamaniowa jest w przypadku przestępstw komputerowych tym, czym jest analiza detektywistyczna w przypadku przestępstw konwencjonalnych. Zadania obu procesów są bardzo zbliżone, należy ustalić: czas, miejsce oraz sprawcę nielegalnej czynności. W obu przypadkach najważniejsze jest wykorzystanie wszelkich dostępnych (często szczątkowych) informacji oraz połączenie ich w logiczną całość. Metody pracy detektywa komputerowego są oczywiście zupełnie odmienne od metod stosowanych przez śledczych ścigających zwykłych przestępców, jednak oba procesy mają jeszcze jedną cechę wspólną. Zarówno zwykły przestępca, jak i komputerowy włamywacz mogą utrudniać pracę ścigających ich detektywów, starając się zatrzeć wszelkie ślady swej działalności. Detektywi specjalizujący się w analizie powłamaniowej muszą więc znać metody zacierania śladów stosowane przez komputerowych przestępców, gdyż bez tego ich praca może spełznąć na niczym lub co gorsza, podejrzenia mogą zostać skierowane na niewinną osobę...

Niestety, zazwyczaj wszelkie publikacje dotyczące analizy powłamaniowej skupiają się na samym procesie analizy logów i plików, metodach sprawdzania działalności użytkowników w systemie operacyjnym, wykrywaniu podejrzanych procesów i połączeń sieciowych. Zazwyczaj zupełnie pomija się fakt, że przecież zapisy w logach mogły zostać po włamaniu celowo zmanipulowane przez crackera, który chce w ten sposób utrudnić przyszłe śledztwo. Prowadząc rzetelną analizę powłamaniową, nie wystarczy więc ustalić, że włamania dokonała osoba łącząca się z serwerem poprzez protokół SSH i korzystająca z zapisanego w logach adresu IP. Należy jeszcze ustalić, czy to, co ukazuje nam proces analizy, to zapis prawdziwych wydarzeń, czy może tylko scenariusz napisany specjalnie dla nas przez samego włamywacza...

Z kim mamy do czynienia?

Ogólnie rzecz biorąc, mamy do czynienia z dwoma rodzajami komputerowych włamywaczy. Pierwsza grupa włamywaczy to tzw. *script kiddies*. Włamywacze ci, pogardliwie określani jako „skrypciarze", stanowią stosunkowo małe zagrożenie ze względu na brak zaawansowanej wiedzy. Jeśli dany system informatyczny jest utrzymywany zgodnie z podstawowymi zasadami bezpieczeństwa (prawidłowa konfiguracja, aktualność systemów operacyjnych i oprogramowania), crackerzy korzystający z gotowych, często przestarzałych exploitów nie powinni stanowić dla nas większego zagrożenia. Jeśli zdarzy się jednak, że skrypciarz uzyska dostęp do naszego systemu, specjalista przeprowadzający analizę powłamaniową prawdopodobnie będzie miał dość łatwe zadanie. Można z dużym prawdopodobieństwem założyć, że

script kiddie nie będzie stosował żadnego zacierania śladów w systemie, do którego dostęp udało mu się uzyskać. Nie dlatego, że nie chce uniknąć odpowiedzialności za swój czyn, lecz dlatego, że jego wiedza jest niewystarczająca. Skrypciarze zazwyczaj nie znają na tyle dobrze atakowanych systemów operacyjnych, rozwiązań sieciowych lub oprogramowania, aby skutecznie zatrzeć ślady swej obecności w systemie.

Sytuacja wygląda zupełnie inaczej w przypadku profesjonalnych crackerów. Wysokiej klasy cracker potrafi sam przeanalizować serwer pod kątem luk w zabezpieczeniach, stworzyć własne exploity, a nawet definiować zupełnie nowe metody ataku. W takim przypadku nawet prawidłowo zabezpieczony system informatyczny może paść ofiarą ataku – wszystko zależy tu tylko od determinacji i talentu crackera. W takim przypadku również analiza powłamaniowa musi być prowadzona ze szczególną uwagą. Można być prawie pewnym, że zaawansowany cracker będzie się starał zatrzeć po włamaniu wszelkie ślady, które mogłyby doprowadzić do niego śledczych. Jako wysokiej klasy specjalista, prawdopodobnie zna doskonale atakowany system operacyjny i wie, gdzie mogły zostać jakieś ślady jego działalności. Zobaczmy więc, jakimi możliwościami w zakresie zacierania śladów dysponują crackerzy. Przedstawionych poniżej metod nie należy jednak traktować jako instruktażu dla początkujących crackerów. Celem artykułu jest zwrócenie uwagi osób zajmujących się amatorsko lub profesjonalnie analizą powłamaniową na problem zacierania bądź preparowania śladów przez komputerowych włamywaczy.

Logi

Jak nietrudno się domyślić, doświadczony cracker będzie się starał przede wszystkim zatrzeć ślady, które zostały po jego działalności w logach zaatakowanego systemu operacyjnego. W systemach operacyjnych typu *nix włamywacz prawdopodobnie rozpocznie zacieranie śladów pozostałych w trzech bardzo ważnych z jego punktu widzenia logach systemowych, mianowicie:

- WTMP – log zawierający informacje o każdym zalogowaniu/wylogowaniu użytkownika wraz z czasem każdego zdarzenia oraz adresem zdalnego hosta,
- UTMP – log zawierający informacje o obecnie zalogowanych użytkownikach,
- LASTLOG – log zawierający informacje o ostatnim logowaniu użytkowników (oraz o adresach, z których nastąpiło logowanie).

Każde logowanie do systemu za pośrednictwem protokołów: telnet, ssh, ftp, rlogin itd. jest zapisywane w powyższych logach i na ich podstawie osoba wykonująca analizę powłamanową jest w stanie ustalić:

- dokładny czas włamania,
- adres hosta wykonującego atak,
- jak długo trwała wroga sesja i na tej podstawie oszacować możliwy zakres działań intruza.

Wiedzą o tym oczywiście doświadczeni crackerzy i po wykonanym ataku będą starali się usunąć ślady własnych połączeń z logów. Jeśli logi zostały po włamaniu po prostu skasowane, oznacza to, że nie mamy do czynienia za zbyt błyskotliwym crackerem. Usunięcie logów nie jest najlepszym rodzajem zacierania śladów – wręcz przeciwnie. Jest to raczej najlepszy sposób na zwrócenie uwagi administratorów atakowanego systemu, że wydarzyło się coś złego.

Doświadczony cracker użyje raczej specjalnego programu – modyfikatora logów. Włamywacz nie musi tworzyć własnego programu, dostępnych jest wiele powszechnie znanych programów umożliwiających automatyczną modyfikację systemowych wpisów. Przykładowo portal *Packet Storm* oferuje kody źródłowe kilkudziesięciu programów tego typu przeznaczonych dla systemów operacyjnych *nix. Crackerskie poradniki często polecają program ZAP lub ZAP2 jako najlepszy modyfikator logów. Działanie tego programu polega na nadpisaniu zerami daty ostatniego logowania użytkownika. Tego typu zerowe wpisy rzucają się jednak w oczy niewiele mniej niż całkowity brak logów, wobec czego nie jest to jednak najlepszy program, jaki może w tej sytuacji zastosować włamywacz. Znacznie więcej możliwości w zakresie użytecznego dla włamywaczy modyfikowania logów systemowych posiada program CLOAK oraz CLOAK2. Jako przykład praktycznego zastosowania modyfikatorów logów chciałbym przedstawić użycie programu CLOAK w celu ukrycia obecności zalogowanego użytkownika w systemie. Kody źródłowe (w języku C) tego programu dostępne są na wspomnianej już witrynie *Packet Storm*. Programy tego typu, tworzone przecież przez crackerów dla crackerów, nie są zazwyczaj wyposażone w żadną instrukcję lub podręcznik użytkownika. W celu zrozumienia sposobu działania programu oraz poznania możliwych parametrów wywołania pozostaje nam analiza samego kodu źródłowego. Zapoznając się z kodem źródłowym, można zauważyć, że ścieżki do plików UTMP oraz LASTLOG zostały przez autora zdefiniowane *na sztywno*, odpowiednio jako: */etc/utmp* oraz */var/adm/lastlog*. Ścieżki te należy więc w źródle zmodyfikować, odpowiednio do docelowego systemu *nix. Jako że przetestowałem działanie programu w systemie CentOS 5, ustawiłem następujące ścieżki do plików:

- */var/run/utmp*,
- */var/log/lastlog*.

Analizując kod źródłowy, dowiemy się, że w celu ukrycia obecności użytkownika w systemie operacyjnym należy uruchomić skompilowany program z parametrem *cloakme*. Po dokonaniu drobnej modyfikacji źródeł oraz zapoznaniu się z możliwościami i

```
[root@localhost ~]# gcc cloak.c
cloak.c: In function âmainâ:
cloak.c:27: warning: incompatible implicit declaration of built-in function âpri
cloak.c:31: warning: incompatible implicit declaration of built-in function âpri
cloak.c:32: warning: incompatible implicit declaration of built-in function âexi
cloak.c:36: warning: incompatible implicit declaration of built-in function âpri
cloak.c:37: warning: incompatible implicit declaration of built-in function âexi
cloak.c:50: warning: incompatible implicit declaration of built-in function âmem
cloak.c:67: warning: incompatible implicit declaration of built-in function âstr
[root@localhost ~]# w
 17:28:22 up  5:49,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/1    192.168.1.191    17:28    0.00s  0.10s  0.01s w
[root@localhost ~]# ./a.out cloakme
You are now cloaked
[root@localhost ~]# w
 17:28:36 up  5:50,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
[root@localhost ~]#
```

Rysunek 1. *CLOAK – ukrywanie obecności użytkownika w systemie *nix*

parametrami programu można wypróbować jego działanie, przykładowa procedura może być następująca:

- *gcc cloak.c* – kompilacja odpowiednio zmodyfikowanych źródeł programu,
- *./a.out cloakme* – uruchomienie skompilowanego programu z parametrem nakazującym ukrycie obecności naszego bieżącego użytkownika w systemie operacyjnym,
- w przypadku udanego ukrycia użytkownika program zwraca komunikat: *You are now cloaked*,
- polecenie *w* pozwala zweryfikować poprawność działania programu – nasz użytkownik nie powinien pojawić się w wynikach generowanych przez ten program.

W testowym systemie CentOS 5 program zadziałał prawidłowo, program *w* przestał informować o obecności użytkownika *root* w systemie (Rysunek 1).

Kończąc rozważania na temat modyfikacji logów WTMP, UTMP oraz LASTLOG, należy jeszcze zauważyć, że w nowoczesnych systemach typu *nix tylko administrator systemu dysponuje możliwością ich edycji. Jednak jeśli crackerowi udało się włamać tylko na konto niepozwalające na zmianę zawartości pliku logu, również może zastosować pewne proste metody zacierania śladów.

Przykładowa procedura nadpisania informacji o ostatnim logowaniu użytkownika (i wykorzystywanym przez niego adresie) może być następująca:

- *ssh user@adres_atakowanego_serwera* – włamanie do serwera na konto użytkownika user (w tym momencie log LASTLOG zawiera adres hosta włamywacza),
- *ssh user@localhost* – ponowne zalogowanie się na konto użytkownika user (w tym momencie log LASTLOG zawiera jako adres hosta, z którego zalogował się *user*, wpis *localhost*, co wygląda bardzo niewinnie w stosunku do obcego adresu IP, z którego dostał się do komputera włamywacz).

Sprytny cracker prawdopodobnie nie zapomni również o usunięciu śladów, które pozostawił w plikach historii powłoki systemowej. Poszczególne powłoki tworzą następujące (ukryte!) pliki historii w katalogu domowym użytkownika:

- *sh* – *.sh_history*,
- *csh* – *.history*,
- *ksh* – *.sh_history*,
- *bash* – *.bash_history*.

Przy wyszukiwaniu logów historii i innych modyfikowanych przez swą działalność plików, cracker może się posłużyć prostym poleceniem: *ls -alt*. Pozwoli to na wyświetlenie wszystkich (również ukrytych) plików, w kolejności ich modyfikacji. Jeśli polecenie to zostanie wykonane w katalogu domowym użytkownika, na którego koncie działa włamywacz, z pewnością jednym z pierwszych wyświetlonych (ostatnio modyfikowanych) plików będzie historia powłoki (Rysunek 2).

W typowej sytuacji, w systemach typu *nix logi znajdują się zazwyczaj w jednej z następujących lokalizacji:

- */var/log*,
- */var/adm*,
- */usr/adm*.

Oczywiście cracker będzie się starał odnaleźć wszystkie logi, nie tylko te powszechnie znane. Może się przecież zdarzyć, że w danym systemie logowanie odbywa się w sposób niekonwencjonalny, a same pliki logów mogą być ukryte. Odnalezienie wszelkich logów może jednak ułatwić prosty program LSOF (ang. *LiSt Open Files*). Wystarczy wykonać komendę *lsof*, aby wyświetlić wszystkie otwarte pliki, wraz z dodatkowymi informacjami na ich temat, między innymi:

- nazwą programu, który używa pliku,
- PID-em (ang. *Process Identifier*),
- typem pliku,
- rozmiarem pliku,
- nazwą węzła.

Wszelkie pliki logów zapisane w formacie tekstowym mogą być przez intruza stosunkowo łatwo zmanipulowane, wystarczy przecież choćby użycie dowolnego edytora tekstu. Sprawa komplikuje się dla crackera w momencie, gdy natrafi on na nietypowe logi w postaci binarnej. Jednak i w takiej sytuacji utalentowanemu włamywaczowi może udać się zatrzeć ślady swej działalności. W przypadku, gdy uda mu się zidentyfikować oprogramowanie logujące, wystarczy przeanalizować jego kody źródłowe i odnaleźć odpowiedni plik nagłówkowy, definiujący struktury danych stosowane w pliku. Następnie wystarczy modyfikacja plików źródłowych wspomnianych wcześniej modyfikatorów logów (ZAP, CLOAK), tak by były one w stanie działać z odnalezionym, nietypowym plikiem logów. Cała procedura jest jednak dość czasochłonna i wymaga sporych umiejętności, więc tego typu działań można się spodziewać tylko po bardzo

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# date
Mon Jan 19 18:08:43 EST 2009
[root@localhost ~]# ls -alt
total 180
-rw-------  1 root root 3154 Jan 19 18:05 .bash_history
drwxr-x--- 18 root root 4096 Jan 19 17:28 .
-rwxr-xr-x  1 root root 6353 Jan 19 17:28 a.out
drwx------  2 root root 4096 Jan 19 17:23 .gconfd
-rw-r--r--  1 root root 1969 Jan 19 17:05 .recently-used.xbel
-rw-------  1 root root   35 Jan 19 16:37 .lesshst
-rw-r--r--  1 root root 1648 Jan 19 16:31 cloak.c
-rw-r--r--  1 root root  607 Jan 19 16:25 .xsession-errors
-rw-r--r--  1 root root 1648 Jan 19 16:24 cloak.c~
drwxr-xr-x  2 root root 4096 Jan 19 16:15 Desktop
drwx------  6 root root 4096 Jan 19 12:01 .gnome2
drwx------  3 root root 4096 Jan 19 11:47 .config
drwx------  3 root root 4096 Jan 19 11:45 .thumbnails
drwx------  4 root root 4096 Jan 19 11:42 .gconf
-rw-------  1 root root  378 Jan 19 11:42 .ICEauthority
drwxr-xr-x 23 root root 4096 Jan 19 11:40 ..
-rw-r--r--  1 root root    0 Jan 18 17:47 find
drwx------  2 root root 4096 Jan 18 14:50 .ssh
drwxr-xr-x  3 root root 4096 Jan 16 10:13 .nautilus
drwx------  3 root root 4096 Jan 14 15:22 .mozilla
drwxr-xr-x  2 root root 4096 Jan 14 15:22 .gstreamer-0.10
drwx------  3 root root 4096 Jan 14 15:22 .metacity
drwxr-x---  2 root root 4096 Jan 14 15:22 .eggcups
drwxr-xr-x  3 root root 4096 Jan 14 15:22 .gnome
drwxr-xr-x  3 root root 4096 Jan 14 15:22 .redhat
drwx------  2 root root 4096 Jan 14 15:22 .Trash
drwx------  2 root root 4096 Jan 14 15:22 .gnome2_private
-rw-r--r--  1 root root   81 Jan 14 15:22 .gtkrc-1.2-gnome2
-rw-------  1 root root   26 Jan 14 15:22 .dmrc
-rw-------  1 root root 1673 Jan 14 10:03 anaconda-ks.cfg
-rw-r--r--  1 root root   24 Jan  6  2007 .bash_logout
-rw-r--r--  1 root root  191 Jan  6  2007 .bash_profile
-rw-r--r--  1 root root  176 Jan  6  2007 .bashrc
```

Rysunek 2. *Polecenie ls -alt pozwoli crackerowi odnaleźć pliki, które zostały przez niego ostatnio zmodyfikowane*

doświadczonych crackerach. Komputerowy włamywacz, chcący zatrzeć ślady swej działalności, szczególną uwagę poświęci na pewno szeroko wykorzystywanemu w systemach *nix programowi *syslog*. Większość programów przeznaczonych do pracy w systemach *nix korzysta właśnie z logowania poprzez funkcje *syslog'a*.

Cracker, chcący zatrzeć ślady generowane przez ten program, powinien przede wszystkim zapoznać się z jego plikiem konfiguracyjnym (zazwyczaj: */etc/syslog.conf*). Na tej podstawie można stwierdzić, do jakich plików wiadomości są zapisywane. Następnie wystarczy już odpowiednie zmodyfikowanie tych zbiorów. W tym miejscu należy pamiętać, że wiadomości mogą być przesyłane nie tylko do plików, ale również do użytkowników, terminali (*tty*) oraz na konsolę (*/dev/console*). Jeśli komunikaty przesyłane są na przykład na konsolę, cracker może zalać ją spreparowanymi przez siebie informacjami, co spowoduje przewinięcie poza ekran właściwych komunikatów.

Z przedstawionych przykładów wynika jasno, że wszelkie ślady swej obecności, powstałe lokalnie w atakowanym systemie, zręczny intruz jest w stanie zatrzeć. Z tego wniosek, że prawdziwą trudność może sprawić doświadczonemu crackerowi zmanipulowanie śladów jedynie w przypadku, gdy logowanie zdarzeń odbywa się na innej maszynie niż ta, do której dostęp udało mu się uzyskać. Jednak tylko pod warunkiem, że nie powiedzie się mu także włamanie do maszyny logującej...

Nie tylko logi

Systemowe logi to niejedyne miejsce, w którym komputerowy włamywacz może pozostawić ślady działalności. Wielu administratorów, zwracających szczególna uwagę na bezpieczeństwo swych systemów informatycznych, nie polega wyłącznie na mechanizmach wbudowanych w systemy operacyjne i korzysta z dodatkowego oprogramowania zabezpieczającego. Przykłady takiego oprogramowania stanowią:

- *Tripwire* – system wykrywania włamań typu HIDS (ang. *host-based intrusion detection system*), którego zadaniem jest między innymi sprawdzanie integralności i poprawności zasobów plikowych,
- *COPS* – zbiór kilkunastu programów służących do monitorowania bezpieczeństwa w systemie *nix,
- *BinAudit* – program o funkcjonalności zbliżonej do Tripwire, główne zadanie: sprawdzanie integralności i poprawności zasobów plikowych.

Jako że programy tego typu są najczęściej wywoływane w systemie okresowo przez demona *cron*, cracker nie musi poszukiwać śladów ich obecności w systemie operacyjnym. Może przyjąć łatwiejszą do ogarnięcia taktykę przeanalizowania wpisów w tabelach *crontab* w poszukiwaniu wywołań wszelkich programów związanych ze sprawdzaniem bezpieczeństwa systemu. Zazwyczaj tabele *crontab* znajdują się w katalogu: */var/spool/cron/crontabs* lub */var/spool/cron*. Jeśli analiza okresowych wywołań wykaże obecność w systemie dodatkowych programów zabezpieczających, doświadczony cracker z pewnością będzie się starał zareagować na ten fakt. Do wyboru ma: wpłynięcie na program (zmiana konfiguracji, dezaktywacja itd.), tak by ten nie zareagował na jego obecność, lub zmianę swego zachowania (np. odinstalowanie trojana i próba zdobycia informacji w inny sposób), tak by ta nie spowodowała podniesienia alarmu przez program. Innym sposobem na poznanie zabezpieczeń atakowanego systemu oraz samych administratorów tegoż systemu, dostępnym dla włamywacza po udanej próbie włamania, jest analiza informacji, jakie udostępnia sam system na temat jego opiekunów. Pierwszym etapem w przypadku tej metody jest wytypowanie grupy administratorów systemu. Możliwe jest to na przykład na podstawie analizy pliku *passwd* oraz przynależności użytkowników do poszczególnych grup. Znając już wszystkich administratorów systemu, intruz może pokusić się o przejrzenie ich katalogów domowych i analizę:

- plików zawierających historię powłoki systemowej (w celu sprawdzenia, jakie komendy i programy są zazwyczaj przez nich wywoływane),
- plików profili powłok systemowych (w celu sprawdzenia ustawionych aliasów oraz ewentualnego logowania).

Poza analizą katalogu domowego, cracker może się pokusić o odnalezienie wszystkich innych katalogów należących do danego administratora (polecenie: *ls -alR ~/*) i analizę ich zawartości. Wszystko to może mu oczywiście pomóc w zidentyfikowaniu nietypowych zabezpieczeń i ich ominięciu.

Programów zapewniających dodatkowe bezpieczeństwo serwerów stworzono mnóstwo i z pewnością włamywacz nie jest w stanie ręcznie sprawdzić obecności każdego z nich w atakowanym systemie. Możliwe jest również, że administrator napisał swój własny program sprawdzający sumy kontrolne plików, gdyż nie jest to nic szczególnie trudnego. Jednak, jeśli w danym systemie używany jest jeden z bardzo popularnych systemów, taki jak *Tripwire* (typowe ścieżki: */usr/adm/tcheck*, */usr/local/adm/tcheck*), należy się spodziewać, że doświadczony cracker zauważy jego obecność na pierwszy rzut oka. W normalnych warunkach *Tripwire* powinien wszcząć alarm, gdy tylko suma kontrolna któregoś z „pilnowanych" plików ulegnie zmianie. Cracker może jednak w dość prosty sposób uśpić czujność tego programu po wprowadzeniu własnego, złośliwego kodu na przynajmniej dwa sposoby:

- zaktualizować bazę sum kontrolnych po wykonaniu podmiany lub modyfikacji kodu (w przypadku programu Tripwire polecenie: *tripwire -update /bin/podmieniony_plik*),
- zmodyfikować listę plików objętych procedurą sprawdzania sum kontrolnych (wykluczyć z listy zmodyfikowany lub dodany przez siebie plik).

Te – proste przecież – czynności spowodują, że pomimo wprowadzenia do systemu złośliwego kodu, niczego nieświadomy administrator będzie spał spokojnie, gdyż jego system HIDS nie zauważy niczego podejrzanego... Również przeprowadzana po ewentualnym wykryciu ataku analiza powłamaniowa może zostać w ten sposób zmylona i przez to nie wykryć prawdziwych intencji intruza.

Unikanie systemów HIDS

Wspomniane już wcześniej systemy wykrywania włamań typu HIDS opierają się na module agenta rezydującego w monitorowanym hoście. Zadaniem modułu jest analiza logów zdarzeń, plików systemowych i innych zasobów systemowych w poszukiwaniu zmian mogących świadczyć o wystąpieniu nieautoryzowanego

dostępu do systemu. W wypadku wykrycia nietypowego zdarzenia, HIDS automatycznie generuje odpowiedni alarm. Monitorowane mogą być między innymi próby logowania do systemu, stan (sumy kontrolne) plików systemowych i aplikacji. Zazwyczaj systemy HIDS opierają się na okresowym wykonywaniu „fotografii" stanu (obliczaniu sum kontrolnych) istotnych plików i porównywaniu ich z utworzoną wcześniej bazą wzorcową. Jeśli włamywaczowi uda się uzyskać dostęp do systemu i wprowadzić do niego własny kod – system wykrywania włamań podniesie alarm. Jak już wcześniej wspomniałem na przykładzie oprogramowania Tripwire, tego typu system może zostać stosunkowo łatwo oszukany przez włamywacza, poprzez odpowiednie (uwzględniające sumy kontrolne złośliwego kodu) zaktualizowanie bazy wzorcowej. Jak widać, systemy HIDS opierające się na okresowej, statycznej analizie systemu operacyjnego nie są zbyt wyrafinowane i mogą zostać w prosty sposób oszukane. Włamywacz może jednak na swej drodze napotkać również bardziej skomplikowane systemy HIDS, bazujące na wykrywaniu anomalii. Tego typu oprogramowanie analizuje dynamiczny stan systemu poprzez ciągłe monitorowanie i przechwytywanie odwołań do jądra systemu operacyjnego lub API i ich rejestrowanie w dzienniku zdarzeń. Systemy wykrywania anomalii rozpoczynają swą pracę od fazy uczenia się. Na tym etapie system tworzy profil typowych zachowań monitorowanego systemu operacyjnego i działających w nim aplikacji poprzez nauczenie się występujących w nim, typowych odwołań systemowych. Faza właściwego działania systemu polega natomiast na wykrywaniu wystąpień nietypowych zachowań (odwołań do jądra lub API) monitorowanego systemu, które najprawdopodobniej świadczą o wystąpieniu nieautoryzowanego dostępu do niego. Włamywacz spotykający na swej drodze system wykrywania anomalii, chcąc zatrzeć ślady incydentu, nie będzie miał z pewnością łatwego zadania. W takim wypadku łatwiejsze może się okazać niedopuszczenie do wszczęcia alarmu przez system HIDS. Ponieważ systemy tego rodzaju tworzą pewien model „normalnego" działania monitorowanego systemu, muszą zawsze zakładać pewien margines błędu. Gdyby prawidłowe zachowanie systemu zostało bardzo ściśle zdefiniowane, taki system generowałby ogromne ilości fałszywych wykryć (ang. *false positive*). Poza tym sekwencje wywołań funkcji systemowych będące wynikiem ataku są zazwyczaj diametralnie różne od wywołań będących wynikiem prawidłowego działania systemu, stąd założenie pewnej swobody w definiowaniu prawidłowego działania systemu nie zmniejsza drastycznie skuteczności wykrywania. Właśnie ten margines dozwolonych operacji, może zostać wykorzystany przez włamywacza do ukrycia swej działalności. Sam pomysł jest dość prosty i sprowadza się do takiego przeprowadzenia ataku, by imitował (ang. *mimicry attacks*) on prawidłowe działanie systemu i tym samym nie spowodował wszczęcia alarmu przez system HIDS. Ataki imitujące prawidłowe działanie systemu mogą być skuteczne przeciwko praktycznie wszystkim rodzajom systemów HIDS wykrywającym nietypowe sekwencje zdarzeń. Są to między innymi systemy oparte o śledzenie wywołań funkcji systemowych oraz wywołań komend systemowych. W celu skutecznego przeprowadzenia tego rodzaju ataku, intruz musi dysponować następującymi informacjami:

- wszechstronną znajomością samego systemu HIDS oraz algorytmu jego działania,
- wiedzą na temat dokładnej konfiguracji atakowanego systemu – jest to niezbędne w celu przewidzenia postaci bazy prawidłowych zachowań systemu (intruz może stworzyć podobny do atakowanego system testowy, uruchomić w nim system HIDS i poddać analizie wygenerowaną bazę zachowań dopuszczalnych, takie podejście pozwoli na wygenerowanie bazy bardzo zbliżonej do bazy w systemie docelowym).

Zazwyczaj w tego typu atakach zakłada się również, że intruz jest w stanie przejąć kontrolę nad atakowaną aplikacją w sposób niezauważalny dla systemu HIDS. Dla wielu typowych ataków warunek ten można dość łatwo spełnić. Systemy HIDS nie wykrywają zazwyczaj fazy penetracyjnej ataku (faza ta najczęściej nie zmienia znacząco sekwencji wywołań systemowych), a jedynie skutki przejęcia kontroli nad systemem. Przykładowo, ataki przepełniające bufory w aplikacjach nie zostaną wykryte przez typowy system HIDS, gdyż nie zmieniają wywołań systemowych, a jedynie ich parametry. Zakładając, że intruz dysponuje pewną niebezpieczną dla atakowanego systemu sekwencją wywołań systemowych, które chciałby uruchomić w ten sposób, by system HIDS nie podniósł alarmu, możliwe jest to na co najmniej kilka sposobów:

- wykonanie ataku w sposób niezmieniający zachowania aplikacji, nad którą kontrolę przejął włamywacz. Jak wcześniej wspomniałem, typowy system HIDS jest w stanie wykryć tylko znaczne odstępstwa od typowych wywołań systemowych związanych z danym programem,
- wykonanie ataku po uprzednim sprawdzeniu, że system HIDS nie jest w stanie go wykryć. Ponieważ metody imitacyjne zakładają, że intruz zna dość dokładnie bazę prawidłowych zachowań systemu, możliwe jest uruchomienie przez niego analogicznego testowego systemu HIDS i przetestowanie, czy dany atak zostanie przez konkretny system wykryty. W przypadku, gdy testowy system HIDS nie wykrywa ataku testowego, istnieje duża szansa, że atak wykonany

W Sieci

- *http://www.cert.org* – Computer Emergency Response Team,
- *http://packetstormsecurity.org/UNIX/penetration/log-wipers* – kody źródłowe wielu modyfikatorów logów systemowych, między innymi: ZAP, ZAP2, CLOAK i CLOAK2,
- *http://freeworld.thc.org/papers/COVER-1.TXT* – THC's HOW TO COVER YOUR TRACKS,
- *http://www.windowsecurity.com/whitepapers/How_to_cover_your_tracks__Practice_.html* – How to cover your tracks – Practice,
- *http://www.nfsec.pl/publikacje/bezpieczenstwo_w_teorii/ids_ips.html* – Rodzaje systemów IDS/IPS,
- *http://www.xcf.berkeley.edu/~paolo/ids-res/mimicry.pdf* – Mimicry Attacks on Host-Based Intrusion Detection Systems,
- *http://www.sysadmindayph.com/blog/cover-your-tracks-after-hacking-a-unix-box* – Cover Your Tracks After Hacking A UNIX Box.

w systemie docelowym również nie zostanie zauważony,

- podmiana parametrów wywołań funkcji systemowych. Co ciekawe, systemy HIDS sprawdzają zazwyczaj sekwencje wywołań poszczególnych funkcji systemowych, ale ignorują zupełnie parametry wywołania tych funkcji. Oznacza to, że przykładowe dwa wywołania: *open("/lib/libc.so", O_RDONLY)* oraz *open("/etc/shadow", O_RDWR)* są dla systemu HIDS nierozróżnialne. Pierwsze wywołanie otwiera tylko do odczytu plik */lib/libc.so*, natomiast drugie otwiera do odczytu i zapisu plik */etc/shadow*. Stąd prosty wniosek, że włamywacz może zupełnie niezauważenie podmienić wywołanie otwarcia dowolnego pliku na otwarcie dowolnego innego pliku w trybie odczytu i zapisu. W jaki sposób zostanie to przez włamywacza wykorzystane, zależy już wyłącznie od jego fantazji,
- przeplatanie sekwencji złośliwych wywołań wywołaniami typu *no-op*. Udowodniono, że odpowiednie urozmaicenie dowolnej sekwencji wywołań funkcji systemowych wywołaniami typu no-operation praktycznie zawsze pozwala na uniknięcie wykrycia. W tym wypadku również wystarczy, że intruz przetestuje specjalnie spreparowane sekwencje w środowisku testowym, by mieć pewność, że właściwy atak nie zostanie wykryty. Przykładami wywołań typu *no-op* są: otwarcie nieistniejącego pliku, otwarcie i natychmiastowe zamknięcie pliku, odczytanie zera bajtów z pliku.

Jak widać, ataki typu *mimicry* stanowią dla włamywaczy potężny oręż w walce z systemami klasy HIDS. Wynika to oczywiście z niedoskonałości samych systemów wykrywania włamań, więc należy zawsze pamiętać, że rozwiązania tego typu nie oferują stuprocentowej wykrywalności włamań. Przeprowadzając analizę powłamaniową, należy zatem zawsze pamiętać, że brak zdarzeń w systemie HIDS nie oznacza, że nic się nie wydarzyło. Być może intruzowi udało się przechytrzyć nasze zabezpieczenia, a dowodów jego działalności trzeba szukać gdzie indziej.

Podsumowanie

Analiza powłamaniowa zawsze była uważana za proces bardzo trudny, wymagający dużej wiedzy, doświadczenia i cierpliwości. W publikacjach na ten temat często zapomina się jednak o jeszcze jednej ważnej rzeczy, mianowicie o tym, że nie wystarczy sama wiedza na temat tego, gdzie w systemie operacyjnym można odnaleźć ślady działalności crackera. Równie istotna, a może nawet ważniejsza jest znajomość technik zacierania i preparowania śladów stosowanych przez komputerowych włamywaczy. Na podstawie przedstawionych przykładów widać wyraźnie, że cracker, będący wysokiej klasy specjalistą w zakresie systemów operacyjnych oraz stosowanych w nich zabezpieczeniach, jest w stanie po udanym włamaniu praktycznie dowolnie spreparować lub usunąć ślady swej obecności. Osoby zajmujące się zbieraniem cyfrowych dowodów muszą więc być szczególnie czujne i ostrożne w ocenie zbieranych śladów. Dane cyfrowe mogą zostać niezwykle łatwo spreparowane, co może skutkować oskarżeniami pod adresem niewinnych osób. Myślę więc, że wykładając metody analizy powłamaniowej, należy większą uwagę poświęcić metodom weryfikacji prawdziwości dowodów odkrywanych w skompromitowanym systemie operacyjnym. Jest to już jednak temat na odrębny artykuł, a może nawet szerszą publikację...

Wojciech Smol
Autor jest absolwentem wydziału Automatyki, Elektroniki i Informatyki Politechniki Śląskiej w Gliwicach. Ukończył studia na kierunku Informatyka, o specjalności Bazy danych, sieci i systemy komputerowe. Pracuje jako administrator sieci i systemów komputerowych w firmie Mostostal Zabrze Holding S.A.
Kontakt z autorem: *wojciech.smol@mz.pl*