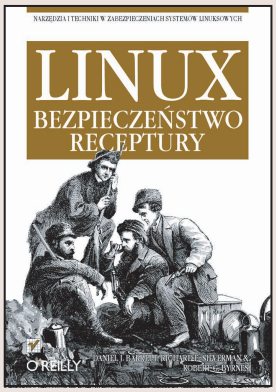
Linux. Bezpieczeństwo. Receptury



Autorzy: Daniel J. Barrett,
Richard E. Silverman, Robert G. Byrnes
Tłumaczenie: Adam Podstawczyński
ISBN: 83-7361-249-1
Tytuł oryginału: Linux Security Cookbook
Format: B5, stron: 346

Zabezpieczanie systemu komputerowego to proces złożony. Nie trzeba jednak od razu wprowadzać złożonych mechanizmów ochrony systemu. Znajomość podstawowych procedur pomaga dostatecznie zwiększyć poziom bezpieczeństwa. Czy chcesz szybko dowiedzieć się, jak wysyłać zaszyfrowane listy elektroniczne z programu Emacs? Jak ograniczyć dostęp do usług sieciowych w określonych porach dnia? Jak zabezpieczyć serwer WWW zaporą sieciową? Skonfigurować uwierzytelnianie z użyciem klucza publicznego przez SSH?

Książka „Linux. Bezpieczeństwo. Przewodnik encyklopedyczny" nauczy Cię, jakie polecenia należy wykonać i co wpisać w plikach konfiguracyjnych, by poprawić bezpieczeństwo Twojego systemu. Nie jest to klasyczny podręcznik; nie znajdziesz tu teorii, lecz rozwiązania konkretnych problemów i sposoby łatania typowych luk w zabezpieczeniach. Dzięki książce nie będziesz tracić cennego czasu, poszukując właściwej składni poleceń. Przeznaczona jest dla średnio zaawansowanych użytkowników i administratorów systemów Linux.

- Kontrola dostępu do systemu na różnych poziomach - od zapory sieciowej aż po poszczególne usługi; programy: iptables, ipchains, xinetd, inted i wiele innych.
- Monitorowanie sieci programami: ethereal, dsniff, netstat i innymi.
- Ochrona połączeń sieciowych technologiami SSH i SSL.
- Wykrywanie włamań programami: tripwire, snort, tcpdump, logwatch i innymi.
- Zabezpieczanie uwierzytelniania za pomocą kluczy kryptograficznych, technologii Kerberos, oprogramowania PAM; autoryzacja przywilejów administratora programem sudo.
- Szyfrowanie plików i wiadomości e-mail oprogramowaniem GnuPG.
- Sondowanie zabezpieczeń własnego systemu programami do łamania haseł, narzędziem nmap i skryptami pomocniczymi.

Jeśli administrujesz systemami linuksowymi, receptury przedstawione w niniejszej książce pozwolą zwiększyć wydajność Twojej pracy: osiągniesz więcej poświęcając mniej czasu. Zdobędziesz pewność, że zastosujesz właściwe rozwiązania gdy pojawiają się konkretne zagrożenia.

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

Spis treści

3

Kontrola dostępu do sieci

3.0. Wprowadzenie

Jednym z najważniejszych zadań związanych z zabezpieczaniem systemów jest kontrolowanie przychodzących połączeń sieciowych. Administrator systemów może zapewnić taką kontrolę na różnych poziomach. Na najniższym poziomie — sprzętowym — może po prostu odłączyć kable. Takie środki są jednak rzadko wymagane, chyba że włamanie było na tyle poważne, iż komputerowi już w żaden sposób nie można ufać. Kontrolowanie za pomocą oprogramowania jest dużo praktyczniejsze i też może być realizowane na różnych poziomach — od ogólnego do specyficznego dla usługi:

Interfejs sieciowy

Interfejs można włączyć lub wyłączyć.

Zapora sieciowa

Konfigurując reguły zapory sieciowej w jądrze linuksowym, kontrolujemy ruch przychodzący (a także wychodzący i przekazywany). Zagadnienia te omówiono w rozdziale 2.

Superdemon lub demon usług internetowych

Superdemon odpowiada za uruchamianie (bądź nie) specyficznych usług sieciowych. Decyzja jest podejmowana na podstawie różnych kryteriów. Załóżmy, że system otrzymuje przychodzące żądanie połączenia Telnet. Superdemon może takie żądanie przyjąć lub odrzucić na podstawie adresu źródłowego, pory dnia lub liczby otwartych połączeń Telnet. Może również całkowicie zablokować dostęp przez Telnet. Superdemony zazwyczaj działają na plikach konfiguracyjnych, za pomocą których można w wygodny sposób sterować dostępem do wszystkich usług systemowych.

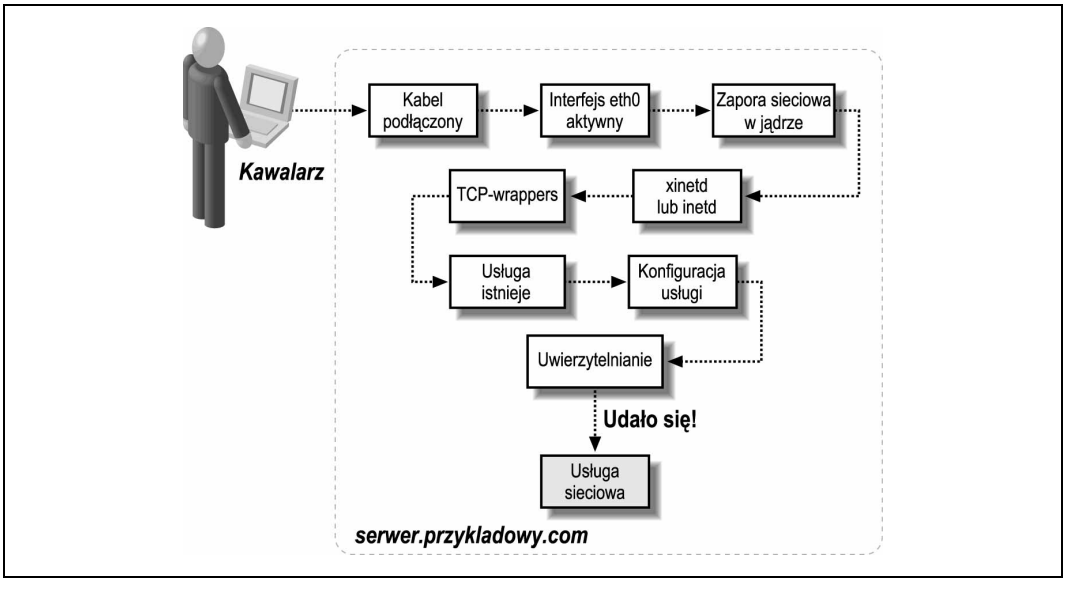
Indywidualne usługi sieciowe

Dowolna usługa sieciowa, np. *sshd* lub *ftpd* może posiadać własne mechanizmy kontroli dostępu. Na przykład, program *sshd* obsługuje słowo kluczowe `AllowUsers`, program *ftpd* korzysta z pliku */etc/ftpaccess*, a inne usługi wymagają uwierzytelnienia użytkownika.

Kiedy nadchodzi żądanie dostępu do usługi, wszystkie te elementy dochodzą do głosu. Załóżmy, że użytkownik *kawalarz* próbuje skorzystać z usługi FTP systemu *serwer.przykladowy.com* przez konto użytkownika *janek*. Sytuację przedstawiono na rysunku 3.1.

Gdy *serwer.przykladowy.com* jest podłączony fizycznie do sieci…
oraz interfejs sieciowy jest włączony…
oraz zapora sieciowa w jądrze wpuszcza pakiety FTP z hosta *kawalarza*…
oraz superdemon jest uruchomiony…
oraz superdemon przyjmuje połączenia FTP z komputera *kawalarza*…
oraz program *ftpd* jest zainstalowany i wykonywalny…
oraz konfiguracja *ftpd* w pliku */etc/ftpaccess* zezwala na połączenie…
oraz *kawalarz* uwierzytelnia się jako *janek*…

— wówczas połączenie uda się (jeśli nic innego się nie stanie, np. nie wystąpi awaria sieci).



Rysunek 3.1. Warstwy zabezpieczeń, przez które przechodzi połączenie przychodzące

Administrator systemu musi być świadomy wszystkich tych poziomów kontroli. W niniejszym rozdziale zostaną omówione:

ifconfig

Niskopoziomowy program do sterowania, włączania, wyłączania i ustawiania parametrów interfejsów sieciowych.

xinetd

Superdemon odpowiadający za wywołanie innych demonów. Jego działanie zależy od plików konfiguracyjnych, zazwyczaj znajdujących się w katalogu */etc/xinetd.d* (jeden plik odpowiada jednej usłudze). Na przykład, plik */etc/xinetd.d/finger* określa sposób wywołania na żądanie demona usługi finger:

/etc/xinetd.d/finger:

```
service finger
{
   server = /usr/sbin/in.fingerd    ścieżka do pliku wykonywalnego
   user = nobody                    usługa uruchamiana jako 'nobody'
   wait = no                        usługa uruchamiana wielowątkowo
   socket_type = stream             usługa na podstawie gniazda strumieniowego
}
```

Program *xinetd* jest wykorzystywany w dystrybucji Red Hat.

inetd

inetd jest starszym odpowiednikiem superdemona *xinetd*. Jego konfigurację opisuje plik */etc/inetd.conf*, w którym jednej usłudze odpowiada jeden wiersz. Analogiczny do przedstawionej wyżej konfiguracji usługi finger w tym przypadku wyglądałby następująco:

/etc/inetd.conf:

```
finger  stream  tcp  nowait  nobody  /usr/sbin/in.fingerd  in.fingerd
```

Program *inetd* jest wykorzystywany w dystrybucji SuSE.

TCP-wrappers

Warstwa kontroli dostępu do systemu ze strony określonych komputerów lub domen oraz na podstawie innych kryteriów. Zasady dostępu są opisane w plikach */etc/hosts.allow* (połączenia przyjmowane) oraz */etc/hosts.deny* (połączenia odrzucane). Na przykład, aby zablokować wszystkie połączenia z usługą *finger*, w pliku */etc/hosts.deny* wpiszemy:

```
finger : ALL : DENY
```

Żeby zaś zezwolić na połączenia z usługą *finger* tylko z komputerów domeny *przyjazna.org*, wpiszemy w pliku */etc/hosts.allow*:

```
finger : *.przyjazna.org
finger: ALL : DENY
```

Nie będziemy tutaj przytaczać całej składni tych plików, ponieważ jest ona opisana na stronie podręcznika *hosts.allow(5).* Należy jednak pamiętać, że mechanizm TCP-wrappers obsługuje również funkcję sprawdzania tożsamości przez protokół IDENT, potrafi uruchamiać dowolne programy zewnętrzne oraz realizować wiele innych ważnych zadań. Oprogramowanie TCP-wrappers jest obecne w obu dystrybucjach, Red Hat i SuSE.

Wszystkie receptury opisane w niniejszym rozdziale mają istotną wadę: nie ograniczają dostępu na podstawie hosta, lecz na podstawie źródłowego adresu IP. Na przykład, można określić, że do danej usługi — świadczonej przez nasz system — może mieć dostęp tylko host o numerze IP 121.108.19.42. Adres źródłowy można jednak w prosty sposób sfałszować. Nasze ograniczenia teoretycznie może obejść komputer, który podszyje się pod adres 121.108.19.42. Jeśli naprawdę chcemy kontrolować dostęp na podstawie hosta, a nie adresu źródłowego, lepiej zastosować mechanizm kryptograficznego uwierzytelniania hosta, np. uwierzytelnianie klienta lub serwera przez protokół SSH albo protokół IPSec.

3.1. Wyświetlanie interfejsów sieciowych

Problem

Chcemy wyświetlić listę wszystkich interfejsów sieciowych.

Rozwiązanie

W celu uzyskania listy wszystkich interfejsów, włączonych i wyłączonych, których sterowniki są załadowane, wydajemy polecenie:

```
$ ifconfig -a
```

Aby wyświetlić tylko włączone interfejsy, napiszemy:

```
$ ifconfig
```

Do wyświetlania informacji o jednym interfejsie — zazwyczaj *eth0* — służy polecenie:

```
$ ifconfig eth0
```

Omówienie

Gdy nie jesteśmy zalogowani jako użytkownik root, polecenia *ifconfig* może nie być w ścieżce dostępu. Wówczas próbujemy wpisać */sbin/ifconfig*.

Polecenie *ifconfig* wywołane z opcją `-a` wyświetla wszystkie interfejsy, bez względu na to, czy są włączone czy nie. Nie wyświetla jednak informacji o interfejsach, których sterowniki nie są załadowane. Załóżmy, że w komputerze zainstalowano dwie karty ethernetowe (*eth0* i *eth1*) różnych producentów, obsługiwane przez różne sterowniki. W systemie Linux skonfigurowano jednak tylko jeden sterownik (*eth0*), to znaczy dla tej jednej karty istnieje plik */etc/sysconfig/network-scripts/ifcfg-**. Z drugiego interfejsu zazwyczaj nie korzystamy. W takim przypadku polecenie `ifconfig -a` nie wyświetli drugiego interfejsu dopóty, dopóki nie wpiszemy `ifconfig eth1` (co spowoduje załadowanie odpowiedniego sterownika).

Zobacz również

ifconfig(8).

3.2. Włączanie i wyłączanie interfejsu sieciowego

Problem

Chcemy zablokować wszelkie połączenia przychodzące i wychodzące z siecią obsługiwaną przez dany interfejs sieciowy.

Rozwiązanie

W celu wyłączenia jednego interfejsu sieciowego, np. *eth0*, napiszemy:

```
# ifconfig eth0 down
```

W celu włączenia jednego interfejsu sieciowego, np. *eth0*, napiszemy:

```
# ifconfig eth0 up
```

W celu wyłączenia całej komunikacji sieciowej wystarczy wpisać:

```
# /etc/init.d/network stop
```

lub:

```
# service network stop       Red Hat
```

W celu włączenia obsługi sieci wpisujemy:

```
# /etc/init.d/network start
```

lub:

```
# service network start       Red Hat
```

Omówienie

W Linuksie istnieją trzy poziomy abstrakcji, na których można włączyć lub wyłączyć interfejsy sieciowe (oprócz fizycznego odłączenia kabla sieciowego):

/sbin/ifconfig

Najniższy poziom — umożliwia włączenie lub wyłączenie jednego interfejsu sieciowego. Program *ifconfig* ułatwia również wykonywanie innych czynności konfiguracyjnych związanych z interfejsem.

/sbin/ifup, */sbin/ifdown*

Te skrypty średniego poziomu operują na pojedynczych interfejsach sieciowych i służą odpowiednio do ich włączania lub wyłączania przez wywołanie programu *ifconfig* z właściwymi argumentami. Inicjalizują również obsługę protokołu DHCP i odpowiadają za szereg innych funkcji. Użytkownik rzadko korzysta z nich bezpośrednio.

/etc/init.d/network

Ten nadrzędny skrypt operuje na wszystkich interfejsach sieciowych. W zależności od potrzeb uruchamia skrypty *ifup* lub *ifdown* dla poszczególnych interfejsów oraz realizuje dodatkowe funkcje: dodaje trasy, tworzy pliki blokujące informujące o włączonej obsłudze sieci i wiele innych. Włącza lub wyłącza nawet interfejs pseudosieci, co — jeśli zależało nam tylko na zablokowaniu ruchu zewnętrznego — może nie być pożądane.

Skrypty *ifup*, *ifdown* oraz *network* są dość krótkie i warto je przestudiować.

Zobacz również

ifconfig(8). Strona podręcznika usernetctl(8) opisuje sposób modyfikacji przez nieuprzywilejowanego użytkownika parametrów interfejsów sieciowych obsługiwanych przez skrypty *ifup* i *ifdown* (przy przyzwoleniu administratora).

3.3. Włączanie i wyłączanie usługi (xinetd)

Problem

Chcemy uniemożliwić uruchamianie określonej usługi TCP przez demon *xinetd*.

Rozwiązanie

Jeśli nazwa usługi brzmi „mojausluga”, otwieramy jej plik konfiguracyjny */etc/xinetd/mojausluga* lub plik */etc/xinetd.conf* i do parametrów usługi dodajemy wpis:

```
disable = yes
```

Na przykład, aby wyłączyć usługę Telnet, w pliku */etc/xinetd.d/telnet* wpisujemy:

```
service telnet
{
    ...
    disable = yes
}
```

Następnie wymuszamy na programie *xinetd* wprowadzenie zmian przez ponowne odczytanie plików konfiguracyjnych:

```
# kill -USR2 `pidof xinetd`
```

W celu włączenia usługi, usuwamy wpis `disable` i ponownie wysyłamy sygnał `SIGUSR2`.

Omówienie

Zamiast wyłączać usługę, można po prostu usunąć jej plik konfiguracyjny programu *xinetd* (np. */etc/xinetd.d/telnet*), a nawet usunąć z komputera plik wykonywalny odpowiedzialny za usługę. Takie operacje trudno jednak cofnąć (natomiast nie należy usuwać pliku wykonywalnego przy pozostawieniu włączonej usługi w plikach konfiguracyjnych. *xinetd* będzie próbował uruchamiać usługę, co spowoduje zgłaszanie błędów).

Ewentualnie można użyć programu *ipchains* lub *iptables* [2.7], jeśli chcemy, aby usługa działała, ale była dostępna tylko dla użytkowników łączących się spod wybranych adresów źródłowych. Niektóre usługi mogą również posiadać własne mechanizmy kontroli dostępu, za pomocą których można ograniczyć dostęp do wybranych adresów klientów.

Zobacz również

xinetd(8). Strona programu *xinetd* znajduje się pod adresem *http://www.synack.net/xinetd*.

3.4. Włączanie i wyłączanie usługi (inetd)

Problem

Chcemy uniemożliwić uruchamianie określonej usługi TCP przez demon *inetd*.

Rozwiązanie

W celu wyłączenia usługi wstawiamy znak komentarza (`#`) przed wierszem tej usługi w pliku */etc/inetd.conf*. Po zmianie przykładowy wpis dla usługi Telnet wygląda następująco:

/etc/inetd.conf:
```
# telnet  stream  tcp  nowait  root  /usr/sbin/in.telnetd  in.telnetd
```

Następnie wymuszamy na programie *inetd* wprowadzenie zmian przez ponowne odczytanie plików konfiguracyjnych (tutaj znak krzyżyka (`#`) jest znakiem zachęty użytkownika root, a nie symbolem komentarza):

```
# kill -HUP `pidof inetd'
```

W celu włączenia usługi, usuwamy odpowiedni znak komentarza i ponownie wysyłamy sygnał `SIGHUP`.

Omówienie

Zamiast wyłączać usługę, można po prostu usunąć cały wiersz w pliku konfiguracyjnym programu *inetd*, a nawet usunąć z komputera plik wykonywalny odpowiedzialny za usługę. Takie operacje trudno jednak cofnąć (natomiast nie należy usuwać pliku wykonywalnego przy pozostawieniu włączonej usługi w plikach konfiguracyjnych — *inetd* będzie próbował uruchamiać usługę, co spowoduje zgłaszanie błędów). Ewentualnie można użyć programu *ipchains* lub *iptables* [2.6], jeśli chcemy, aby usługa działała, ale nie była dostępna ze zdalnych systemów.

Zobacz również

inetd(8), inetd.conf(5).

3.5. Dodawanie nowej usługi (xinetd)

Problem

Chcemy dodać nową usługę sieciową, którą ma kontrolować demon *xinetd*.

Rozwiązanie

Tworzymy nowy plik konfiguracyjny w katalogu */etc/xinetd.d*, zawierający przynajmniej następujące informacje:

```
service NAZWA_USŁUGI                    Nazwa z /etc/services; patrz services(5)
{
    server = /ŚCIEŻKA/DO/SERWERA        Plik wykonywalny usługi
    server_args = DOWOLNE_ARGUMENTY     Dowolne argumenty; tę opcję można pominąć
    user = UŻYTKOWNIK                   Użytkownik, z prawami którego uruchamiana jest usługa
    socket_type = TYP                   stream, dgram, raw lub seqpacket
    wait = YES lub NO                   yes = działanie jednowątkowe;
                                        no = działanie wielowątkowe
```

Określamy nazwę usługi *NAZWA_USŁUGI* i wymuszamy na programie *xinetd* ponowne odczytanie plików konfiguracyjnych. [3.3]

Omówienie

Aby utworzyć plik konfiguracyjny usługi dla demona *xinetd*, musimy oczywiście wiedzieć, jakie właściwości ma mieć usługa i jak ma być uruchamiana. Czy wykorzystuje gniazda typu strumieniowego czy datagramowego? Czy ma działać jedno- czy wielowątkowo? Z jakimi ewentualnymi argumentami uruchamiany jest program serwera?

Pliki konfiguracyjne programu *xinetd* pozwalają wprowadzać bardzo wiele dodatkowych słów kluczowych i wartości. Szczegółowe informacje można znaleźć na stronie podręcznika xinetd.conf(5).

xinetd odczytuje wszystkie pliki z katalogu */etc/xinetd.d* tylko wtedy, gdy tak określono w pliku */etc/xinetd.conf* za pomocą następującego wiersza:

```
includedir /etc/xinetd.d
```

Warto upewnić się — zaglądając do wspomnianego pliku — jaką wartość przypisano zmiennej `includedir`.

Zobacz również

xinetd(8), xinetd.conf(5), services(5). Strona programu *xinetd* znajduje się pod adresem *http://www.synack.net/xinetd*.

3.6. Dodawanie nowej usługi (inetd)

Problem

Chcemy dodać nową usługę sieciową, którą ma kontrolować demon *inetd*.

Rozwiązanie

Dodajemy nowy wiersz do pliku */etc/inetd.conf*, zawierający dane w następującej postaci:

```
NAZWA_USŁUGI   TYP_GNIAZDA  PROTOKÓŁ WĄTKOWANIE  UŻYTKOWNIK
    /ŚCIEŻKA/DO/SERWERA  ARGUMENTY
```

Następnie wymuszamy na programie *inetd* ponowne odczytanie pliku konfiguracyjnego */etc/inetd.conf*. [3.4]

Omówienie

Wiersz odpowiadający usłudze w pliku */etc/inetd.conf* zawiera następujące wartości:

1. ***Nazwa usługi***. Nazwa wymieniona w pliku */etc/services*. Jeśli w pliku tym nie ma żądanej nazwy, dodajemy odpowiedni wpis określający nazwę usługi, numer portu i protokół. Więcej informacji można znaleźć na stronie podręcznika services(5).
2. ***Typ gniazda***. Jedna z następujących wartości: `stream`, `dgram`, `raw`, `rdm` lub `seqpacket`.
3. ***Protokół***. Zazwyczaj `tcp` lub `udp`.

4. ***Wątkowanie***. W przypadku usługi jednowątkowej: `wait`; w przypadku usługi wielowątkowej: `nowait`.
5. ***Użytkownik***. Z prawami tego użytkownika jest uruchamiana usługa.
6. ***Ścieżka*** do pliku wykonywalnego serwera.
7. ***Argumenty serwera*** rozdzielone znakami białymi. Wyliczanie argumentów należy rozpocząć od argumentu zerowego, tj. samej nazwy pliku serwera. Na przykład, w przypadku serwera */usr/sbin/in.telnetd* argumentem zerowym byłby *in.telnetd*.

Oto pełny przykład:

```
telnet   stream  tcp  nowait  root  /usr/sbin/in.telnetd  in.telnetd
```

W wierszu pliku *inetd.conf* mogą się znaleźć jeszcze inne informacje, np. wielkości buforów, adres lokalnego hosta, na którym odbywa się nasłuchiwanie itd. Szczegółowe informacje można znaleźć na stronie podręcznika.

Zobacz również

inetd(8), inetd.conf(5), services(5).

3.7. Ograniczanie dostępu zdalnym użytkownikom

Problem

Chcemy, aby do usługi TCP mieli dostęp tylko wybrani zdalni użytkownicy. Nie możemy przewidzieć, z jakich hostów będą nawiązywane połączenia.

Rozwiązanie

Za pomocą zapory sieciowej blokujemy port TCP tej usługi obsługujący połączenia przychodzące. [2.6] Uruchamiamy serwer SSH i pozwalamy użytkownikom nawiązywać połączenia tunelowane przez SSH z przekazaniem portu. W przydzielaniu lub blokowaniu dostępu będą od tej pory wykorzystywane mechanizmy uwierzytelniania SSH. Dostęp przez protokół SSH proponujemy przez udostępnienie klucza publicznego.

Na przykład, aby uzyskać dostęp do serwera grup dyskusyjnych (port TCP 119) naszego serwera *serwer.przykladowy.com*, użytkownik korzystający z hosta *mojklient* będzie musiał zbudować następujący tunel SSH z dowolnego portu lokalnego (np. 23456) do serwera grup dyskusyjnych:

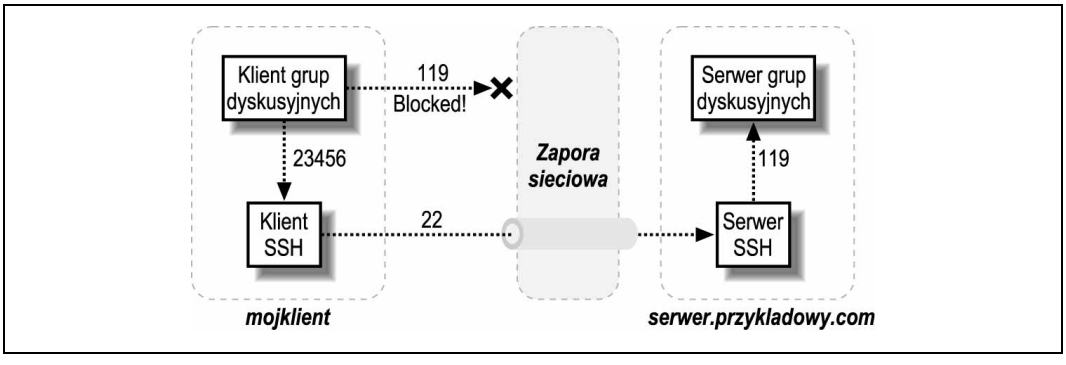
```
mojklient$ ssh -f -N -L 23456:serwer.przykladowy.com:119 serwer.przykladowy.com
```

Teraz musi już tylko połączyć się z tym tunelem. Przykład dla czytnika grup dyskusyjnych *tin*:

```
mojklient$ export NNTPSERVER=localhost
mojklient$ tin -r -p 23456
```

Omówienie

Tunelowanie SSH (czyli przekazywanie portów) polega na przekierowaniu połączenia TCP w taki sposób, że przebiega ono przez klienta i serwer SSH w sposób niemal niewidoczny[1]. [6.14] Tunel ten łączy port lokalny ze zdalnym, szyfrując ruch w punkcie wejściowym i rozszyfrowując w punkcie wyjściowym. Na przykład, tunel NNTP (usługa grup dyskusyjnych, port 119) działa na następującej zasadzie: czytnik grup dyskusyjnych komunikuje się z klientem SSH, który przekazuje dane tunelem do serwera SSH, a ten z kolei komunikuje się z serwerem NNTP (rysunek 3.2).



Rysunek 3.2. Tunelowanie protokołu NNTP przez SSH

Blokując port usługi (119), uniemożliwiamy realizowanie z tym portem połączeń z zewnątrz. Jednak SSH wykorzystuje inny port (22), który nie jest zablokowany zaporą sieciową.

Ewentualnie można sprawdzić, czy dana usługa nie posiada własnych mechanizmów uwierzytelniających. Na przykład, serwer *wu-ftpd* korzysta z pliku */etc/ftpaccess*, demon *sshd* obsługuje słowo kluczowe `AllowUsers` itd.

Zobacz również

ssh(1), sshd(8), tin(1).

[1] Połączenie takie nie jest przezroczyste dla usług, w przypadku których istotne znaczenie mają parametry gniazd, np. FTP. Jednak w innych przypadkach połączenie przez SSH jest „widziane" przez usługę jak połączenie standardowe.

3.8. Ograniczanie dostępu zdalnym hostom (xinetd)

Problem

Chcemy, aby tylko określone zdalne hosty miały dostęp do usługi TCP przez *xinetd*.

Rozwiązanie

Stosujemy słowa kluczowe `only_from` i `no_access`:

```
service ftp
{
    only_from = 192.168.1.107
    ...
}

service smtp
{
    no_access = haker.zly.org
    ...
}
```

Następnie resetujemy *xinetd*, tak aby zmiany zostały wprowadzone. [3.3]

Omówienie

To chyba najprostszy sposób kontrolowania dostępu do poszczególnych usług. Oczywiście, można tak kontrolować tylko usługi obsługiwane przez demon *xinetd*.

Słowa kluczowe `only_from` oraz `no_access` mogą występować wielokrotnie w sekcji jednej usługi:

```
{
     no_access = haker.zly.org      blokujemy konkretny host
     no_access += 128.220.          blokujemy wszystkie hosty danej sieci
}
```

Jeśli host, który próbuje nawiązać połączenie, jest wymieniony na obu listach (`only_from` i `no_access`), *xinetd* podejmuje jedno z następujących działań:

- Jeśli host pasuje do wpisów w obu listach, pod uwagę jest brany wpis **bardziej specyficzny**. Na przykład, określenie 128.220.13.6 jest bardziej specyficzne niż 128.220.13.
- Jeśli komputer pasuje do wpisów w obu listach i wpisy te są tak samo specyficzne, *xinetd* zgłasza taki stan jako błąd konfiguracyjny i nie uruchamia usługi w danym przypadku.

Zatem przy następującym wpisie:

```
service cokolwiek
{
    no_access = 128.220.          haker.zly.org       klient.przykladowy.com
    only_from = 128.220.10.       .zly.org            klient.przykladowy.com
}
```

połączenia z adresu 128.220.10.3 są dozwolone, a te z adresu 128.220.11.2 — blokowane. Podobnie host *haker.zly.org* nie może się połączyć, ale inne komputery centralne z domeny *zly.org* mogą. *klient.przykladowy.com* jest wpisem nieprawidłowym i połączenia z tego komputera zostaną zablokowane. Ponadto dowolny host, który nie pasuje do żadnego z wpisów, nie zdoła się połączyć z usługą.

Zobacz również

xinetd.conf(5).

3.9. Ograniczanie dostępu zdalnym hostom (xinetd z biblioteką libwrap)

Problem

Chcemy, aby tylko określone zdalne hosty miały dostęp do usługi TCP przez program *xinetd* skompilowany z obsługą biblioteki *libwrap*.

Rozwiązanie

Kontrolujemy dostęp, używając plików */etc/hosts.allow* i */etc/hosts.deny*. Na przykład, aby zezwolić na połączenia Telnet tylko z adresu 192.168.1.100 oraz hostom z domeny *przykladowy.com*, do pliku */etc/hosts.allow* dodajemy następujące wpisy:

```
in.telnetd : 192.168.1.100
in.telnetd : *.przykladowy.com
in.telnetd : ALL : DENY
```

Następnie resetujemy *xinetd*, tak aby zmiany zostały wprowadzone. [3.3]

Omówienie

Ta metoda przeznaczona jest dla osób, którym zależy na przeniesieniu kontroli dostępu z mechanizmów specyficznych dla demona *xinetd* [3.8] do plików */etc/hosts.allow* i */etc/hosts.deny* albo którym bardziej odpowiada składnia i możliwości tych plików. Składnia ta daje bardzo szerokie możliwości definiowania hostów i sieci, którym zezwalamy lub nie zezwalamy na połączenia ze specyficznymi usługami świadczonymi przez nasz system.

Przedstawiona receptura działa tylko wtedy, gdy program *xinetd* skompilowano z włączoną obsługą biblioteki libwrap. W celu sprawdzenia, czy program *xinetd* został tak skompilowany, sprawdzamy wynik polecenia:

```
$ strings /usr/sbin/xinetd | grep libwrap
libwrap refused connection to %s from %s
%s started with libwrap options compiled in.
```

Jeśli zobaczymy charakterystyczne wiersze w formacie `printf` — takie jak wyżej — posiadamy wersję programu *xinetd* z wkompilowaną obsługą biblioteki *libwrap*.

Zobacz również

xinetd(8), hosts.allow(5).

3.10. Ograniczanie dostępu zdalnym hostom (xinetd w połączeniu z tcpd)

Problem

Chcemy, aby tylko określone zdalne hosty miały dostęp do usługi TCP przez program *xinetd*, ale programu *xinetd* **nie** skompilowano z obsługą biblioteki *libwrap*.

Rozwiązanie

Definiujemy reguły kontroli dostępu w plikach */etc/hosts.allow* i (lub) */etc/hosts.deny*. Na przykład, aby zezwolić na połączenia Telnet tylko z adresu 192.168.1.100 oraz hostom z domeny *przykladowy.com*, do pliku */etc/hosts.allow* dodajemy następujące wpisy:

```
in.telnetd : 192.168.1.100
in.telnetd : *.przykladowy.com
in.telnetd : ALL : DENY
```

Następnie modyfikujemy pliki */etc/xinetd.conf* lub */etc/xinetd.d/nazwaluslugi* tak, aby zamiast danej usługi był uruchamiany program *tcpd*:

```
Stara wersja pliku /etc/xinetd.conf lub /etc/xinetd.d/telnet:
service telnet
{
    ...
    flags = ...
    server = /usr/sbin/in.telnetd
    ...
}
```

Nowa wersja pliku /etc/xinetd.conf lub /etc/xinetd.d/telnet:

```
service telnet
{
    ...
    flags = ... NAMEINARGS
    server = /usr/sbin/tcpd
    server_args = /usr/sbin/in.telnetd
    ...
}
```

Następnie resetujemy *xinetd*, tak aby zmiany zostały wprowadzone. [3.3]

Omówienie

Przedstawiona receptura dotyczy tylko tych rzadkich przypadków, gdy z jakiegoś powodu nie chcemy korzystać z wbudowanych mechanizmów kontroli dostępu programu *xinetd* [3.8], a program *xinetd* nie posiada wkompilowanej obsługi biblioteki *libwrap*. Opisana metoda jest wzorowana na pierwotnej metodzie kontroli dostępu stosowanej w programie *inetd* i wykorzystującej mechanizm TCP-wrappers. [3.11]

W pliku należy umieścić znacznik `NAMEINARGS`, w wyniku którego program *xinetd* szuka nazwy pliku wykonywalnego w opcji `server_args` (w tym przypadku jest to */usr/sbin/in.telnetd*).

Zobacz również

xinetd(8), hosts.allow(5), tcpd(8).

3.11. Ograniczanie dostępu zdalnym hostom (inetd)

Problem

Chcemy, aby tylko określone zdalne hosty miały dostęp do usługi TCP przez *inetd*.

Rozwiązanie

Używamy demona *tcpd* i określamy reguły dostępu w plikach */etc/hosts.allow* i (lub) */etc/hosts.deny*. Poniżej przedstawiono przykład „opakowania” (ang. *wrap*) demona usługi Telnet, *in.telnetd* tak, aby zezwolić na połączenia tylko z adresu 192.168.1.100 oraz hostom z domeny *przykladowy.com*, do pliku */etc/hosts.allow* dodajemy następujące wpisy:

```
in.telnetd : 192.168.1.100
in.telnetd : *.przykladowy.com
in.telnetd : ALL : DENY
```

Następnie modyfikujemy odpowiednie pliki konfiguracyjne, wpisując zamiast nazwy usługi słowo *tcpd* i restartujemy program *inetd*.

Omówienie

Pliki kontroli dostępu */etc/hosts.allow* i */etc/hosts.deny* zawierają definicje reguł, na podstawie których zdalne hosty uzyskują dostęp do lokalnych usług TCP. Demon kontroli dostępu *tcpd* przetwarza te reguły i określa, czy w danym przypadku uruchamiać usługę opisaną regułami.

Najpierw ustawiamy reguły kontroli dostępu w plikach */etc/hosts.allow* i (lub) */etc/hosts.deny*. Następnie modyfikujemy plik */etc/inetd.conf* tak, aby dana usługa była uruchamiana przez *tcpd*:

Stara wersja pliku /etc/inetd.conf:
```
telnet  stream  tcp  nowait  root  /usr/sbin/in.telnetd  in.telnetd
```

Nowa wersja pliku /etc/inetd.conf:
```
telnet  stream  tcp  nowait  root  /usr/sbin/tcpd  /usr/sbin/in.telnetd
```

Następnie restartujemy *inetd*, tak aby zmiany zostały wprowadzone. [3.4]

Zobacz również

hosts.allow(5), tcpd(8), inetd.conf(5).

3.12. Ograniczanie dostępu o określonych porach dnia

Problem

Chcemy, aby usługa była dostępna tylko o określonych porach dnia.

Rozwiązanie

W przypadku programu *xinetd* używamy obsługiwanego przez ten program atrybutu `access_times`. Na przykład, aby usługa *telnetd* była dostępna codziennie od 8:00 do 17:00, napiszemy:

Plik /etc/xinetd.conf lub /etc/xinetd.d/telnet:
```
service telnet
{
    ...
    access_times = 8:00-17:00
}
```

W przypadku demona *inetd* musimy zaimplementować taką funkcję własnoręcznie. W tym celu zastosujemy makroprocesor *m4* i usługę *cron*. Najpierw wymyślamy łańcuchy znaków określające pory dnia, np. „praca" oznacza godzinę 8:00, a „odpoczynek" godzinę 17:00. Następnie piszemy skrypt (np. *uslugi-inetd*, który za pomocą makroprocesora *m4* wybiera wiersze z pliku szablonu, tworzy plik konfiguracyjny demona *inetd* i wymusza na demonie *inetd* odczytanie tego pliku):

/usr/local/sbin/uslugi-inetd:
```
#!/bin/sh
m4 "$@" /etc/inetd.conf.m4 > /etc/inetd.conf.$$
mv /etc/inetd.conf.$$ /etc/inetd.conf
kill -HUP `pidof inetd`
```

Kopiujemy oryginalny plik */etc/inetd.conf* do pliku szablonu */etc/inetd.conf.m4*. Edytujemy szablon tak, aby usługi były włączane warunkowo, w zależności od wartości pewnego parametru, np. `PORA_DNIA`. Przykładowy, oryginalny wiersz dotyczący usługi Telnet:

```
telnet  stream  tcp  nowait  root  /usr/sbin/tcpd  in.telnetd
```

wyglądałaby następująco:

```
ifelse(PORA_DNIA,praca,telnet  stream  tcp  nowait  root  /usr/sbin/tcpd
    in.telnetd)
```

Powyższy zapis należy interpretować tak: „jeśli zmienna `PORA_DNIA` ma wartość `praca`, dołącz następujący wiersz dotyczący usługi Telnet; w przeciwnym razie nie dołączaj jej". Na koniec wprowadzamy odpowiednie wpisy w pliku *crontab*, które będą włączały lub wyłączały usługi o określonych porach przez ustawienie parametru `PORA_DNIA`:

```
0  8 * * * /usr/local/sbin/uslugi-inetd -DPORA_DNIA=praca
0 17 * * * /usr/local/sbin/uslugi-inetd -DPORA_DNIA=odpoczynek
```

Omówienie

W przypadku demona *xinetd* możemy z łatwością kontrolować dostęp do poszczególnych usług za pomocą parametru `access_times`. Czas określa się w formacie 24-godzinnym.

W przypadku demona *inetd* sprawa jest bardziej skomplikowana. Żeby włączyć lub wyłączyć usługę o określonej porze dnia, musimy podmienić plik konfiguracyjny. Przedstawioną recepturę można w prosty sposób rozszerzyć o obsługę dodatkowych parametrów i wartości podobnych do `PORA_DNIA`. Zauważmy, że w programie *xinetd* definiowane są przedziały czasowe, natomiast w naszej recepturze dla programu *inetd* podajemy tylko pory, o których *cron* uruchamia skrypt *uslugi-inetd*.

Zobacz również

xinetd.conf(5), inetd.conf(5), m4(1), crontab(5).

3.13. Ograniczanie określonym hostom dostępu do serwera SSH

Problem

Chcemy ograniczyć dostęp do serwera *sshd* konkretnym, zdalnym hostom.

Rozwiązanie

Wykorzystujemy wbudowaną w program *sshd* obsługę mechanizmu TCP-wrappers. Wystarczy do plików */etc/hosts.allow* i */etc/hosts.deny* dodać odpowiednie reguły i jako usługę wpisać w nich `sshd`. Na przykład, aby tylko host 192.168.0.37 miał dostęp do serwera SSH, wstawimy następujące wiersze do pliku */etc/hosts.allow*:

```
sshd: 192.168.0.37
sshd: ALL : DENY
```

Omówienie

Wywołanie *tcpd* czy innego programu nie jest konieczne, ponieważ program *sshd* sam odczytuje reguły w plikach */etc/hosts.allow* i */etc/hosts.deny*.

Obsługa mechanizmu TCP-wrappers w programie *sshd* jest opcjonalna i włącza się ją w czasie kompilacji. W dystrybucji Red Hat 8.0 opcja ta została wkompilowana w *sshd*, ale w dystrybucji SuSE już nie. Jeśli nie jesteśmy pewni, czy nasza wersja programu *sshd* obsługuje mechanizm TCP-wrappers albo gdy *sshd* najwyraźniej ignoruje wpisy w plikach */etc/hosts.allow* i */etc/hosts.deny*, wykonujemy następujący test:

```
$ strings /usr/sbin/sshd | egrep 'hosts\.(allow|deny)'
/etc/hosts.allow
/etc/hosts.deny
```

Jeśli program *egrep* nie zwraca żadnych wyników, obsługa mechanizmu TCP-wrappers nie została wkompilowana w naszą wersję programu *sshd*. Pozostaje wtedy pobranie oprogramowania OpenSSH z adresu *http://www.openssh.com* (lub pobranie pakietu RPM z kodem źródłowym przygotowanego przez producenta naszej dystrybucji) i przebudowanie go:

```
$ ./configure --with-libwrap  ...inne wymagane opcje...
$ make
# make install
```

Zobacz również

sshd(8), hosts_access(5).

3.14. Ograniczanie dostępu do serwera SSH w zależności od konta systemowego

Problem

Chcemy, aby przychodzące połączenia SSH były dozwalane tylko niektórym kontom systemowym.

Rozwiązanie

W pliku konfiguracyjnym */etc/ssh/sshd_config* wykorzystujemy słowo kluczowe `AllowUsers`. Na przykład, aby zezwolić na połączenia SSH z dowolnego miejsca na świecie z kontami użytkowników `kowalski` i `nowak` — i tylko z tymi kontami — użyjemy wpisu:

/etc/ssh/sshd_config:
```
AllowUsers kowalski nowak
```

Dopuszczając połączenia z adresu *zdalny.przykladowy.com* z kontem `kowalski` i żadne inne połączenia przychodzące, napiszemy:

```
AllowUsers kowalski@zdalny.przykladowy.com
```

Zauważmy, że **nie** ma tu znaczenia zdalny użytkownik o nazwie *kowalski@zdalny.przykladowy.com*, a jedynie połączenie **ze** zdalnego hosta *zdalny.przykladowy.com* **z** lokalnym kontem *kowalski*.

Po zmodyfikowaniu pliku *sshd_config* restartujemy *sshd*, tak aby zmiany zostały uwzględnione.

Omówienie

Słowo `AllowUsers` umożliwia określenie listy kont lokalnych, dla których są dozwolone połączenia SSH. Jest to lista definitywna: jeśli jakieś konto nie jest na niej wymienione, nie będzie możliwe połączenie się z nim przez SSH.

Zapis w drugim z wyżej wymienionych formacie (`uzytkownik@host`) wygląda dość niefortunnie, bo przypomina adres e-mail lub określenie użytkownika zdalnego — a nie jest ani jednym, ani drugim. Wiersz:

```
AllowUsers uzytkownik@zdalnyhost
```

oznacza „zezwalaj zdalnemu systemowi o nazwie *zdalnyhost* na połączenia przez SSH z naszym lokalnym kontem o nazwie *uzytkownik*".

Sama obecność wpisu ze słowem kluczowym `AllowUsers` nie gwarantuje jeszcze dostępu do konta przez *sshd*: zdalny użytkownik musi jeszcze dokonać zwykłego uwierzytelniania (przez hasło, klucz publiczny itd.), nie mówiąc już o ewentualnych innych blokadach (reguły zapory sieciowej itp.).

Zobacz również

sshd_config(5).

3.15. Ograniczanie zasobów usług do określonych katalogów systemu plików

Problem

Chcemy ograniczyć zasoby usługi do określonego katalogu (i podkatalogów) systemu plików.

Rozwiązanie

Tworzymy ***klatkę*** *chroot*, uruchamiając zamiast usługi program GNU *chroot*. Jako argument tego programu podajemy nazwę pliku wykonywalnego usługi. Innymi słowy, zmieniamy następujący wpis:

Plik /etc/xinetd.conf lub /etc/xinetd.d/mojausluga:

```
service mojausluga
{
    ...
    server          = /usr/sbin/mojausluga -a -b
    ...
}
```

na następujący:

```
service mojausluga
{
    ...
    user = root
    server          = /usr/sbin/chroot
    server_args = /var/klatka /usr/sbin/mojausluga -a -b
    ...
}
```

Omówienie

Program *chroot* wymaga dwóch argumentów: nazwy katalogu i nazwy programu. Powoduje, że program zachowuje się tak, jakby dany katalog był katalogiem głównym systemu plików (/). Takie rozwiązanie uniemożliwia uzyskanie przez program dostępu do plików spoza hierarchii katalogu podanego jako argument *chroot* — czyli spoza „klatki chroot" — ponieważ z punktu widzenia programu *chroot* pliki te nie mają nazwy. Nawet jeśli program jest uruchamiany z przywilejami roota, tego ograniczenia nie można obejść. Funkcja systemowa wywoływana przez program *chroot* (o identycznej nazwie, `chroot`) działa jednokierunkowo: kiedy zostanie już wywołana, nie istnieje funkcja systemowa odwracająca jej działanie w kontekście wywołującego ją procesu lub jego procesów potomnych.

Klatka *chroot* działa najefektywniej, gdy program usługi zaraz po uruchomieniu przestaje działać z prawami użytkownika root — i wiele demonów można tak skonfigurować. Zamknięty w klatce *chroot* program uruchamiany z prawami roota może zaszkodzić w inny sposób: przez utworzenie i wykorzystanie nowego specjalnego pliku urządzenia lub wywołania funkcji systemowych niezwiązanych z systemem plików (np. `reboot`!).

Program działający w zwykłym środowisku może korzystać z plików niezwiązanych bezpośrednio z zadaniem, które wykonuje. To z kolei może ograniczyć możliwość praktycznego zastosowania programu *chroot*. Może być konieczne skopiowanie tak dużej części systemu plików do klatki *chroot*, że uzyskane w ten sposób zabezpieczenie przestaje być już tak silne, szczególnie jeśli kopiowane pliki mają charakter poufny (np. plik z hasłami na potrzeby uwierzytelniania) albo jeśli podlegają zmianom. W pierwszym przypadku lepiej byłoby, gdyby dana usługa sama w specjalny sposób obsługiwała mechanizm „klatki" — tak, aby program *chroot* dochodził do głosu dopiero po tym, jak program usługi uzyska już dostęp do wymaganych ogólnych zasobów systemowych. W drugim przypadku można utworzyć dowiązania sztywne do odpowiednich nazwanych plików spoza klatki *chroot*, jednak jest to możliwe tylko w przypadku plików znajdujących się na tym samym systemie plików co katalog klatki. Dowiązania symboliczne nie wystarczą, ponieważ program będzie z nich korzystał w kontekście klatki.

Program *chroot* musi być uruchamiany z prawami użytkownika root, a katalog klatki musi zawierać linuksową strukturę katalogów wystarczającą do uruchomienia usługi *mojausluga*. We wspomnianym przykładzie katalog */var/klatka* będzie musiał zawierać plik */var/klatka/usr/sbin/mojausluga*, katalog */var/klatka/lib* (wraz ze wszystkimi bibliotekami wykorzystywanymi przez usługę *mojausluga*) itd. W przeciwnym razie wystąpią błędy w rodzaju:

```
chroot: cannot execute nazwa_programu: No such file or directory
```

Konfiguracja nieco przypomina w tym przypadku pracę detektywa. Na przykład, aby zadziałało następujące proste polecenie:

```
# chroot /var/klatka /usr/bin/who
```

w katalogu */var/klatka* będą musiały zostać odwzorowane następujące wiersze:

/usr/bin/who
/lib/ld-linux.so.2
/lib/libc.so.6
/var/log/wtmp
/var/run/utmp.

W określeniu tego, które biblioteki współużytkowane i które pliki są wykorzystywane przez usługę, mogą pomóc programy *ldd* oraz *strings*, na przykład:

```
$ ldd /usr/sbin/mojausluga
   ...wynik...
$ strings /usr/sbin/mojausluga | grep /
   ...wynik...
```

Zobacz również

chroot(1), xinetd.conf(5), strings(1), ldd(1). Jeśli w systemie nie ma strony podręcznika *ldd*, informacje o sposobie jego użycia można uzyskać, wpisując polecenie `ldd --help`.

3.16. Zapobieganie atakom powodującym odmowę obsługi

Problem

Chcemy zapobiec atakom powodującym odmowę obsługi (ang. *Denial of Service — DoS*) i skierowanym na usługę sieciową.

Rozwiązanie

Konfigurując program *xinetd*, wykorzystujemy słowa kluczowe `cps`, `instances`, `max_load` i `per_source`.

```
Pliki /etc/xinetd.conf lub /etc/xinetd.d/mojausluga:
service mojausluga
{
    ...
    cps = 10 30          Ograniczenie do 10 połączeń na sekundę.
                         Jeśli przekroczone — odczekujemy 30 sekund.
    instances = 4        Ograniczenie do 4 jednoczesnych kopii procesu mojausluga.
    per_source = 2       Ograniczenie do 2 jednoczesnych sesji usługi na jeden adres IP.
                         Jeśli nie ma ograniczeń, wpisujemy UNLIMITED (ustawienie domyślne).
    max_load = 3.0       Odrzucamy nowe żądania, jeśli średnie obciążenie systemu
                         przez minutę przekracza 3.0.
}
```

W przypadku programu *inetd* używamy opcji `-R`, która umożliwia określenie, ile razy najwięcej dana usługa może być wywoływana przez 1 minutę. Domyślna wartość to 256.

Omówienie

Te słowa kluczowe mogą występować pojedynczo lub w kombinacjach. Słowo kluczowe `cps` ogranicza liczbę połączeń z daną usługą na sekundę. Jeśli limit zostanie przekroczony, program *xinetd* wyłącza usługę na czas określony (w sekundach) drugim argumentem.

Słowo kluczowe `instances` ogranicza liczbę jednoczesnych kopii danej usługi. Konfigurację domyślną — brak ograniczeń — możemy również wyrazić jawnie, wpisując:

```
instances = UNLIMITED
```

Słowo kluczowe `per_source` działa podobnie: zamiast ograniczać liczbę kopii procesu usługi, ogranicza liczbę sesji dla danego źródłowego adresu IP. Na przykład, w celu uniemożliwienia zdalnemu hostowi nawiązywania wielu połączeń FTP z naszym systemem, napiszemy:

Plik /etc/xinetd.conf lub /etc/xinetd.d/ftp:
```
service ftp
{
    ...
    per_source = 1

}
```

Wreszcie, słowo kluczowe `max_load` wyłącza usługę, jeśli średnie obciążenie systemu lokalnego przekroczy dopuszczalny limit, co zapobiega przeciążeniu procesora.

inetd jest mniej elastyczny: obsługuje tylko opcję `-R`, która ogranicza liczbę uruchomień usługi na minutę. Ograniczenie dotyczy każdej usługi indywidualnie. W przypadku jego przekroczenia, *inetd* zgłasza komunikat w formacie:

```
telnet/tcp server failing (looping), service terminated
```

Tak naprawdę świadczenie usługi nie kończy się (ang. *terminate*), tylko jest zawieszane na 10 minut. Czasu tego nie można zmienić.

Podobne funkcje posiadają niektóre zapory sieciowe. Na przykład, oprogramowanie *iptables* umożliwia ograniczenie całkowitej liczby połączeń przychodzących. Z drugiej strony, *iptables* nie obsługuje odpowiednika wspomnianego wyżej słowa kluczowego `per_source` — nie można ograniczyć liczby połączeń nawiązywanych z jednego adresu źródłowego.

Zobacz również

xinetd.conf(5).

3.17. Przekierowanie do innego gniazda

Problem

Chcemy przekierować połączenie do innego hosta i (lub) do innego portu tego samego lub innego komputera.

Rozwiązanie

Używamy słowa kluczowego `redirect` obsługiwanego przez program *xinetd*:

Plik /etc/xinetd.conf lub /etc/xinetd.d/ftp:
```
service mojausluga
```

```
{
    ...
    server = ścieżka do oryginalnej usługi
    redirect = adres_IP  numer_portu

}
```

Wymaga się słowa kluczowego `server`, ale jego wartość jest ignorowana. Program *xinetd* nie uruchamia usługi, jeśli nie wprowadzono słowa `server`, nawet gdy usługa będzie przekierowywana.

Omówienie

Na przykład, w celu przekierowania przychodzących połączeń z usługą *finger* (port 79) do innego komputera dostępnego pod adresem 192.168.14.21 napiszemy:

Plik /etc/xinetd.conf lub /etc/xinetd.d/ftp:

```
service finger
{
    ...
    server = /usr/sbin/in.fingerd
    redirect = 192.168.14.21 79

}
```

Oczywiście można przekierować połączenia do zupełnie innej usługi, np. *qotd* działającej na porcie 17:

```
service finger
{
    ...
    server = /usr/sbin/in.fingerd
    redirect = 192.168.14.21 17

}
```

Teraz zamiast informacji typowych dla usługi *finger* klient otrzyma zabawny „cytat dnia", jeśli na wspomnianym komputerze usługa *qotd* rzeczywiście działa. Można również przekierować żądania do innego portu tego samego komputera.

Zobacz również

xinetd.conf(5). Odpowiedni samouczek można znaleźć pod adresem *http://www.macsecurity.org/resources/xinetd/tutorial.shtml*.

3.18. Rejestrowanie przypadków korzystania z usług

Problem

Chcemy wiedzieć, kto korzysta z usług obsługiwanych przez demon *xinetd*.

Rozwiązanie

W pliku konfiguracyjnym usługi włączamy opcję rejestrowania:

Plik /etc/xinetd.conf lub /etc/xinetd.d/ftp:
```
service mojausluga
{
    ...
    log_type = SYSLOG mechanizm poziom
    log_on_success = DURATION EXIT HOST PID USERID
    log_on_failure = ATTEMPT HOST USERID

}
```

Domyślnie *xinetd* rejestruje zdarzenia przez usługę *syslog*. Może również rejestrować je do podanego pliku, co wymaga następującej wartości przy słowie kluczowym `log_type`:

```
log_type = FILE nazwa_pliku
```

Omówienie

xinetd może rejestrować komunikaty diagnostyczne przez usługę *syslog* lub bezpośrednio do pliku. Aby korzystać z usługi *syslog*, wybieramy mechanizm (`daemon`, `local0` itp.) i opcjonalnie poziom rejestrowania (`crit`, `warning` itp.). Domyślny poziom to `info`.

```
log_type = SYSLOG daemon            mechanizm = daemon, poziom = info
log_type = SYSLOG daemon warning    mechanizm = daemon, poziom = warning
```

W celu rejestrowania do pliku wystarczy podać jego nazwę:

```
log_type = FILE /var/log/mojausluga.log
```

Opcjonalnie można ustawić stałe i płynne ograniczenia wielkości pliku dziennika; zobacz xinetd.conf(5).

Komunikaty o zdarzeniach mogą być generowane w przypadku gdy uruchomienie i zakończenie usługi powiodło się (słowo `log_on_success`) lub gdy się nie powiodło albo żądanie zostało odrzucone (`log_on_failure`).

Jeśli rejestrowanie nie działa, najbardziej prawdopodobną przyczyną jest niewłaściwa konfiguracja w pliku */etc/syslog.conf*. Łatwo jest popełnić subtelny błąd konfiguracyjny i spowodować, że komunikaty diagnostyczne są kierowane do niewłaściwego miejsca. Żeby sprawdzić, gdzie komunikaty są kierowane, można uruchomić nasz skrypt testujący konfigurację *syslog*. [9.28]

Zobacz również

xinetd.conf(5), syslog.conf(5), inetd.conf(5).

3.19. Uniemożliwianie logowania się przez terminal jako root

Problem

Chcemy uniemożliwić rootowi zalogowanie się bezpośrednio z terminala lub pseudoterminala.

Rozwiązanie

Edytujemy plik */etc/securetty*. Zawiera on nazwy urządzeń — po jednej w każdym wierszu — z użyciem których root może się zalogować. Nie powinny znaleźć się tam pseudoterminale (ang *.pseudo-tty — pty*); inaczej użytkownik root będzie mógł zalogować się przez sieć. Nie powinniśmy tam wpisywać także innych urządzeń, które wydają się problematyczne. We wpisach nie umieszczamy pełnej ścieżki dostępu do urządzeń (*/dev/*). Wiersze rozpoczynające się od znaku krzyżyka (#) są komentarzami, na przykład:

/etc/securetty:
```
# terminale szeregowe
tty1
tty2
# urządzenia devfs
vc/1
vc/2
```

Omówienie

Jeśli to możliwe, w ogóle nie zezwalamy użytkownikowi root na bezpośrednie logowanie do systemu. W przeciwnym razie otwieramy drogę dostępu do systemu: atakujący może wykonać np. atak słownikowy na dany terminal. Użytkownicy powinni logować się ze zwykłymi przywilejami i dopiero potem zdobywać uprawnienia roota (w odpowiedni sposób, który został opisany w rozdziale 5.).

Zobacz również

securetty(5). Dokumentacja systemu plików devfs znajduje się pod adresem *http://www.atnf.csiro.au/people/rgooch/linux/docs/devfs.html*.