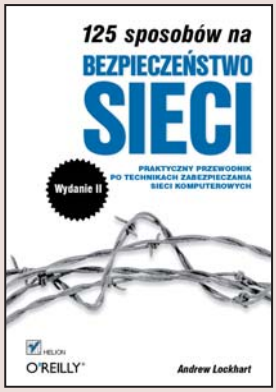
125 sposobów na bezpieczeństwo sieci. Wydanie II



Autor: Andrew Lockhart
Tłumaczenie: Leszek Sagalara na podstawie „100 sposobów na bezpieczeństwo sieci" w tłumaczeniu Witolda Zioło
ISBN: 978-83-246-0986-4
Tytuł oryginału: Network Security Hacks: Tips & Tools for Protecting Your Privacy
Format: B5, stron: 440

Praktyczny przewodnik po technikach zabezpieczania sieci komputerowych

- Jak sprawnie zabezpieczyć system?
- Jak zapewnić sobie prywatność w internecie?
- Jak skutecznie walczyć z sieciowymi włamywaczami?

W naszych domach montujemy solidne zamki i drzwi, chronimy samochody wymyślnymi alarmami, w firmach zakładamy systemy monitoringu, jednak nadal wiele osób nie zwraca wystarczającej uwagi na bezpieczeństwo komputerów w sieciach domowych oraz korporacyjnych. Luki w systemach informatycznych powodują każdego roku straty rzędu miliardów dolarów, a przecież dostępnych jest wiele narzędzi i technik, które pozwalają skutecznie zabezpieczyć komputery przed atakami crackerów.

„125 sposobów na bezpieczeństwo w sieci. Wydanie II" to zaktualizowany i rozbudowany zbiór praktycznych porad dotyczących zabezpieczania systemów informatycznych przed atakami. Dzięki wskazówkom przygotowanym przez cenionych profesjonalistów dowiesz się, jak stosować najnowsze narzędzia systemowe i niezależne programy do zabezpieczania systemów i danych, ochrony własnej prywatności w sieci czy bezpiecznego łączenia zdalnych maszyn. Nauczysz się także zastawiać pułapki na sieciowych napastników, wykrywać ich obecność w sieci i szybko przywracać normalne funkcjonowanie systemu po ewentualnym ataku.

- Zabezpieczanie systemów Windows i uniksowych
- Zapewnianie prywatności w internecie
- Konfigurowanie i testowanie zapór sieciowych
- Bezpieczne korzystanie z usług
- Zabezpieczanie sieci przewodowych i bezprzewodowych
- Monitorowanie działania sieci
- Stosowanie silnego szyfrowania i uwierzytelniania
- Wykrywanie włamań i przywracanie działania sieci

Stosuj sprawdzone sposoby zapewniania bezpieczeństwa w sieci

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Bezpieczeństwo systemu Unix

Sposoby 1. – 22.

Budowa sieci polega na łączeniu komputerów, co oznacza, że sieć komputerowa nie jest bardziej bezpieczna niż komputery, z których się składa. Jeden niezabezpieczony komputer stanowi zagrożenie dla całej sieci, gdyż gdy tylko dostanie się w niepowołane ręce, może zostać wykorzystany jako narzędzie rekonesansu lub platforma prowadzenia ataku. Gdy w którymś z serwerów działają łatwe do zawładnięcia usługi, zapory sieciowe (ang. *firewall*), systemy wykrywania włamań i inne środki bezpieczeństwa stają się bezradne. Przed zabezpieczeniem samej sieci, należy najpierw w dostatecznym stopniu zabezpieczyć komputery tej sieci.

W rozdziale przedstawiono wiele sposobów zmniejszenia ryzyka związanego z działaniem usług systemu uniksowego. Mimo że każde zaprezentowanie tu rozwiązanie stanowi zamkniętą całość, warto zapoznać się z resztą rozdziału. Stosując środki obronne tylko jednego rodzaju, ryzykuje się, że wszelkie starania pójdą na marne, gdy tylko włamywacz odkryje jak je obejść. Skarbiec na Wawelu nie jest chroniony jedynie drzwiami z kłódką, podobnie żadne pojedyncze rozwiązanie nie może skutecznie zabezpieczyć żadnego serwera, a liczba potrzebnych środków bezpieczeństwa rośnie wraz z wartością chronionych zasobów.

Zapewnienie bezpieczeństwa jest stałym procesem, który musi trwać i musi być doglądany. Nie ma, poza odłączeniem komputera od sieci, żadnego innego pojedynczego sposobu jego zabezpieczenia. Pamiętając o tym, można przystąpić do budowy bezpiecznego serwera zaspokajającego określone potrzeby.

SPOSÓB 1.

Zabezpieczenie punktów montowania

Za pomocą opcji montowania można ograniczyć działania włamywacza.

Z Uniksa korzysta się przeważnie przez jego system plików. Dlatego warto ograniczyć to, co z plikami może zrobić włamywacz po przedostaniu się do systemu. Jednym ze sposobów na to jest użycie ograniczających opcji montowania.

Opcja montowania decyduje o sposobie dostępu do systemu plików. Przekazywana jest do jądra systemu operacyjnego w czasie uruchamiania systemu plików. Opcje montowania można wykorzystać, by uniemożliwić traktowanie plików jako urządzeń, wykonywanie programów binarnych oraz wykorzystanie bitu SUID. W tym celu używa się parametrów odpowiednio `nodev`, `noexec` i `nosuid`. System plików można również za pomocą opcji `ro` zamontować w trybie „tylko do odczytu".

Parametry te podaje się wierszu poleceń, wydając polecenie `mount` z opcją `-o`. Przeznaczywszy na przykład dla katalogu */tmp* oddzielną partycję, będącą trzecią partycją pierwszego dysku IDE systemu, można zamontować ją za pomocą parametrów `nodev`, `noexec` i `nosuid` za pomocą poniższego polecenia:

```
# mount -o nodev,noexec,nosuid /dev/hda3 /tmp
```

Poleceniu temu odpowiada zapis w pliku */etc/fstab*:

```
/dev/hda3    /tmp    ext3    defaults,nodev,noexec,nosuid    1 2
```

Po starannym przeanalizowaniu swoich potrzeb i podzieleniu dysku na kilka systemów plików można za pomocą opcji montowania zwiększyć trudności, jakie napotka włamywacz, próbując głębiej wniknąć do systemu. Najlepiej to zrobić, dzieląc drzewo katalogów na obszary, w których do poprawnego działania systemu musi istnieć możliwość zapisywania oraz na takie, w których taka możliwość nie jest konieczna. Opcję „tylko do odczytu" można zastosować we wszystkich częściach systemu plików, których zawartość nie zmienia się często. Dobrym kandydatem do użycia tej opcji może być na przykład katalog */usr*, ale zależy to od częstotliwości uaktualniania oprogramowania systemowego.

Oczywiście wiele katalogów (na przykład */home*) musi być montowanych w trybie „do odczytu i do zapisu". Mimo to jest bardzo mało prawdopodobne, by użytkownicy tradycyjnego systemu wielodostępnego musieli uruchamiać w swoich katalogach macierzystych programy binarne z bitem SUID, lub by musieli w nich tworzyć pliki urządzeń. Dlatego katalogi macierzyste użytkowników można przechowywać w oddzielnym systemie plików zamontowanym z opcjami `nodev` i `nosuid`. Dodatkowo, jeżeli dojdzie się do wniosku, że użytkownicy nie muszą uruchamiać programów przechowywanych w swych katalogach macierzystych, można też użyć opcji montowania `noexec`. Podobna sytuacja ma miejsce w przypadku katalogów */tmp* i */var*, w których prawdopodobieństwo, by jakiś proces miał uzasadnione powody wykonywania programów binarnych z bitem SUID, lub bez niego, albo korzystał z plików urządzeń jest bardzo małe. Tego rodzaju zabezpieczenie zapobiega pozostawieniu przez włamywacza trojana na przykład w katalogu */tmp* lub w katalogu któregoś użytkownika. W takim przypadku włamywaczowi uda się zainstalować program, ale nie będzie mógł go już uruchomić, bez względu na bity `chmod` programu.

Należy zauważyć, że usługi działające w środowisku `chroot()` **[Sposób 10.]** mogą nie działać, gdy opcja `nodev` zostanie użyta do systemu plików działającym w tym środowisku. Wynika to z tego, że urządzenia takie jak */dev/log* czy */dev/null* muszą być dostępne w środowisku `chroot()`.

Istnieje kilka sposobów, za pomocą których włamywacz może obejść ograniczenia punktów montowania. W Linuksie działanie opcji `noexec` można ominąć, uruchamiając programy binarne znajdujące się w takim systemie plików za pomocą biblioteki */lib/ld-linux.so*. Na pierwszy rzut oka wydaje się, że można temu zaradzić uniemożliwiając uruchomienie biblioteki *ld-linux.so*, ale to uniemożliwiłoby również wykonywanie się wszystkich programów korzystających z bibliotek dołączanych dynamicznie.

Zatem jeżeli wszystkie używane programy nie są połączone z bibliotekami statycznie (a najprawdopodobniej nie są), użycie opcji `noexec` w Linuksie ma niewielkie znaczenie. Poza tym, włamywacz, który zdobył uprawnienia użytkownika root nie będzie zbyt dotkliwie ograniczony zamontowaniem systemu plików ze specjalnymi opcjami, gdyż może on rozmontować go i zamontować ponownie za pomocą opcji `-o remount`. Jednak stosując opcje montowania, można z łatwością ograniczyć możliwość działania włamywacza, zanim ten posiądzie uprawnienia użytkownika root.

SPOSÓB 2.

Wynajdywanie programów z ustawionymi bitami SUID i SGID

Szybki sposób wykrycia programów, które mogą umożliwić uzyskanie uprawnień użytkownika root oraz programów otwierających tylne wejścia.

Jedna z możliwości zwiększenia przez użytkownika swoich uprawnień w systemie polega na wykorzystaniu słabości programu opatrzonego bitem SUID lub SGID. Bity SUID i SGID służą do uzyskania przez program specjalnych uprawnień, wyższych od posiadanych przez uruchamiającego go użytkownika. Jednym z takich programów jest *passwd*. Pozwolenie użytkownikowi na zmianę hasła, przy jednoczesnym uniemożliwieniu mu modyfikowania systemowego pliku haseł oznacza, że program *passwd* musi działać z uprawnieniami użytkownika root. Dlatego program ten ma ustawiony bit SUID, który powoduje, że program wykonywany jest z uprawnieniami jego właściciela. Podobnie, gdy ustawiony jest bit SGID, program wykonuje się z uprawnieniami grupy będącej właścicielem pliku.

Wynik działania polecenia `ls -l` na programie binarnym mającym ustawiony bit SUID wygląda następująco:

```
-r-s--x--x    1 root     root        16336 Feb 13  2003 /usr/bin/passwd
```

Zamiast bitu wykonania (`x`), w grupie bitów właściciela pliku programu występuje bit `s`, który oznacza, że program posiada ustawiony bit SUID.

Niestety, źle napisany program binarny posiadający ustawiony bit SUID lub SGID może posłużyć do szybkiego i stosunkowo łatwego podniesienia uprawnień użytkownika. Poza tym włamywacz, który już zdobył uprawnienia użytkownika root, może ukryć w systemie programy binarne SUID, by otworzyć sobie tylne wejście do systemu. Z tych powodów należy odnaleźć w systemie wszystkie programy binarne SUID i SGID. Jest to prosta czynność, którą można wykonać za pomocą polecenia:

```
# find / \( -perm -4000 -o -perm -2000 \) -type f -exec ls -la {} \;
```

Ważną rzeczą, którą należy stwierdzić, jest to czy program SUID jest skryptem powłoki, czy plikiem binarnym, gdyż niezwykle łatwo jest zmienić potencjalnie nieszkodliwy skrypt w tylne wejście do systemu. Na szczęście większość systemów operacyjnych ignoruje bity

SUID i SGID skryptów powłoki. Chcąc odnaleźć wszystkie skrypty z ustawionymi bitami SUID lub SGID, należy zmienić argument opcji `-exec` ostatniego polecenia i dodać do niego potok:

```
# find / \( -perm -4000 -o -perm -2000 \) \
  -type f -exec file {} \; | grep -v ELF
```

W tym przypadku, po odkryciu pliku z ustawionym bitem SUID lub SGID uruchomiony zostanie program *file*, który sprawdzi, jakiego rodzaju jest badany plik. Jeżeli jest to plik wykonywalny, program *grep* odfiltruje go. W przeciwnym razie informacje o pliku pojawią się na ekranie.

W większości systemów operacyjnych pliki wykonywalne zapisane są w formacie `ELF`. W przypadku systemów operacyjnych nie korzystających z tego formatu (starsze wersje Linuksa używały formatu `a.out`, a AIX używa `XCOFF`), należy w poleceniu `grep` zastąpić typ `ELF` typem formatu binarnego używanego w danym systemie operacyjnym. W przypadku wątpliwości, jaki format jest używany w danym systemie, program *file* należy wykonać na dowolnym programie binarnym, w wyniku czego na ekranie pojawi się poszukiwany łańcuch znaków.

Oto przykład użycia programu *file* na pliku binarnym systemu Mac OS X:

```
$ file /bin/sh
/bin/sh: Mach-O executable ppc
```

Można jeszcze pójść dalej i za pomocą programu *cron* uruchamiać polecenie wyszukujące programy SUID codziennie, a wynik jego działania zapisywać do pliku. Na przykład poniższy zapis znajdujący się w pliku *crontab* służy do wyszukiwania plików z ustawionym bitem SUID lub SGID i do porównania aktualnych wyników poszukiwań z wynikami z dnia poprzedniego, a następnie wysyłania wyników poprzez email (wszystkie polecenia należy wpisać w jednym wierszu):

```
0 4 * * * find / \( -perm -4000 -o -perm -2000 \) -type f \
  > /var/log/sidlog.new &&
  diff /var/log/sidlog.new /var/log/sidlog &&
  mv /var/log/sidlog.new /var/log/sidlog
```

Aktualna lista plików SUID i SGID zostanie zapisana do pliku */var/log/sidlog*.

SPOSÓB 3.

Wynajdywanie udostępnionych katalogów

Szybki sposób wykrycia niezabezpieczonych katalogów.

Katalogi, w których mogą zapisywać wszyscy (ang. *world-writable*) lub członkowie grup, stanowią z punktu widzenia bezpieczeństwa poważny problem. Jeżeli maski `umask` są niepoprawne, użytkownicy mogą nieświadomie tworzyć niebezpieczne pliki, nie zdając sobie przy tym sprawy z wynikających z tego konsekwencji. Mając to na uwadze, należy odszukać w systemie wszystkie katalogi o rozluźnionych uprawnieniach. Podobnie jak w poprzednim przypadku **[Sposób 2.]**, można to zrobić za pomocą programu *find*:

```
# find / -type d \( -perm -g+w -o -perm -o+w \) -exec ls -lad {} \;
```

Wszystkie katalogi, których nazwy pojawią się w wyniku działania tego polecenia, powinny mieć ustawiony w uprawnieniach „lepki" bit reprezentowany przez literę `t`. Ustawienie „lepkiego" bitu w uprawnieniach do katalogu, w którym zapisywać mogą wszyscy powoduje, że mimo że wszyscy mogą tworzyć w nim pliki, to jednak nikt nie może usuwać ani modyfikować plików innych użytkowników.

Dostrzegając w wyniku działania powyższego polecenia katalog nie zawierający „lepkiego" bitu należy się zastanowić, czy rzeczywiście uprawnienia do zapisu w tym katalogu muszą mieć wszyscy, czy może lepiej jest zastosować w nim uprawnienia grupowe lub listy ACL **[Sposób 4.]**. Jeżeli uprawnienia do zapisywania w katalogu muszą mieć rzeczywiście wszyscy, należy za pomocą polecenia `chmod +t` ustawić w nim „lepki" bit.

Aby uzyskać listę katalogów, które nie mają ustawionego „lepkiego" bitu, należy wydać polecenie:

```
# find / -type d \( -perm -g+w -o -perm -o+w \) \
  -not -perm -a+t -exec ls -lad {} \;
```

W przypadku systemów, które dla każdego użytkownika tworzą oddzielną grupę (na przykład przy tworzeniu użytkownika andrew tworzona jest równocześnie jego podstawowa grupa o tej samej nazwie), należy zmodyfikować poprzednie polecenie, by nie poszukiwało katalogów zapisywalnych przez grupy. W przeciwnym razie na ekranie pojawi się wiele nieistotnych informacji. W tym celu z polecenia należy usunąć człon `-perm -g+w`.

SPOSÓB 4.

Tworzenie elastycznych hierarchii uprawnień za pomocą list ACL standardu POSIX

Tam, gdzie tradycyjne uniksowe uprawnienia nie wystarczą, należy wykorzystać listy ACL.

W większości przypadków tradycyjny system uprawnień systemu Unix sprawdza się znakomicie. Jednak w środowiskach wielu współpracujących ze sobą użytkowników wymagających dostępu do różnych plików, system ten staje się nieporęczny. Listy kontroli dostępu *ACL* (ang. *Access Control List*) są czymś stosunkowo nowym w systemach uniksowych, ale od jakiegoś czasu są wykorzystywane w ich komercyjnych odpowiednikach. Listy ACL co prawda nie zwiększają bezpieczeństwa systemu, za to zmniejszają złożoność zarządzania uprawnieniami. List ACL są nowym sposobem nadawania uprawnień do plików i katalogów, nie wymagającym tworzenia niepotrzebnych grup.

Listy ACL przechowywane są w postaci rozszerzonych atrybutów w metadanych systemu plików. Jak sama nazwa wskazuje, służą do przyznawania lub odmawiania dostępu do danego pliku na podstawie zdefiniowanych kryteriów. Listy ACL nie polegają jednak na całkowitej rezygnacji z tradycyjnego systemu uprawnień. Listy ACL można definiować zarówno dla użytkowników, jak i grup, i wciąż związane są z dostępem do odczytu, do zapisu i do wykonania. Dodatkowo listę kontroli dostępu można zdefiniować dla dowolnego użytkownika lub grupy, którym nie została przypisana żadna inna lista ACL, co ma podobne znaczenie jak bity „pozostali" w uprawnieniach do pliku.

System list kontroli dostępu używa też tak zwanych *masek ACL*, filtrujących uprawnienia wszystkich list ACL dotyczących określonych użytkowników lub grupy. Działanie masek ACL jest podobne do działania polecenia `umask`, choć nie do końca. Gdy na przykład maska ACL przyjmuje wartość `r--`, dowolna lista ACL odnosząca się do określonego użytkownika lub grupy, która pozwala na więcej (na przykład `rw-`), zostaje w rezultacie ograniczona do `r--`. Katalogi mogą zawierać domyślne listy ACL, które definiują początkowe listy ACL tworzonych w nich plików i podkatalogów.

Aktywacja list kontroli dostępu

Większość stosowanych obecnie systemów plików w Linuksie (Ext2/3, ReiserFS, JFS i XFS) potrafi obsługiwać listy ACL. W przypadku korzystania z systemu Linux należy sprawdzić, czy ustawiona została jedna z poniższych opcji konfiguracyjnych jądra, odpowiednio do używanego systemu plików:

```
CONFIG_EXT2_FS_POSIX_ACL=y
CONFIG_EXT3_FS_POSIX_ACL=y
CONFIG_REISERFS_FS_POSIX_ACL=y
CONFIG_JFS_POSIX_ACL=y
CONFIG_FS_POSIX_ACL=y
CONFIG_XFS_POSIX_ACL=y
```

Aby aktywować listy ACL w systemie FreeBSD, należy zamontować dowolny system plików z opcją montowania `acls`:

```
# mount -o acls -u /usr
# mount
/dev/ad0s1a on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ad0s1e on /tmp (ufs, local, soft-updates)
/dev/ad0s1f on /usr (ufs, local, soft-updates, acls)
/dev/ad0s1d on /var (ufs, local, soft-updates)
```

Opcja `-u` powoduje aktualizację montowania, co umożliwia zmianę opcji montowania dla aktualnie zamontowanego systemu plików. W celu cofnięcia operacji można wyłączyć listy ACL, używając opcji `noacls`. Aby automatycznie włączyć listy ACL wraz ze startem systemu, należy w następujący sposób zmodyfikować wpis w pliku */etc/fstab*:

```
/dev/ad0s1f             /usr            ufs     rw,acls         2       2
```

Zarządzanie listami ACL

Listy ACL modyfikuje się i usuwa za pomocą programu *setfacl*. Aby zmodyfikować listę ACL, należy posłużyć się opcją `-m`, po której należy podać listę ACL oraz nazwę lub nazwy plików. Aby usunąć listę ACL, należy posłużyć się opcją `-x`, po której należy podać listę lub listy ACL.

Istnieją trzy ogólne postacie list ACL — listy użytkownika, grupy i pozostałych. Przyjrzyjmy się im bliżej:

```
# Lista ACL użytkownika
u:[użytkownik]:<tryb_dostępu>
# Lista ACL grupy
```

```
g:[grupa]:<tryb_dostępu>
# Lista ACL pozostałych
o:<tryb_dostępu>
```

Należy zauważyć, że w listach ACL użytkownika i grupy nazwy użytkowników i grup, których dotyczą listy ACL, nie są wymagane. Jeżeli nazwy te zostaną pominięte, oznacza to, że lista ACL zostanie zastosowana do bazowej listy ACL uzyskanej z bitów trybu dostępu do pliku. Dlatego, jeżeli zostanie ona zmieniona, zmianie ulegną również bity trybu dostępu, i odwrotnie.

Działanie list ACL najlepiej poznać własnoręcznie na przykładzie, tworząc plik i modyfikując jego bazową listę ACL:

```
$ touch mojplik
$ ls -l mojplik
-rw-rw-r--    1 andrew   andrew          0 Oct 13 15:57 mojplik
$ setfacl -m u::---,g::---,o:--- mojplik
$ ls -l mojplik
----------    1 andrew   andrew          0 Oct 13 15:57 mojplik
```

Z przykładu wynika również, że w jednym poleceniu można podać naraz kilka list ACL, oddzielając je od siebie przecinkami.

Listy ACL można definiować dla dowolnej liczby grup i użytkowników:

```
$ touch costam
$ setfacl -m u:jlope:rwx,g:wine:rwx ,o:--- costam
$ getfacl costam
# file: costam
# owner: andrew
# group: andrew
user::rw-
user:jlope:rwx
group::---
group:wine:rwx
mask::rwx
other::---
```

Jeżeli zmieni się maskę ACL na `r--`, lista ACL użytkownika jlope i grupy wine zostanie w efekcie ograniczona do `r--`:

```
$ setfacl -m m:r-- costam
$ getfacl costam
# file: costam
# owner: andrew
# group: andrew
user::rw-
user:jlope:rwx                  #effective:r--
group::---
group:wine:rwx                  #effective:r--
mask::r--
other::---
```

Jak powiedziano wcześniej, do plików tworzonych w katalogach stosowane są domyślne listy ACL katalogów. Domyślne listy ACL tworzy się, poprzedzając listę ACL znakami `d:`:

```
$ mkdir mojkatalog
$ setfacl -m d:u:jlope:rwx mojkatalog
$ getfacl mojkatalog
# file: mojkatalog
```

```
# owner: andrew
# group: andrew
user::rwx
group::---
other::---
default:user::rwx
default:user:jlope:rwx
default:group::---
default:mask::rwx
default:other::---

$ touch mojkatalog/nowyplik
$ getfacl mojkatalog/nowyplik
# file: mojkatalog/nowyplik
# owner: andrew
# group: andrew
user::rw-
user:jlope:rwx                  #effective:rw-
group::---
mask::rw-
other::---
```

Jak łatwo zauważyć w powyższych przykładach, zawartość list ACL można uzyskać za pomocą programu *getfacl*. Jest to prosty program posiadający jedynie kilka opcji, z których najczęściej używaną jest opcja `-R`, umożliwiająca rekursywne przeglądanie list ACL, działająca bardzo podobnie do polecenia `ls -R`.

SPOSÓB 5.

Zabezpieczenie dzienników zdarzeń przed modyfikacją

Za pomocą atrybutów plików można uniemożliwić intruzom zatarcie śladów włamania.

Jest praktycznie pewne, że włamanie do systemu zostanie odnotowane w wielu różnych systemowych dziennikach zdarzeń. Jest to bardzo wartościowy ślad, który należy dobrze chronić. Bez wiarygodnych dzienników zdarzeń, odkrycie, w jaki sposób włamywacz przedostał się do systemu lub skąd został przeprowadzony atak, może być bardzo trudne. Tego rodzaju informacje są absolutnie niezbędne do przeprowadzenia właściwej analizy całego incydentu, a następnie do podjęcia stosownej reakcji na incydent, na przykład przez skontaktowanie się z zamieszanymi w to stronami **[Sposób 125.]**. Jeżeli włamanie powiedzie się i włamywacz uzyska uprawnienia użytkownika root, czy jest coś, co może powstrzymać go przed usunięciem śladów włamania?

W takim przypadku z pomocą przychodzą atrybuty plików. Zarówno systemy linuksowe, jak i systemy BSD przydzielają plikom i katalogom dodatkowe atrybuty, wykraczające poza standardowy uniksowy system zabezpieczeń, w którym atrybuty plików dotyczą w jednakowym stopniu wszystkich użytkowników systemu. Dodatkowe atrybuty decydują o dostępie do plików w znacznie większym stopniu niż uprawnienia do plików czy listy ACL **[Sposób 4.]**. W Linuksie atrybuty plików można przeglądać lub ustawiać za pomocą programów *lsattr* i *chattr*. Aby przejrzeć atrybuty w systemach BSD, należy wydać polecenie `ls -lo`, a żeby je ustawić — polecenie `chflags`.

Jednym z bardziej przydatnych atrybutów jest atrybut „tylko do dopisywania" (ang. *append-only*). Gdy atrybut ten jest ustawiony, pliku nie można usunąć, a zapis do pliku jest możliwy tylko na jego końcu.

Aby w systemie Linux ustawić atrybut „tylko do dopisywania", należy wydać następujące polecenie:

```
# chattr +a nazwa_pliku
```

W systemie BSD należy wydać polecenie:

```
# chflags sappnd nazwa_pliku
```

Żeby przekonać się, jak działa atrybut `+a`, należy utworzyć plik i ustawić w nim ten atrybut:

```
# touch /var/log/logfile
# echo "atrybut tylko do dopisywania nie jest ustawiony" > /var/log/logfile
# chattr +a /var/log/logfile
# echo "atrybut tylko do dopisywania jest ustawiony" > /var/log/logfile
bash: /var/log/logfile: Operation not permitted
```

Druga próba zapisu zakończyła się niepowodzeniem, gdyż nie można nadpisać pliku. Jednak dopisywanie na końcu pliku jest możliwe:

```
# echo "dopisywanie do pliku" >> /var/log/logfile
# cat /var/log/logfile
atrybut tylko do dopisywania nie jest ustawiony
dopisywanie do pliku
```

Oczywiście włamywacz, który zdobył uprawnienia użytkownika root, może odkryć, że w systemie wykorzystywane są atrybuty plików i za pomocą polecenia `chattr -a` usunąć zastosowany do plików dzienników zdarzeń atrybut „tylko do dopisywania". Aby temu zapobiec, należy wyłączyć możliwość usuwania atrybutu „tylko do dopisywania". W tym celu, w Linuksie należy skorzystać z jego mechanizmu możliwości, a w systemach BSD skorzystać z funkcji *poziomu bezpieczeństwa* (ang. *securelevel*).

Model możliwości systemu Linux dzieli wszystkie przywileje dane użytkownikowi root i umożliwia selektywne wyłączanie niektórych z nich. Aby uniemożliwić użytkownikowi usunięcie z pliku atrybutu „tylko do dopisywania", należy pozbyć się możliwości `CAP_LINUX_IMMUTABLE`. Zmian możliwości systemu można dokonać za pomocą prostego programu *lcap* (*http://snort-wireless.org/other/lcap-0.0.6.tar.bz2*).

Poniższe polecenie rozpakuje i skompiluje program:

```
# tar xvfj lcap-0.0.6.tar.bz2 && cd lcap-0.0.6 && make
```

Aby uniemożliwić modyfikację atrybutu „tylko do dopisywania", należy wykonać następujące polecenia:

```
# ./lcap CAP_LINUX_IMMUTABLE
# ./lcap CAP_SYS_RAWIO
```

Pierwsze polecenie wyłącza możliwość zmiany atrybutu „tylko do dopisywania", natomiast drugie wyłącza możliwość wykonywania pierwotnych operacji wejścia-wyjścia. Jest to konieczne po to, żeby nie było możliwości zmodyfikowania chronionego pliku poprzez dostęp do urządzenia blokowego, na którym się on znajduje. Zabezpiecza to również przed dostępem do urządzeń */dev/mem* i */dev/kmem*, który mógłby stwarzać włamywaczowi możliwość ponownego włączenia możliwości `CAP_LINUX_IMMUTABLE`.

Aby usunąć obie możliwości podczas startu systemu, powyższe polecenia należy umieścić w skryptach uruchomieniowych systemu (na przykład w */etc/rc.local*). Obie możliwości powinny zostać usunięte w dalszej części porządku uruchamiania systemu, aby nie spowodować problemów innych skryptów uruchomieniowych. Po usunięciu przez program *lcap* możliwości jądra, można je przywrócić jedynie przez ponowne uruchomienie systemu.

W systemach BSD to samo osiąga się, wykorzystując poziomy bezpieczeństwa. Poziom bezpieczeństwa to zmienna wykorzystywana przez jądro, której wartość decyduje o wyłączeniu pewnych możliwości systemu. Użycie poziomu bezpieczeństwa równego 1 jest równoznaczne z wyłączeniu omówionych wcześniej możliwości systemu Linux. Wartości poziomu bezpieczeństwa większej niż 0 nie można zmniejszyć. Domyślnie, w trybie wielodostępnym, OpenBSD podnosi wartość poziomu bezpieczeństwa do 1. W systemie FreeBSD domyślną wartością jest –1.

Aby zmienić poziom bezpieczeństwa, należy w pliku */etc/sysctl.conf* umieścić poniższy wiersz:

```
kern.securelevel=1
```

Zanim się to jednak zrobi, należy zdać sobie sprawę, że użycie atrybutu „tylko do dopisywania" uniemożliwi wykonywanie się skryptów rotujących pliki dzienników zdarzeń. Mimo to użycie atrybutu „tylko do dopisywania" w dużym stopniu podnosi bezpieczeństwo dzienników zdarzeń, których wartość w przypadku włamania do systemu jest nieoceniona.

SPOSÓB 6.

Podział zadań administracyjnych

Zadania administracyjne mogą wykonywać również inni użytkownicy, nie mający nawet uprawnień użytkownika root.

Za pomocą programu *sudo* można powierzyć część zadań administracyjnych innym użytkownikom, i to bez potrzeby przydzielania im dostępu do konta użytkownika root. Program *sudo* jest programem binarnym działającym z uprawnieniami użytkownika root, który wykonuje polecenia autoryzowanych użytkowników po podaniu przez nich ich hasła.

Aby skonfigurować program *sudo*, należy jako root uruchomić polecenie */usr/sbin/visudo*, otwierające do edycji listę użytkowników programu *sudo*. Domyślna lista ma następującą postać:

```
root ALL=(ALL) ALL
```

Niestety wielu administratorów wykorzystuje ten zapis jako szablon i przyznaje wszystkim innym administratorom niczym nieograniczony dostęp z uprawnieniami użytkownika root:

```
root ALL=(ALL) ALL
rob ALL=(ALL) ALL
jim ALL=(ALL) ALL
david ALL=(ALL) ALL
```

I choć metodę tę można wykorzystać do nadawania użytkownikom uprawnień użytkownika root bez zdradzania im jego hasła, to jednak należy z niej korzystać jedynie wówczas,

gdy użytkownikom programu *sudo* można całkowicie ufać. Program *sudo* ma jednak niezwykle elastyczne możliwości konfiguracyjne, dzięki którym można pozwolić na wykonywanie dowolnych poleceń, z uprawnieniami dowolnych użytkowników (pod dowolnym identyfikatorem UID).

Składnia wiersza konfiguracji programu *sudo* wygląda następująco:

```
użytkownik komputer=(użytkownik_efektywny) polecenie
```

Pierwsza kolumna określa użytkownika programu *sudo*. Druga kolumna określa komputery, których dotyczy ta pozycja. Umożliwia to używanie tego samego pliku konfiguracyjnego w wielu komputerach.

Załóżmy, że programista musi mieć dostęp z uprawnieniami użytkownika root do komputera testowego, ale nie do żadnego innego:

```
peter beta.oreillynet.com=(ALL) ALL
```

Następna kolumna (ujęta w nawiasy) określa efektywnego użytkownika, który uruchamia polecenia. Przydaje się to do uruchamiania programów z uprawnieniami użytkowników innych niż root:

```
peter lists.oreillynet.com=(mailman) ALL
```

Ostatnia kolumna zawiera wszystkie polecenia, które użytkownik może uruchamiać:

```
david ns.oreillynet.com=(bind) /usr/sbin/rndc,/usr/sbin/named
```

W przypadku długich list poleceń (użytkowników, komputerów) można skorzystać z aliasów programu *sudo*. W każdym wierszu konfiguracji programu *sudo*, w miejsce każdej pozycji można umieścić odpowiadający jej alias:

```
User_Alias ADMINS=rob,jim,david
User_Alias WEBMASTERS=peter,nancy
Runas_Alias DAEMONS=bind,www,smmsp,ircd
Host_Alias WEBSERVERS=www.oreillynet.com,www.oreilly.com,www.perl.com
Cmnd_Alias PROCS=/bin/kill,/bin/killall,/usr/bin/skill,/usr/bin/top
Cmnd_Alias APACHE=/usr/local/apache/bin/apachectl
WEBMASTERS WEBSERVERS=(www) APACHE
ADMINS ALL=(DAEMONS) ALL
```

Zamiast nazw użytkowników można podawać nazwy grup systemowych, w ten sposób dane polecenia będzie mógł wykonywać każdy użytkownik należący do podanej grupy. Nazwę grupy należy poprzedzić znakiem `%`:

```
%wwwadmin WEBSERVERS=(www) APACHE
```

W tym przypadku użytkownik należący do grupy wwwadmin może w dowolnym serwerze WWW uruchamiać jako użytkownik www program *apachectl*.

Bardzo interesująca jest opcja `NOPASSWD:`, której użycie powoduje, że do uruchomienia polecenia nie trzeba podawać hasła:

```
rob ALL=(ALL) NOPASSWD: PROCS
```