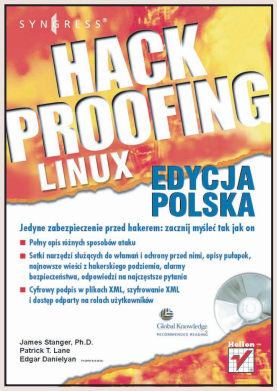
Hack Proofing Linux. Edycja polska



Autorzy: James Stanger Ph.D., Patrick T. Lane, Edgar Danielyan
Tłumaczenie: Rafał Szpoton
ISBN: 83-7361-007-3
Tytuł oryginału: Hack Proofing Linux
Format: B5, stron: 592

Kompletny przewodnik, z którego nauczysz się stosować zabezpieczenia open source.

Książka ta pokazuje doświadczonym administratorom systemów, w jaki sposób używać narzędzi typu open source w trzech kluczowych obszarach: zabezpieczanie serwera, zabezpieczanie sieci komputerowych oraz zabezpieczanie granic sieci komputerowych. Dostarczy Ci praktycznych umiejętności, pozwalających na uszczelnienie sieci komputerowej, zabezpieczenie i monitorowanie systemu operacyjnego oraz sprawdzanie słabych punktów zarówno w sieciach lokalnych, jak i rozległych. Poznasz również sposób utrzymywania i sprawdzania zapory ogniowej oraz rejestrowania zachodzących w niej zdarzeń, pozwalający na utworzenie funkcjonalnej bariery pomiędzy Twoją siecią a światem zewnętrznym.

1. **Powstrzymaj hakera, myśląc dokładnie tak jak on**
   Opanuj czynności potrzebne do włamania się do używanego przez Ciebie systemu operacyjnego oraz przewiduj różnorodne rodzaje ataków.
2. **Zwiększ bezpieczeństwo swojego serwera**
   Dowiedz się, jak zabezpieczyć serwer oparty na Linuksie przy użyciu prostych „ręcznych” poprawek oraz rozwiązań typu open source.
3. **Naucz się używać narzędzi sprawdzających system**
   Pełny opis narzędzi skanujących, w tym programów: AntiVir, Zombie Zapper oraz Nmap.
4. **Podstawy systemów wykrywających włamania (IDS)**
   Dowiedz się o usługach dostarczanych przez systemy IDS, jak również o różnych aplikacjach tego typu i ich charakterystykach.
5. **Przechwyć ruch przesyłany w sieci komputerowej**
   Użyj programów przechwytujących pakiety sieciowe w celu rozwiązania problemów z siecią komputerową oraz potwierdzenia ataków przeprowadzonych przez hakerów.
6. **Zminimalizuj możliwość niewłaściwego wykorzystania narzędzi służących do przechwytywania pakietów**
   Dowiedz się, w jaki sposób wykorzystać rozwiązania używające haseł jednorazowych, system Kerberos v5 oraz szyfrowanie.
7. **Wprowadź bezpieczną autoryzację oraz szyfrowanie na poziomie warstwy sieci**
   Zabezpiecz sieć poprzez użycie Wirtualnych Sieci Prywatnych (VPN).
8. **Ustanów bezpieczną granicę sieci komputerowej**
   Skonfiguruj oraz utrzymuj zaporę sieciową zabezpieczającą Twoją sieć komputerową.
9. **Płyta CD dołączona do książki**
   Dołączona płyta CD dostarcza narzędzia open source oraz kod źródłowy zawarty w książce.

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

Spis treści

Rozdział 7.

Zapobieganie przechwytywaniu pakietów w sieci poprzez szyfrowanie

Kwestie opisywane w niniejszym rozdziale:

- Podstawy szyfrowania w sieci.
- Przechwytywanie i analiza niezaszyfrowanych pakietów w sieci.
- Wykorzystanie OpenSSH do szyfrowania pakietów przesyłanych pomiędzy dwoma serwerami.
- Instalacja OpenSSH.
- Konfiguracja SSH.
- Wykorzystanie SSH do zabezpieczania transmisji danych przez niezabezpieczoną sieć komputerową.
- Przechwytywanie i analiza zaszyfrowanych pakietów w sieci.
- Podsumowanie.
- Szybki przegląd rozwiązań.
- Najczęściej zadawane pytania (FAQ).

Wprowadzenie

W tym momencie rozumiesz już, w jaki sposób możliwe jest rozszerzenie autoryzacji przy użyciu oprogramowania typu open source, dostarczanego przez niezależnych producentów. Masz także świadomość niektórych pułapek związanych z używaniem tego rodzaju oprogramowania w różnych systemach. W poprzednim rozdziale na przykład dowiedziałeś się, w jaki sposób używać autoryzacji z zastosowaniem haseł jednorazowych oraz programu Kerberos. Tego rodzaju implementacja autoryzacji umożliwia systemowi sprawdzenie tożsamości użytkownika do niego zalogowanego oraz integralności danych.

W tym rozdziale poznasz rozwiązania pozwalające na zastosowanie silnego szyfrowania w celu zwiększenia bezpieczeństwa sieci. Szyfrowanie zapewnia poufność danych dzięki użyciu specjalnych algorytmów szyfrujących dane przed wysłaniem ich przez sieć. Serwer otrzymujący dane odszyfrowuje je następnie do postaci pozwalającej na ich odczytanie. Rozwiązania przedstawione w tym rozdziale łączą zarówno techniki autoryzacji, jak i szyfrowania i zawierają kompletny przewodnik, w jaki sposób krok po kroku zaimplementować je w niezabezpieczonej sieci.

Podstawy szyfrowania danych w sieci

Szyfrowanie danych zapewnia niemożliwość ich przeczytania przez osoby niepowołane podczas przesyłania ich przez sieć. Jeśli program przechwytujący pakiety (ang. *sniffer*) przechwyci dane, nie będzie mógł z nich skorzystać, gdyż będą zaszyfrowane. Dlatego też haker nie jest w stanie podejrzeć żadnych nazw użytkowników lub haseł, co powoduje, że informacje przesyłane przy użyciu sieci są bezpieczne. Jedynym wymaganiem jest konieczność obsługiwania przez wszystkie systemy uczestniczące w komunikacji tej samej techniki szyfrowania, na przykład SSH (ang. *secure shell*, czyli bezpieczna powłoka systemowa).

Szyfrowanie używane jest w przypadku każdego transferu danych wymagających zachowania poufności. Ponieważ Internet jest siecią publiczną, zastosowanie szyfrowania jest niezbędne. Transakcje przeprowadzane w e-handlu (ang. *e-commerce*) muszą zapewniać poufność w celu ochrony danych kart kredytowych oraz informacji osobistych. Strony internetowe systemów bankowych, jak również instytucji inwestycyjnych wymagają również przesyłania bardzo poufnych informacji, takich jak chociażby numery kont bankowych, numery identyfikacji podatkowej (NIP). W przypadku gdyby dane te (nazwy użytkowników, hasła, informacje osobiste) dostały się w niepowołane ręce, mogłyby służyć do ataku frontalnego, ponieważ haker mógłby podszywać się pod uprawnionego użytkownika.

Do trzech cieszących się złą sławą, niebezpiecznych protokołów należą `rlogin`, protokół zdalnej powłoki (ang. *rsh — remote shell*) oraz `Telnet`. Do operacji zdalnego logowania oraz transmisji danych jakiegokolwiek typu, nie używają one żadnego szyfrowania. Jeśli na przykład jesteś administratorem systemu i chcesz zalogować się do

innego systemu za pomocą usługi `Telnet`, Twoja nazwa użytkownika oraz hasło przesłane zostaną w postaci jawnego tekstu. W podobny sposób dane pomiędzy dwoma serwerami przesyłane są przez usługi `rsh` oraz `rlogin` (jednak w tym przypadku hasło nie jest wymagane).

Jeśli osoba podsłuchująca pakiety przechwyci te przeznaczone dla administratora systemu, będzie mogła ostatecznie przechwycić pakiety zawierające jego nazwę użytkownika oraz hasło i w ten sposób będzie zdolna dostać się do systemu, podszywając się pod uprawnionego użytkownika.

Przechwytywanie i analiza niezaszyfrowanych pakietów w sieci

W celu przejrzenia niezaszyfrowanej sesji logowania, musisz przechwycić wymieniane podczas niej pakiety. Gdy wykonasz poniższe kroki, będziesz mógł zalogować się do serwera za pomocą usługi `Telnet` i zarejestrować niezabezpieczone pakiety, wykorzystując w tym celu program przechwytujący pakiety o nazwie `Ethereal`, dostępny na zasadach open source.

Zauważ, że w celu poprawnego wykonania poniższego przykładu musisz posiadać dwa systemy: klienta usługi `Telnet` oraz zdalny serwer udostępniający tę usługę. Wszystkie instalacje systemu Linux standardowo posiadają wspomnianą usługę, dlatego też do wykonania przykładu nie są potrzebne żadne dodatkowe programy.

1. Upewnij się, czy program `Ethereal` jest zainstalowany w systemie poprzez wpisanie poniższego polecenia:

   ```
   rpm -qs | grep ethereal
   ```

2. Jeśli nie dostaniesz poprawnej odpowiedzi, musisz pobrać, a następnie zainstalować wspomniany program. `Ethereal` jest zawarty na dołączonej do książki płycie CD (*ethereal-0.8.9-4.i386.rpm*).

3. Po upewnieniu się, że program `Ethereal` jest zainstalowany poprawnie, jesteś gotów do przechwytywania pakietów.

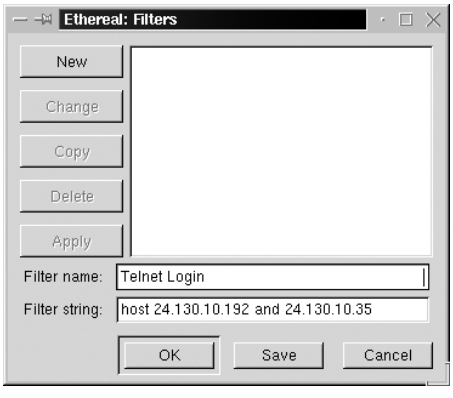
4. W celu dodania do programu `Ethereal` filtrów, bez wykorzystywania nazw serwerów, uruchom interpreter poleceń powłoki i wpisz:

   ```
   ethereal -n
   ```

5. Wybierz menu *Edit*, a następnie pozycję *Filters*. Pojawi się okno wyboru filtra (*Ethereal:Filters*). Ponieważ żadne filtry nie zostały jeszcze skonfigurowane, ekran konfiguracyjny pozostanie pusty.

6. W celu utworzenia filtra pozwalającego na komunikację pomiędzy Twoim serwerem a innym, określonym serwerem docelowym, musisz podać nazwę filtra oraz definiujący go łańcuch znakowy. I tak, aby utworzyć filtr pomiędzy Twoim serwerem a komputerem o adresie 24.130.10.35, podaj nazwę filtra

i łańcuch zaprezentowany na rysunku 7.1. Zauważ jednocześnie, że adres Twojego komputera będzie inny od przedstawionego na rysunku. Następnie będziesz zmuszony do wybrania jeszcze adresu IP Twojego komputera i adresu IP systemu, z którym chcesz się połączyć za pomocą usługi `Telnet`.



Rysunek 7.1. *Tworzenie filtra pomiędzy dwoma serwerami*

7. Po uzupełnieniu obu pól, naciśnij przycisk *Save*, a następnie *New*. W celu opuszczenia ekranu konfiguracji filtra kliknij przycisk *OK*.

8. W celu rozpoczęcia przechwytywania pakietów zaznacz menu *Capture*, a następnie wybierz pozycję *Start*. Pojawi się okno właściwości przechwytywania. Naciśnij przycisk *Filter* i wybierz przed chwilą utworzony filtr o nazwie *Telnet Login*. Po dwukrotnym kliknięciu *OK* rozpocznie się przechwytywanie pakietów.

9. W celu utworzenia pakietów zawierających dane logowania, połącz się z komputerem zdalnym, wykorzystując w tym celu usługę `Telnet`. Jeśli na przykład chciałbyś zalogować się do komputera o adresie 24.130.10.35, napisałbyś:

```
telnet 24.130.10.35
```

10. Aby zalogować się do systemu, podaj nazwę użytkownika oraz hasło. Gdy nie posiadasz powyższych danych, utwórz użytkownika o nazwie `telnet` oraz takim samym haśle za pomocą następującego polecenia:

```
useradd telnet
```

Utwórz hasło dla użytkownika `telnet` przez wpisanie:

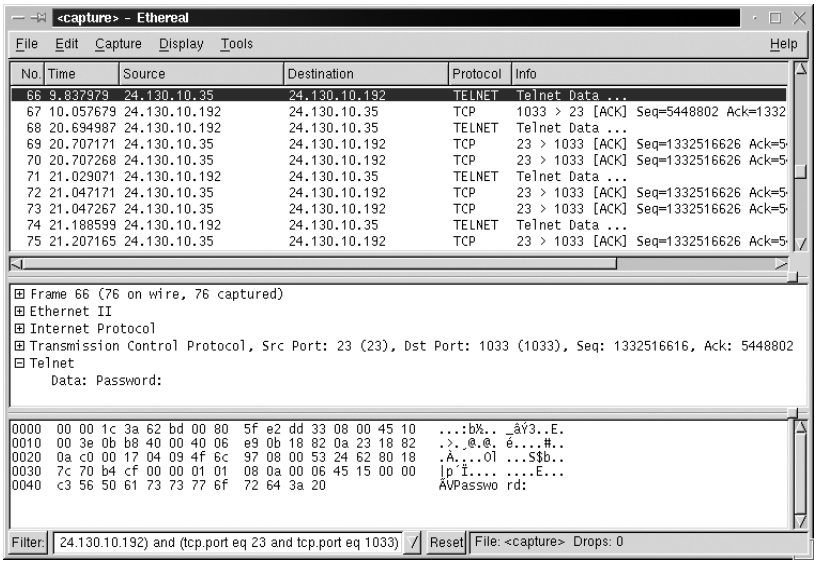
```
passwd telnet
Changing password for user telnet
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
```

11. Po zalogowaniu się do zdalnego komputera jako użytkownik `telnet`, zakończ sesję programu `Telnet`.

12. Zatrzymaj przechwytywanie pakietów w programie `Ethereal` naciśnięciem przycisku *Stop*.

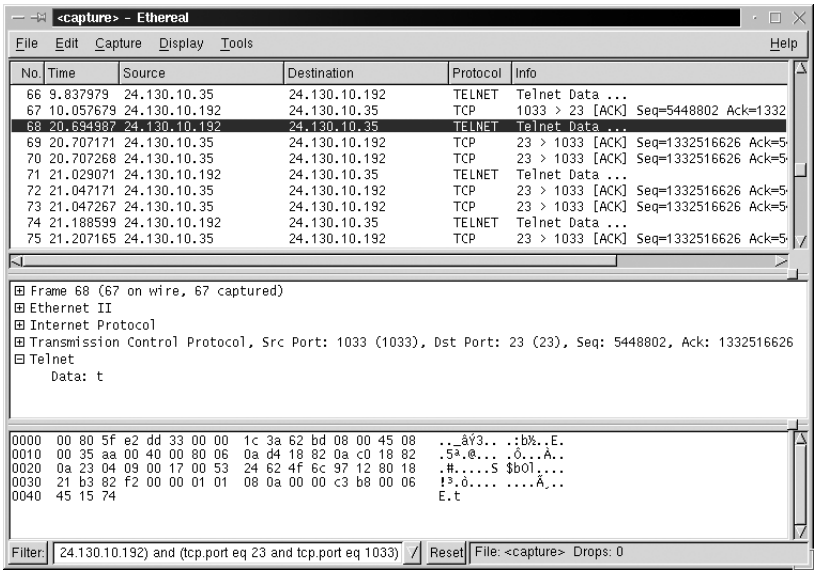
13. W oknie programu `Ethereal` ukażą się przechwycone pakiety. Znajdź pakiet z danymi sesji programu `Telnet` zawierający pola *data:password*. Ekran Twojego komputera będzie przypominać ten przedstawiony na rysunku 7.2, z wyróżnionym pierwszym pakietem zawierającym hasło.

Rysunek 7.2. *Przechwytywanie sesji programu Telnet*



14. Przewiń ekran do drugiego pakietu z hasłem. Pole z danymi zawiera pierwszy znak hasła. W tym przypadku znakiem tym jest litera `t`, jak pokazano to na rysunku 7.3.

Rysunek 7.3. *Określanie pierwszego znaku hasła usługi Telnet*



15. Program `Telnet` przesyła każdy znak hasła jako oddzielny pakiet. Jeśli będziesz dalej przewijać ekran z przechwyconymi pakietami i przeglądać zawarte w nich dane, poznasz całe hasło.

16. Prostszym sposobem na poznanie hasła jest śledzenie całego strumienia TCP. W tym celu zaznacz dowolny pakiet związany z połączeniem za pomocą programu `Telnet`. Może to być dowolny pakiet TCP lub programu `Telnet`, pod warunkiem, że należy on do sesji.

17. Po wybraniu odpowiedniego pakietu wybierz menu *Tools*, a następnie pozycję *Follow TCP Stream*. Pojawi się zawartość strumienia TCP, w sposób pokazany na rysunku 7.4.

Rysunek 7.4. *Odnajdywanie nazwy użytkownika i hasła za pomocą śledzenia strumienia TCP*

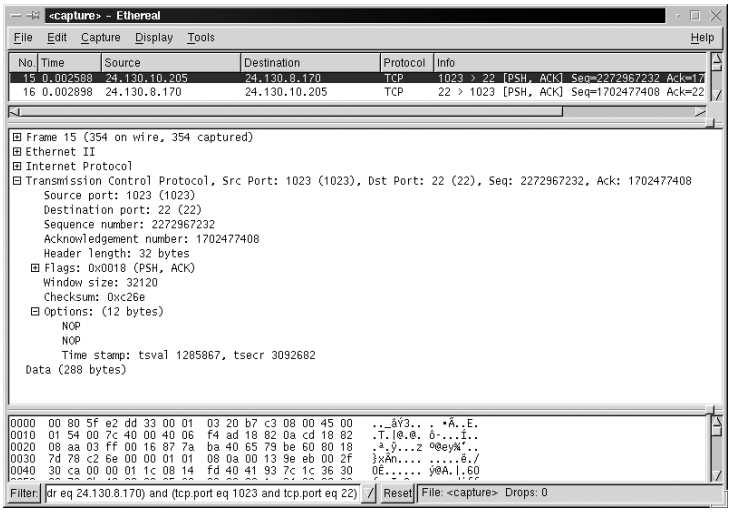
18. Nazwa użytkownika oraz hasło są wyświetlone w postaci jawnego tekstu. Zauważ, że znaki w nazwie użytkownika są powtórzone. Domyślnie, system odróżnia znaki wprowadzone przez użytkownika (zaznaczone na brązowo) od tekstu systemowego (kolor niebieski). Nawet jeśli na tej wydrukowanej stronie nie możesz zobaczyć zaznaczonego na brązowo tekstu, przesłanego przez klienta, możesz dostrzec, że program przechwytujący pakiety odkrył nazwę użytkownika oraz jego hasło.

19. Zapisz przechwycony pakiet jako *unsectel*, a następnie zakończ działanie programu `Ethereal`.

Jak widzisz, możliwe jest przechwycenie sesji logowania i ujawnienie nazwy użytkownika oraz jego hasła. W tym wypadku haker mógłby teraz zalogować się do zdalnego serwera z tymi samymi uprawnieniami, które posiada użytkownik o nazwie `telnet`. Jest on uprawnionym użytkownikiem, nawet w przypadku gdy nie jest nim osoba podszywająca się. Zdalny komputer stałby się ofiarą ataku frontalnego.

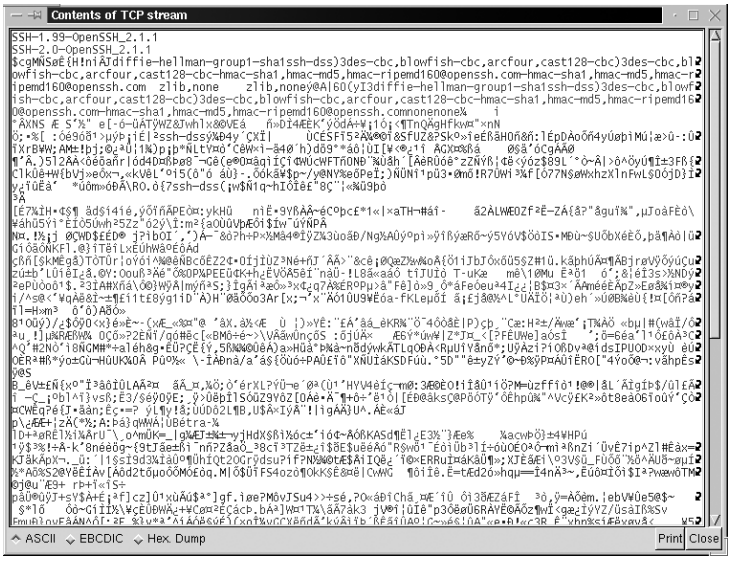
Gdy w sieci zaimplementowane jest szyfrowanie, również sesja logowania byłaby zaszyfrowana. W takim przypadku wszystkie dane przekazywane pomiędzy dwoma komputerami stałyby się nieprzydatne, gdyż dane aplikacji (takie jak pakiet programu `Telnet`) byłyby zaszyfrowane, tak jak zostało to pokazane na rysunku 7.5.

Rysunek 7.5. *Zaszyfrowana sesja logowania*



Gdy haker w celu poznania nazwy użytkownika i hasła podejmie próby odczytania strumienia TCP, uzyska również bezużyteczne informacje, pokazane na rysunku 7.6.

Rysunek 7.6. *Śledzenie strumienia TCP z włączonym szyfrowaniem sieci*



Hakerzy nie przechwytują całego ruchu w sieci w nadziei, że znajdą oni ważne informacje. W Internecie panuje po prostu zbyt duży ruch, tak więc technikę tę można by przyrównać do przysłowiowego szukania igły w stogu siana. Zamiast tego, koncentrują się oni raczej na jednym serwerze lub interfejsie użytkownika lub próbują włamać się do serwera zawierającego ważne, poszukiwane przez nich informacje. Na przykład mogą próbować włamać się do serwera przechowującego bankową bazę danych zawierającą numery kart kredytowych klientów lub informacje osobiste. W opisanych w tym fragmencie krokach został utworzony filtr pakietów koncentrujący się na serwerze, z którym nawiązywane jest połączenie za pomocą programu `Telnet`.

Uszkodzenia i Obrona...

Zabezpieczanie transakcji w handlu elektronicznym

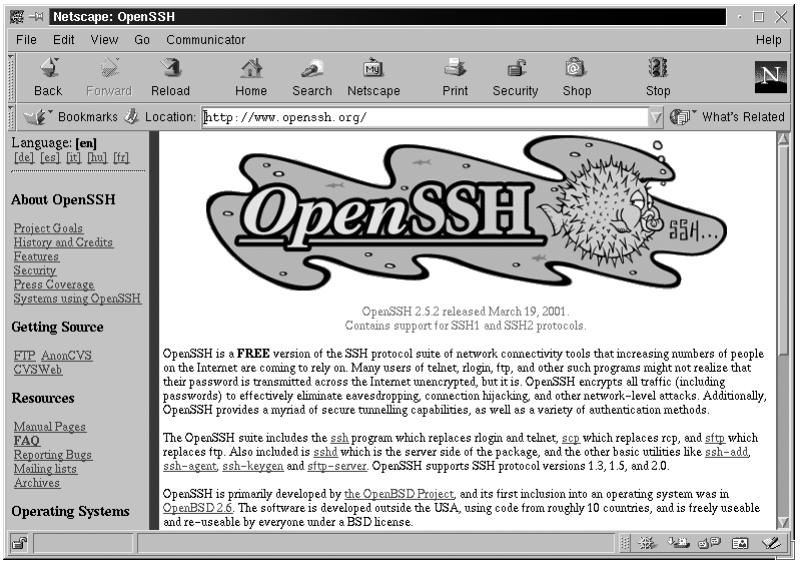
Gdyby hakerzy odkryli niezabezpieczony serwer, mogliby w celu uzyskania poszukiwanych danych przechwytywać wchodzące do serwera i wychodzące z niego pakiety. Jeśli na przykład serwer uczestniczący w handlu elektronicznym nie używa podczas transakcji szyfrowania, istnieje całkiem duży zasób potencjalnych danych, możliwych do zgromadzenia przez hakera. Niestety, wiele małych firm i przedsiębiorstw uruchamia własne serwery internetowe, bez świadomości możliwych problemów związanych z bezpieczeństwem, a do przetwarzania formularzy z danymi o płatnościach używa prostych skryptów. Chociaż taka transakcja dochodzi do skutku, przebiega ona w sposób niezabezpieczony. Każdy pakiet może zawierać cenne informacje, które mogą być w prosty sposób zaobserwowane w sieci. Do przeprowadzania bezpiecznych transakcji w e-handlu nadaje się idealnie technologia SSL (ang. *Secure Sockets Layer*, czyli bezpieczna warstwa gniazd).

Gdy haker przechwyci numer karty kredytowej, może go wykorzystać lub sprzedać. Kiedy klient dokonujący zakupów przez Internet zacznie określać, gdzie jego numer został skradziony, może zdać sobie sprawę, że nastąpiło to zaraz po dokonaniu transakcji internetowej. Może spowodować to zszarganie reputacji witryny internetowej, ponieważ jej właściciel straci przynajmniej jednego klienta, a w ostateczności może doprowadzić również do jego bankructwa. Jest to szczególnie kłopotliwe, gdy serwer sieciowy znajduje się pod obserwacją hakera.

W następnym podrozdziale przedstawiony zostanie przykład demonstrujący, w jaki sposób zaimplementować ważny system szyfrowania sieci: OpenSSH. System ten pozwoli na zapewnienie bezpieczeństwa wszystkich danych transmitowanych pomiędzy serwerami i uczyni je nieprzydatnymi dla hakerów.

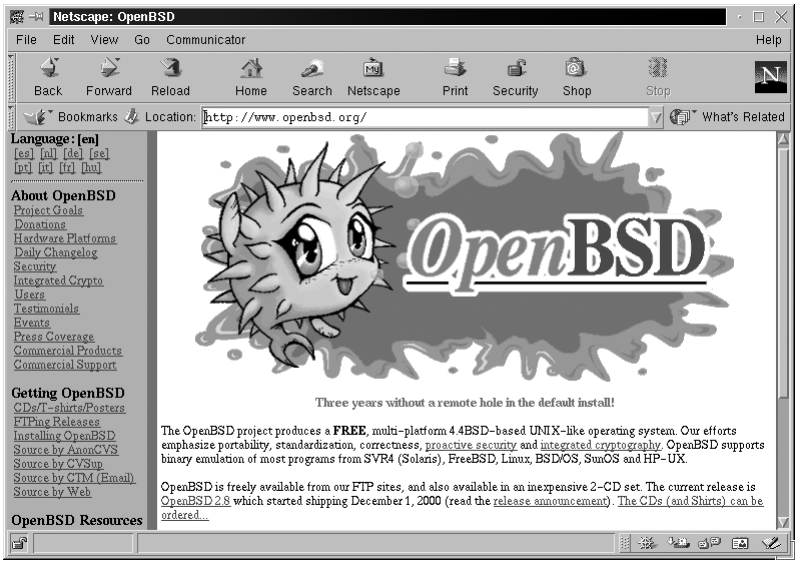
Wykorzystanie OpenSSH do szyfrowania pakietów przesyłanych pomiędzy dwoma serwerami

OpenSSH (*www.openssh.org*) jest programem typu open source, szyfrującym wszystkie dane przesyłane pomiędzy komputerami używającymi bezpiecznej powłoki (SSH). Jest bezpiecznym programem zastępującym powszechne programy internetowe, służące do nawiązywania zdalnego połączenia z komputerem, takie jak `Telnet`, `rlogin` oraz `rsh`. Ponieważ szyfruje on wszystkie dane, powoduje również ukrycie nazw użytkowników i haseł używanych podczas zdalnego połączenia. Po jego nawiązaniu i zalogowaniu, szyfruje również wszystkie przesyłane dane. Open SSH jest darmową wersją pakietu SSH firmy SSH Communications Security Corporation (*www.ssh.org*). Jak w przypadku większości tego rodzaju oprogramowania open source, ceną jest brak dostępnego wsparcia producenta. Nie myl OpenSSH z płatnym pakietem SSH. Strona domowa OpenSSH pokazana jest na rysunku 7.7.



Rysunek 7.7. *Strona domowa programu OpenSSH*

Program OpenSSH rozwinięty został w ramach projektu OpenBSD (*www.openbsd.org*), wraz z wersją systemu operacyjnego Unix o nazwie OpenBSD. Jest to bezpłatny system operacyjny oparty na systemie 4.4BSD, który został zaprojektowany z myślą o bezpieczeństwie. Wykorzystuje on silne techniki szyfrowania w celu odpierania ataków hakerów. Specjaliści zaangażowani w prace nad projektem OpenBSD twierdzą, że w domyślnej instalacji systemu nie wykryto żadnych luk w bezpieczeństwie od ponad trzech lat. W ramach projektu OpenBSD program OpenSSH został również przeniesiony na inne systemy operacyjne: Linux, HP-UX, AIX, Irix, SCO, MacOS X, Cygwin, Digital Unix/ Tru64/OSF, SNI/Reliant Unix, NeXT oraz Solaris. Strona domowa projektu OpenBSD przedstawiona została na rysunku 7.8.



Rysunek 7.8. *Strona domowa projektu OpenBSD*

Pakiet OpenSSH

OpenSSH stanowi zestaw bezpiecznych sieciowych programów komunikacyjnych i zawiera następujące składniki:

- **Klient SSH programu OpenSSH (SSH)**. Program używany do zdalnego logowania do systemu, wykorzystywany do przeprowadzania bezpiecznych operacji logowania oraz szyfrowania sesji. Stanowi on bezpieczną alternatywę dla programów `rlogin` oraz `Telnet`.
- **Program bezpiecznego kopiowania (SCP)**. Program służący do bezpiecznego kopiowania plików pomiędzy komputerami umieszczonymi w sieci. Obsługuje on również nazwy użytkowników oraz hasła.
- **Program bezpiecznego przesyłania plików (SFTP)**. Wykorzystywany do bezpiecznego interakcyjnego przesyłania plików. Stanowi bezpieczną alternatywę dla usługi `FTP`.
- **Demon SSH programu OpenSSH (SSHD)**. Jest demonem SSH.

W pakiecie OpenSSH zawartych jest wiele funkcji zapewniających bezpieczną transmisję z wykorzystaniem sieci oraz rozszerzających użyteczność programu. Tabela 7.1 zawiera listę cech pakietu OpenSSH.

Pakiet OpenSSH 2.0 obsługuje protokoły SSH 1.3, 1.5 oraz 2.0. Ważną informację stanowi fakt, że SSH 2.0 nie używa algorytmów RSA (ang. *Rivest, Shamir, Adleman*), lecz je obsługuje. Ponieważ algorytmy RSA są wciąż chronione patentami, twórcy SSH 2.0 zdecydowali się na użycie w zamian algorytmów DH (ang. *Diffie-Hellman*) oraz DSA (ang. *Digital Signature Algorithm*, czyli algorytm podpisu cyfrowego). Dlatego też w przypadku, gdy używany przez Ciebie klient SSH oraz serwery wykorzystują protokoły SSH 1.3, 1.5 lub 2.0, będą ze sobą zgodne. Większość komercyjnych implementacji na systemy Unix oraz Windows (oraz inne) używa właśnie tych wersji.

Instalacja OpenSSH

Implementacje OpenSSH różnią się znacząco pomiędzy systemami, ze względu na autoryzację. W rzeczywistości twórcy pakietu podzielili jego rozwój na dwie kategorie:

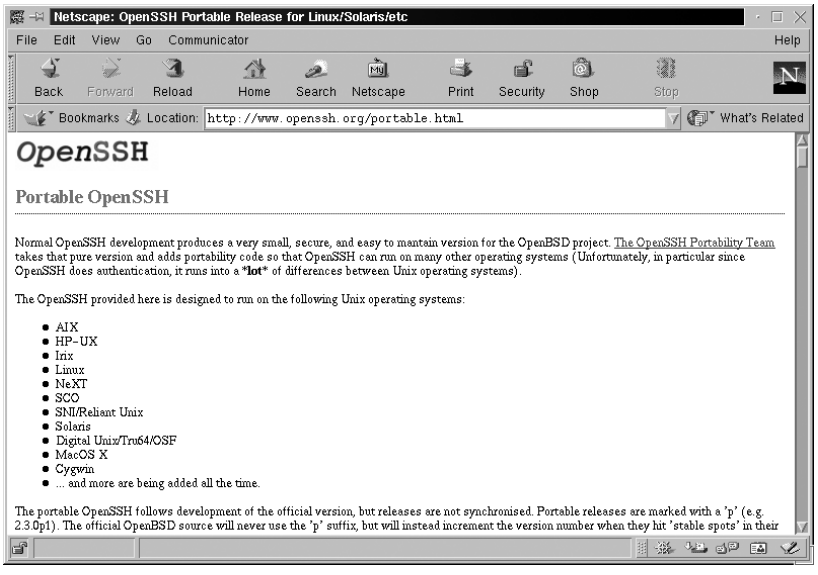
- **Rozwój narzędzi opartych na OpenBSD**. W ramach tej kategorii tworzony jest bezpieczny, czysty i prosty kod OpenSSH dla systemu operacyjnego OpenBSD.
- **Zespół zajmujący się przenośnością OpenSSH**. Wykorzystuje kod OpenSSH z systemu OpenBSD w celu tworzenia przenośnych wersji dla innych systemów operacyjnych. Każda wersja przenośna oznaczona jest symbolem „p” w celu odróżnienia jej od wersji OpenBSD. Wersje przenośne nie są publikowane w tym samym czasie co wersje OpenBSD. Opracowanie tych wersji zajmuje zazwyczaj więcej czasu, ponieważ jest on potrzebny do utworzenia dodatkowego kodu.

Tabela 7.1. *Cechy pakietu OpenSSH*

Cecha	Opis
Silne szyfrowanie danych za pomocą standardu potrójnego szyfrowania danych (3DES) oraz algorytmu Blowfish.	Algorytmy szyfrowania 3DES oraz Blowfish są bezpłatnie dostępne we wszystkich krajach. 3DES jest lepiej sprawdzony, zaś Blowfish udostępnia szybsze szyfrowanie, dzięki wykorzystaniu szybkich szyfrów blokowych. Może zostać użyty dowolny z tych algorytmów szyfrowania. Algorytmy te stosowane są jeszcze przed autoryzacją, w celu zapewnienia szyfrowania wszystkich nazw użytkowników i haseł, jak również danych sesji.
Silna autoryzacja za pomocą kluczy publicznych, haseł jednorazowych oraz autoryzacji Kerberos.	Zabezpiecza słabe punkty autoryzacji, takie jak atak za pomocą podszywania się (ang. *spoofing*) pod adres IP lub DNS (ang. *Domain Name System*, czyli system nazw domeny) oraz możliwość fałszywego trasowania pakietów. Wraz z OpenSSH używane są cztery rodzaje autoryzacji: ♦ autoryzacja jedynie za pomocą klucza publicznego, ♦ autoryzacja serwera oparta na kluczu publicznym wraz z plikiem *.rhosts*, ♦ autoryzacja za pomocą haseł jednorazowych, ♦ autoryzacja Kerberos.
Przesyłanie zaszyfrowanych danych X Window.	Szyfruje wszystkie dane przesyłane pomiędzy zdalnymi systemami wykorzystującymi usługę X Window. Chroni przed podsłuchiwaniem danych przekazywanych przez `xterm` oraz przejmowaniem jego sesji.
Zaszyfrowanie portów.	Pozwala na przekazywanie do innego systemu zaszyfrowanych portów TCP/IP. Jest to idealne rozwiązanie dla protokołów internetowych, które nie wspierają wewnętrznie szyfrowania, takich jak POP lub SMTP.
Przekazywanie agenta dla pojedynczych logowań w sieci.	Klucze autoryzujące użytkownika mogą być przechowywane na jego lokalnym komputerze, który staje się agentem autoryzującym. Gdy użytkownik korzysta z sieci z innego systemu, połączenie jest przekazywane do tego agenta autoryzacji. Zapobiega to instalacji kluczy autoryzacji na każdym serwerze sieciowym i pozwala na dostęp użytkownika do sieci z dowolnego systemu.
Współdziałanie.	Jest zgodny z wieloma wersjami protokołu SSH (SSH 1.3, 1.5 oraz 2.0).
Kompresja danych.	Pozwala na kompresję danych przed zaszyfrowaniem. Jest to bardzo ważne w przypadku sieci z wolnymi połączeniami.
Przekazuje do zdalnego systemu identyfikatory Kerberos oraz AFS (ang. *Andrew File System*).	Pozwala użytkownikom na dostęp do usług AFS oraz Kerberos za pomocą jednorazowo wprowadzanego hasła.

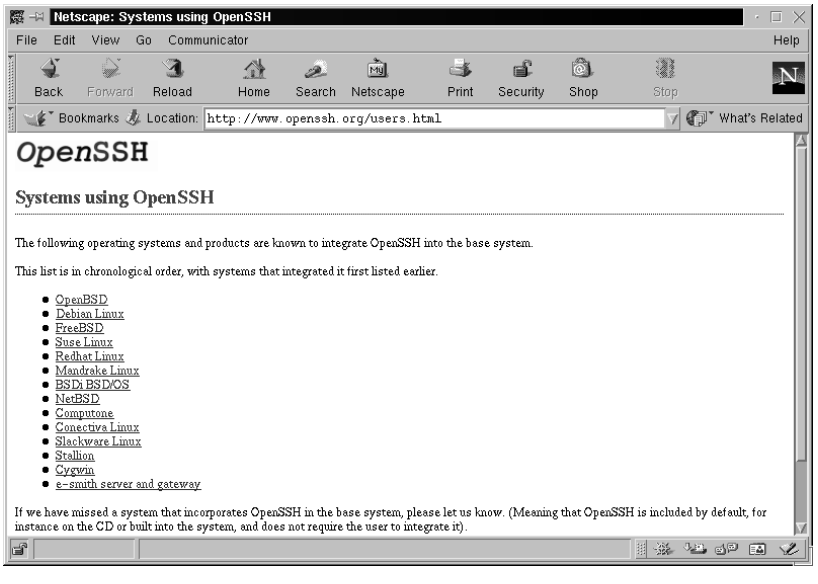
Upewnij się, czy pobrałeś odpowiednią wersję dla używanego przez Ciebie systemu operacyjnego. W celu określenia czy system ten jest obsługiwany, odwiedź stronę domową przenośnego OpenSSH, umieszczoną pod adresem *www.openssh.org/portable.html*. Łącza wskazujące na każdą z wersji umieszczone są na dole strony. Przewiń ją w celu odnalezienia serwera znajdującego się najbliżej Twego miejsca zamieszkania i wskaż na określony system operacyjny. Pliki instalacyjne OpenSSH wymagane dla systemu Linux zawarte są na płycie CD dołączonej do tej książki. Wspomniana strona domowa OpenSSH pokazana została na rysunku 7.9.

Rysunek 7.9. *Systemy operacyjne obsługiwane przez przenośną wersję OpenSSH*



OpenSSH staje się coraz popularniejszy i wiele systemów operacyjnych jest publikowanych z domyślnie zainstalowaną wersją tego programu. I tak, na przykład Red Hat Linux 7 instaluje OpenSSH 2.1.1p7-1 podczas procesu instalacji. Inne systemy operacyjne zawierające OpenSSH w podstawowej wersji pokazane zostały na rysunku 7.10. Należą do nich między innymi Red Hat, SuSE, Mandrake Linux, FreeBSD.

Rysunek 7.10. *Systemy operacyjne zawierające OpenSSH*



Na podstawie powyższych informacji jesteś w stanie określić, czy potrzebujesz pobierać i instalować program OpenSSH. Przykłady w tym rozdziale wykorzystują Red Hat Linux 7. Dla tego systemu również została już utworzona przenośna wersja OpenSSH. Jest ona rozprowadzana w postaci archiwum programu RPM (ang. *Red Hat Package*

Manager, czyli zarządca pakietów programu Red Hat), dołączonego do podstawowego pakietu instalacyjnego programu Red Hat Linux 7. Wspomniane pliki instalacyjne umieszczone są również na płycie CD dołączonej do tej książki. Dlatego też zaraz po upewnieniu się, że OpenSSH jest zainstalowany, jesteśmy gotowi do rozpoczęcia jego konfiguracji.

1. By upewnić się, czy w systemie zainstalowana jest wersja RPM pakietu OpenSSH, wpisz poniższe polecenie:

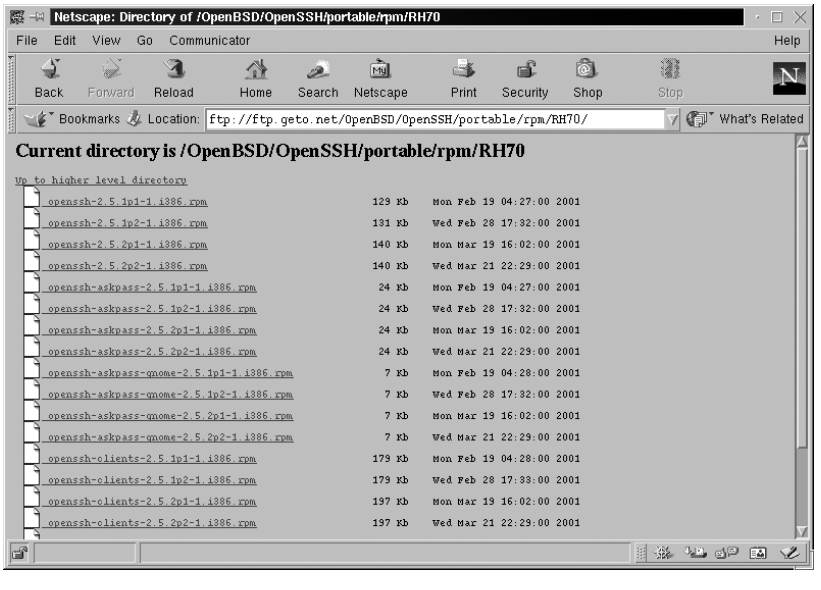
   ```
   rpm -qa | grep ssh
   ```

2. W przypadku zainstalowania SSH powinieneś otrzymać następującą odpowiedź:

   ```
   openssh-askpass-gnome-2.1.1p4-1
   openssh-clients-2.1.1p4-1
   openssh-2.1.1p4-1
   openssh-server-2.1.1p4-1
   openssh-askpass-2.1.1p4-1
   ```

   Gdy otrzymasz tego rodzaju odpowiedź, jesteś gotów do konfiguracji OpenSSH.

3. Jeśli nie otrzymałeś odpowiedzi, zainstaluj pliki z dołączonej płyty CD lub użyj strony *www.openssh.org/portable.html* w celu zlokalizowania najbliższego serwera i pobrania przenośnej wersji, zgodnej z używanym przez Ciebie systemem operacyjnym. W moim przypadku, ponieważ mieszkam w Los Angeles, wybrałbym pozycję *Santa Barbara*, *CA*, *USA*, a następnie łącze *Linux RPMs*. Na koniec wskazałbym katalog *RH70/* (lub analogiczny) i pobrał odpowiadające pliki RPM wymienione w kroku 2. Numery wersji tych plików będą wyższe niż plików dołączonych do instalacji Red Hat Linux 7, tak jak to pokazano na rysunku 7.11.



Rysunek 7.11. *Pobieranie archiwum RPM z programem OpenSSH dla systemu Red Hat Linux 7*

4. W przypadku każdego pliku RPM będzie istnieć kilka wersji. Dla każdego z nich pobierz najnowszą.
5. Zainstaluj archiwa RPM za pomocą polecenia `rpm -i`. Zaraz po zainstalowaniu będziesz gotów do konfiguracji programu OpenSSH.

Konfiguracja SSH

`SSH` jest klientem SSH programu OpenSSH i współpracuje z programem SSHD (demon SSH). Razem zastępują one programy `rlogin` oraz `rsh`. SSH może również zastępować program `Telnet`. Program ten jest używany do przeprowadzenia operacji logowania na zdalnym systemie i wykonywania na nim poleceń. Jedyną różnicą pomiędzy programami `Telnet`, `rlogin`, `rsh` oraz SSH jest fakt, iż ten ostatni jest bezpieczny.

Na początku tego rozdziału użyłeś programu `Telnet` w celu nawiązania połączenia ze zdalnym serwerem. Cała sesja była niezabezpieczona, ponieważ wszystkie dane przesłane zostały w postaci jawnego tekstu, nawet nazwa użytkownika i jego hasło. Za pomocą programu o nazwie `Ethereal` przechwytującego pakiety, byłeś w stanie zarejestrować pakiety sesji, jak również śledzić strumień TCP (ang. *Transmission Control Protocol*, czyli Protokół Sterowania Transmisją). Poznałeś w ten sposób nazwę użytkownika i hasło wykorzystywane do nawiązania połączenia. Ponieważ posiadasz te dane oraz (w tym przypadku) adres IP (ang. *Internet Protocol*, czyli protokół internetowy) zdalnego komputera, możesz zalogować się jako użytkownik, którego pakiety zostały przechwycone.

Wykorzystując podobną metodę, w celu określenia wymaganych danych autoryzacji, możesz również przechwycić sesje programów `rlogin` oraz `rsh`. Na przykład, ponieważ zarówno `rlogin`, jak i `rsh` używają autoryzacji serwera, możesz określić adres IP lub pełną kwalifikowaną nazwę domeny oraz nazwę użytkownika wymagane do zalogowania do serwera. Zaraz po określeniu nazwy komputera oraz nazwy użytkownika wykorzystywanych do autoryzacji, mogą być one użyte do podszywania się pod adres IP oraz DNS (ang. *IP and DNS spoofing*).

Klient SSH zastępuje programy `Telnet`, `rlogin` oraz `rsh`. Umożliwia utworzenie bezpiecznego kanału danych pomiędzy dwoma serwerami w sieci. Komputery te mogą nie być zaufane, zaś sieć może być niezabezpieczona. Aby usługa działała poprawnie, oba serwery muszą obsługiwać SSH. Jeden z serwerów łączy się z drugim przy użyciu połączenia zaszyfrowanego. Z tego powodu każdy haker, który przechwyci dane będzie potrzebować bardzo dużo czasu na ich odszyfrowanie.

W jaki sposób działa SSH

Implementacja SSH łączy w sobie techniki podobne do zdalnych poleceń (ang. *r-command*) wraz z metodami klucza publicznego i prywatnego. By zrozumieć jak działa SSH, warto wcześniej poznać sposób działania starszych zdalnych poleceń.

Niezabezpieczona autoryzacja zdalnych poleceń

W przypadku zdalnych poleceń, takich jak `rlogin`, `rsh` lub `rcp`, w systemie Red Hat Linux 7 używana jest następująca metoda autoryzacji. Jako przykład zostanie omówione polecenie `rlogin`. Każdy użytkownik logujący się do zdalnego systemu musi posiadać w nim swoje konto użytkownika. W tym przykładzie wykorzystamy konto o nazwie `susan`. Jeśli jest ona zalogowana lokalnie podczas nawiązywania połączenia ze zdalnym komputerem, do uzyskania dostępu nie jest wymagane żadne hasło. Dzieje się tak dlatego, że jej konto jest autoryzowane dzięki wpisowi w pliku *.rhosts* umieszczonym w katalogu *$HOME/susan* znajdującym się na zdalnym serwerze. Plik *.rhosts* musi zostać utworzony w katalogu domowym *Susan*.

Plik *.rhosts* zawiera nazwę serwera oraz nazwę użytkownika wymagane podczas logowania `Susan` do systemu. Nazwa komputera `Susan` brzmi *we-24-130-10-205.we.mediaone.net*, zaś jej nazwa użytkownika to `susan`. W przypadku gdy polecenie `rlogin` dopasuje wpis w pliku *.rhosts*, uzyska ona dostęp do systemu. Nie jest wymagane żadne hasło. Plik *.rhosts* ma następujący format:

```
nazwa_serwera nazwa_użytkownika
```

Nazwa serwera powinna być podana w postaci FQDN (ang. *Full Qualified Domain Name*, czyli w pełni kwalifikowana nazwa domeny), a nie skróconej. Na przykład zamiast:

```
we-24-130-10-205
```

użyj:

```
we-24-130-10-205.we.mediaone.net
```

Nazwa użytkownika musi być zgodna z nazwą konta systemowego. Gdy dla danej nazwy użytkownika istnieje odpowiednie konto, użytkownik ma dostęp do wszystkich kont z wyjątkiem konta administratora `root`.

Jak zostało wspomniane wcześniej, aby Susan mogła zalogować się zdalnie w systemie, administrator musi wcześniej utworzyć w katalogu *$HOME/susan* odpowiedni plik *.rhosts*, zawierający nazwę serwera oraz nazwę użytkownika. Jeśli na przykład komputer używany przez `Susan` posiada nazwę *we-24-130-10-205.we.mediaone.net*, administrator wprowadzi następujący wiersz:

```
we-24-130-10-205.we.mediaone.net susan
```

W momencie kiedy Susan będzie gotowa do zalogowania się do zdalnego serwera, użyje następującego polecenia `rlogin`:

```
rlogin -l susan we-24-130-8-170.we.mediaone.net
```

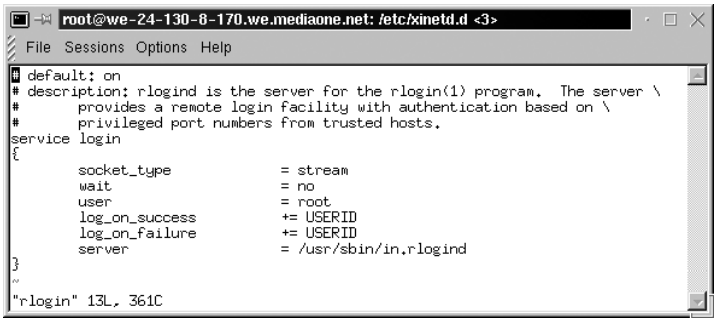
gdzie `we-24-130-8-170.we.mediaone.net` jest nazwą zdalnego serwera. Parametr `-l` wskazuje nazwę konta użytkownika używanego do logowania, którym w tym przypadku jest `susan`.

Tego typu metoda nie jest bezpieczna, jako że w celu autoryzacji nazwa serwera oraz nazwa użytkownika przesyłane są do zdalnego komputera otwartym tekstem. Umożliwia to podszywanie się pod adres IP, DNS zdalnego komputera oraz modyfikację wyznaczania trasy przesyłania pakietów.

Z powodu tego rodzaju podatności na atak, zaleca się wyłączenie narzędzi służących do zdalnego wykonywania poleceń. Taki proces, polegający na wykorzystaniu katalogu */etc/xinetd.d* i wykomentowaniu odpowiednich plików tekstowych został opisany w rozdziale 2. Katalog */etc/xinetd.d* pozwala na szybkie wyłączenie usług, które nie są wykorzystywane przez system. Możesz na przykład wyłączyć usługi `rlogin` oraz `rsh`, poprzez wykomentowanie wpisów `rlogin` oraz `rsh` w odpowiednich plikach, a następnie ponowne uruchomienie usługi. Gdy usługa jest wykomentowana, nie zostanie uruchomiona. Natychmiast po wyłączeniu usługi, do serwera nie będzie mieć dostępu żaden użytkownik używający zdalnych poleceń.

1. W celu wyłączenia `rlogin`, musisz zmodyfikować plik */etc/xinetd.d/rlogin*. W tym celu otwórz wspomniany plik za pomocą edytora `vi` lub innego, tak jak zostało to pokazane na rysunku 7.12.

Rysunek 7.12. *Wyłączenie usługi rlogin za pomocą pliku /xinetd.d/rlogin*



2. Wykomentuj wiersz `service login` — dodaj przed nim znak krzyżyka (#):

```
#service login
```

3. Zapisz zmiany i zamknij plik.

4. Następnie musisz ponownie uruchomić `xinetd` przez wpisanie:

```
/etc/rc.d/init.d/xinetd restart
Stopping xinetd:          [ OK. ]
Starting xinetd:          [ OK. ]
```

5. Przy użyciu tej samej metody (poprzez modyfikację plików */etc/xinetd.d/rsh* oraz */etc/xinetd.d/rexec* i wykomentowanie wierszy `service shell`), wyłącz usługę `rsh`.

Usługa `rexec` jest innym narzędziem do zdalnego wykonywania poleceń, udostępniającym w tym celu autoryzację na podstawie nazw użytkowników i hasła. Domyślnie jest ona wyłączona.

6. W celu zapewnienia dodatkowego bezpieczeństwa, przy użyciu tej samej metody (tj. poprzez edycję pliku */xinetd.d/telnet* i wykomentowanie wiersza `service telnet`) wyłącz usługę `Telnet`.

7. Uruchom ponownie `xinetd`.

Wyłączyłeś programy zdalnych klientów wysyłających informacje bez szyfrowania. Ponieważ programy te są podatne na ataki, powinny być całkowicie zastąpione SSH. Następny podpunkt prezentuje w jaki sposób SSH zastępuje wspomniane programy.

Bezpieczna autoryzacja SSH

SSH oparty jest na kryptografii za pomocą klucza publicznego. Wersje wcześniejsze tego programu (przed SSH 2.0) wykorzystywały autoryzację opartą na RSA. Natomiast wersja 2.0 oraz wszystkie późniejsze używają w zamian nieobjętego patentami DSA. Kryptografia oparta na kluczu publicznym używa do zapewnienia autoryzacji kluczy publicznych i prywatnych. Klucz prywatny jest znany jedynie użytkownikowi, zaś klucz publiczny jest dostępny dla każdego, jak chociażby dla zdalnego komputera.

SSH może utworzyć dla użytkownika parę kluczy DSA (publiczny i prywatny) za pomocą polecenia `ssh-keygen -d`. W SSH 2.0 prywatny klucz DSA przechowywany jest w pliku *$HOME/.ssh/id_dsa*. Klucz publiczny natomiast umieszczony jest w *$HOME/.ssh/id_ dsa.pub*. Ten ostatni klucz powinien być przekopiowany do zdalnego komputera pod nazwą *$HOME/.ssh/authorized_keys2*. Plik o nazwie *authorized_keys2* zawiera w każdym wierszu jeden klucz publiczny.

Plik *$HOME/.ssh/authorized_keys2* umieszczony na zdalnym serwerze jest używanym w SSH odpowiednikiem pliku *$HOME/.rhosts*, wykorzystywanego przez narzędzia pozwalające na zdalne wykonywanie poleceń. We wcześniejszej części tego rozdziału został przedstawiony sposób jego konfiguracji w celu umożliwienia zdalnego logowania użytkownika za pomocą polecenia `rlogin`.

W wersji 1 programu SSH użyta została autoryzacja RSA. W dowolnej wersji OpenSSH klucze publiczne i prywatne RSA tworzone są za pomocą polecenia `ssh-keygen` użytego bez opcji `-d`. Klucz prywatny przechowywany jest w pliku *$HOME/.ssh/identity*, zaś klucz publiczny w pliku *$HOME/.ssh/identity.pub*, które umieszczone są w katalogu domowym użytkownika. Klucz publiczny powinien być przekopiowany pod zmienioną nazwą do pliku *$HOME/.ssh/authorized_keys* umieszczonego na zdalnym komputerze.

SSH umożliwia również autoryzację za pomocą hasła, w przypadku gdy autoryzacja oparta na kluczu publicznym nie powiedzie się. Taka autoryzacja możliwa jest również wówczas, gdy klucz publiczny nie jest dostępny. Elastyczność tej metody pozwala na zaszyfrowanie hasła i przekazanie go za pomocą sieci, nawet gdy nie działa autoryzacja za pomocą klucza publicznego. Zapewnia to więc zachowanie integralności danych. W tabeli 7.2 podsumowane zostało położenie kluczy publicznych i prywatnych używanych w SSH.

Inne ważne pliki wykorzystywane do identyfikacji kluczy publicznych w systemie wymienione są w tabeli 7.3.

Tabela 7.2. *Położenie kluczy publicznych używanych w SSH*

SSH wersja 2	Domyślne położenie w systemie lokalnym	Położenie w systemie zdalnym
Klucz prywatny	*$HOME/.ssh/id_dsa*	Nie dotyczy
Klucz publiczny	*$HOME/.ssh/id_dsa.pub*	*$HOME/.ssh/authorized_keys2*
SSH wersja 1	**Domyślne położenie w systemie lokalnym**	**Położenie w systemie zdalnym**
Klucz prywatny	*$HOME/.ssh/identity*	Nie dotyczy
Klucz publiczny	*$HOME/.ssh/identity.pub*	*$HOME/.ssh/authorized_keys*

Tabela 7.3. *Dodatkowe pliki używane w SSH*

Plik SSH	Opis
$HOME/.ssh/known_hosts	Zawiera klucze publiczne dla wszystkich komputerów, do których zalogował się użytkownik. Umieszczone są w nim klucze publiczne, które nie są wymienione w pliku */etc/ssh_known_hosts*.
/etc/ssh_known_hosts	Zawiera wszystkie klucze publiczne RSA dla wszystkich komputerów znanych w systemie. Na przykład, każdy komputer mogący połączyć się z danym systemem powinien posiadać swój klucz publiczny wymieniony w tym pliku. Administrator sieci powinien skonfigurować ten plik umieszczając w nim wszystkie klucze publiczne użytkowników firmy. Plik ten powinien być ogólnie dostępny do odczytu i zawierać jeden klucz publiczny w wierszu.
/etc/ssh_known_hosts2	Plik ten jest identyczny z */etc/ssh_known_hosts*, z wyjątkiem tego, że zawiera klucze publiczne DSA dla wszystkich komputerów znanych systemowi.
$HOME/.ssh/config	Plik konfiguracyjny właściwy dla każdego użytkownika. Każdy użytkownik może posiadać określony specyficzny plik konfiguracyjny (jeśli jest potrzebny), używany przez klienta SSH.
/etc/ssh/ssh_config	Plik konfiguracyjny dla całego systemu. Jest on ogólnie dostępny i zawiera ustawienia dla użytkowników, którzy nie posiadają plików konfiguracyjnych. Plik ten jest wymagany przez program SSHD podczas jego uruchamiania i jest tworzony w sposób automatyczny podczas instalacji OpenSSH.
$HOME/.ssh/rc	Zawiera polecenia, które zostaną wykonane podczas logowania użytkownika. Polecenia te wykonywane są przed udostępnieniem użytkownikowi powłoki systemowej. Wykorzystywane są do uruchamiania dowolnej wymaganej procedury (w przypadku takiej potrzeby), zanim katalog domowy zostanie udostępniony użytkownikowi. Na przykład, może być wymagany przez użytkownika AFS.
/etc/sshrc	Podobny do pliku *$HOME/.ssh/rc*, z wyjątkiem tego, że zawiera polecenia, które muszą być wykonywane dla całego systemu. Powinien być ogólnie dostępny.

Wykorzystanie SSH do zabezpieczenia transmisji danych poprzez niezabezpieczoną sieć

Zanim rozpoczniesz implementację SSH, musisz upewnić się czy zarówno system lokalny, jak i zdalny posiadają zainstalowany pakiet SSH. Dobrym pomysłem jest również wykorzystanie na każdym z systemów SSH 2.0 lub wersji późniejszej. Zapewnia to stosowanie algorytmu DSA zamiast RSA, który w niektórych krajach podlega ochronie patentowej. W następnych przykładach oba systemy używają systemu Red Hat Linux 7 z zainstalowanym programem SSH 2.1. Dlatego też zostanie użyty algorytm DSA, a instalacja SSH nie jest konieczna, jako że jest on wbudowany w system operacyjny.

Do utworzenia kluczy autoryzacyjnych SSH i zarządzania nimi wykorzystane zostało polecenie `ssh-keygen`. Musisz uruchomić ten program w celu utworzenia kluczy publicznych i prywatnych klienta używającego autoryzacji RSA lub DSA. Poniższe kroki prezentują, w jaki sposób użyć SSH do zabezpieczenia dostępu do zdalnego systemu.

1. W systemie klienta utwórz użytkownika. Nazwij go na przykład `dilbert`. W tym celu wpisz:

```
useradd dilbert
```

2. Dla wspomnianego użytkownika utwórz hasło przez wpisanie:

```
passwd dilbert
Changing password for user dilbert
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
```

3. Zaloguj się jako użytkownik `dilbert`.

4. Utwórz klucze publiczny i prywatny (parę kluczy) dla `dilberta` przy użyciu następującego polecenia:

```
ssh-keygen -d
```

W przypadku wersji 1.3 oraz 1.5 program SSH domyślnie tworzy klucze RSA. W celu wygenerowania kluczy DSA dla wersji programu SSH 2.0, musisz podczas tworzenia pary kluczy określić opcję `-d`.

5. Otrzymasz następującą odpowiedź:

```
Generating DSA parameter and key.
Enter file in which to save the key (/home/dilbert/.ssh/id_dsa) :
```

6. Naciśnij *Enter* w celu zapisania klucza w domyślnym katalogu i pliku (*/home/dilbert/.ssh/id_dsa*). Ujrzysz wtedy następującą odpowiedź:

```
Created directory '/home/dilbert/.ssh'.
Enter passphrase (empty for no passphrase):
```

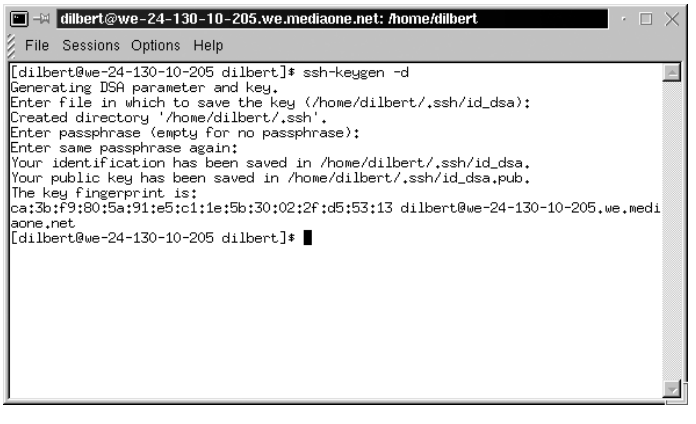
7. Program prosi o podanie hasła. W tym przypadku nie wprowadzaj go. Hasło jest wykorzystywane przez algorytm 3DES w celu zaszyfrowania prywatnej części klucza prywatnego. Dla kluczy serwera hasło musi pozostać puste. Gdy podajesz hasło, nie używaj do jego stworzenia prostych zdań. Zamiast tego podaj przynajmniej od 10 do 30 znaków zawierających cyfry, symbole i litery. Hasło może być później zmienione za pomocą opcji `-p`. W celu wprowadzenia pustego hasła, naciśnij dwukrotnie *Enter*.

Hasła nie można odzyskać. Jeśli zapomnisz użyte hasło, musi być utworzony nowy klucz. Następnie będziesz zmuszony przenieść nowy klucz publiczny na wszystkie wymagane systemy.

8. Zostanie wyświetlone podsumowanie programu tworzącego klucze:

```
Your identification has been saved in /home/dilbert/.ssh/id_dsa.
Your public key has been saved in /home/dilbert/.ssh/id_dsa.pub.
The key fingerprint is:
ca:3b:f9:80:5a:91:e5:c1:1e:5b:30:02:2f:d5:53:13
dilbert@we-24-130-10-205.we.mediaone.net
```

9. Cały proces tworzenia klucza jest pokazany na rysunku 7.13.



Rysunek 7.13. *Tworzenie klucza prywatnego i publicznego za pomocą polecenia ssh-keygen*

10. Utworzyłeś dla Dilberta klucze publiczny i prywatny. Klucz prywatny możesz przejrzeć po wpisaniu następującego polecenia:

```
cat /home/dilbert/.ssh/id_dsa
```

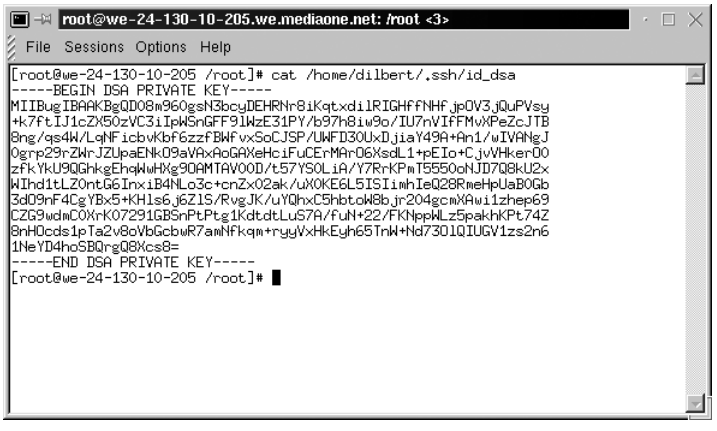
Klucz prywatny będzie podobny do tego pokazanego na rysunku 7.14. Musi on zawsze pozostać zabezpieczony na serwerze lokalnym.

11. Zobacz klucz publiczny, wpisawszy w tym celu:

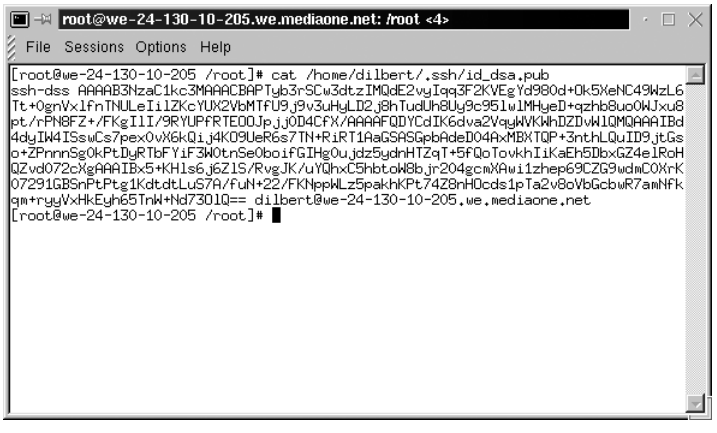
```
cat /home/dilbert/.ssh/id_dsa.pub
```

Klucz publiczny należący do Dilberta może być swobodnie rozpowszechniany. Każdy system, z którym Dilbert będzie chciał wymieniać informacje w sposób bezpieczny, będzie zmuszony uzyskać jego klucz publiczny. Klucz publiczny Dilberta będzie przypominać ten przedstawiony na rysunku 7.15.

Rysunek 7.14. *Klucz prywatny DSA dla użytkownika o nazwie dilbert*



Rysunek 7.15. *Klucz publiczny DSA dla użytkownika o imieniu Dilbert*



Przed rozpowszechnieniem możesz zmienić nazwę klucza publicznego. Możesz na przykład nazwać go *dilbert.pub*. Jest on wtedy znacznie prostszy do zapamiętania i będzie różnić się od innych kluczy publicznych użytkowników utworzonych za pomocą DSA. Po utworzeniu klucza publicznego i prywatnego powinieneś zawsze zmienić nazwę klucza publicznego.

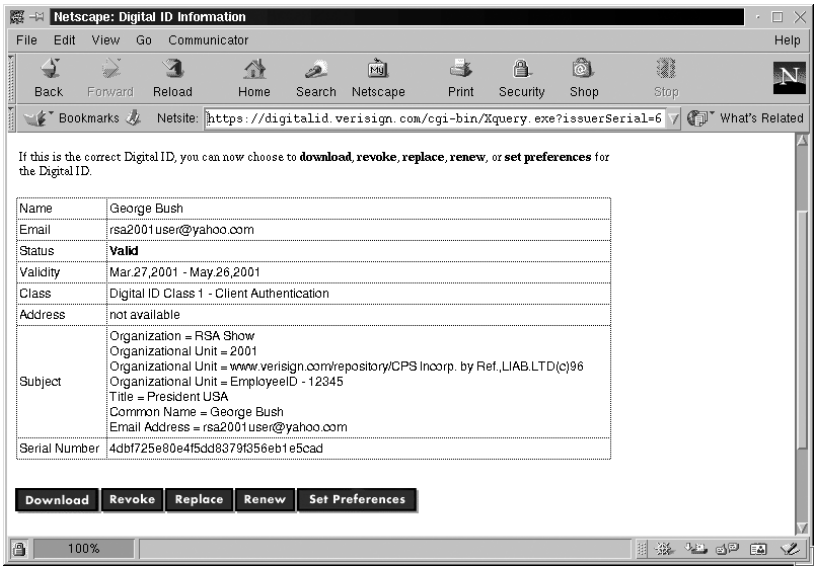
Rozpowszechnianie klucza publicznego

Teraz musisz aktywować klucze przez umieszczenie klucza publicznego w odpowiednim miejscu na zdalnym serwerze. Klucz publiczny może być swobodnie rozpowszechniany, dlatego też możesz wysłać go dowolnym sposobem na zdalny komputer.

Na przykład, firma VeriSign oferuje dla klientów e-mail usługę cyfrowego ID (ang. *digital ID*), służącą do bezpiecznego wymieniania wiadomości e-mail. Cyfrowy identyfikator jest zwykłym kluczem publicznym z osadzoną wewnątrz informacją o ID. VeriSign wysyła w sposób automatyczny klucz publiczny użytkownika na swoją witrynę internetową. Każdy użytkownik pragnący otrzymać dostęp do klucza publicznego dla

określonego użytkownika, może pobrać jego identyfikator cyfrowy ze strony internetowej firmy VeriSign, znajdującej się pod adresem: *https://digitalid.verisign.com/services/client/index.html*.

Rysunek 7.16 pokazuje informacje możliwego do pobrania klucza publicznego (ang. *digital ID*) dla użytkownika o nazwisku George Bush. Istnieją pewne wątpliwości, czy ten George Bush jest prezydentem Stanów Zjednoczonych. Federalne Biuro Dochodzeniowe (FBI) z pewnością chciałoby, aby klucz publiczny prezydenta nie był wyświetlany, co jest opcjonalnie dostępne dla wszystkich użytkowników VeriSign. Gdybyś potrzebował wymieniać informacje z tym użytkownikiem w sposób bezpieczny, musiałbyś pobrać jego klucz publiczny. On również musiałby pobrać Twój klucz publiczny. W momencie gdy obydwaj posiadalibyście klucze publiczne, moglibyście przesyłać do siebie dane w sposób bezpieczny.



Rysunek 7.16. *Pobieranie kluczy publicznych ze strony internetowej firmy VeriSign*

Na potrzeby tej prezentacji przeniesiesz klucz publiczny Dilberta na serwer WWW. Serwer zdalny pobierze następnie klucz publiczny i aktywuje go. Klucz publiczny zostanie aktywowany w momencie umieszczenia w pliku *$HOME/.ssh/authorized_keys2* znajdującym się w katalogu Dilberta na zdalnym serwerze. Poniższe kroki pokazują, w jaki sposób tego dokonać.

1. Przekażesz klucz publiczny Dilberta do serwera WWW o nazwie Apache. Upewnij się czy jest on zainstalowany w Twoim systemie przez wpisanie:

```
rpm -qa | grep apache
```

2. Gdy serwer Apache jest zainstalowany, powinieneś otrzymać odpowiedź podobną do poniższej:

```
apache-manual-1.3.12-25
apache-1.3.12-25
apache-devel-1.3.12-25
```

3. Jeśli nie otrzymasz odpowiedzi, musisz pobrać i zainstalować serwer Apache.

4. W domyślnym katalogu głównym programu Apache utwórz katalog *pubkeys* przez wpisanie w tym celu:

```
mkdir /var/www/html/pubkeys
```

5. Do tego katalogu przekopiuj klucz publiczny Dilberta, zmieniwszy jego nazwę.

```
cp /home/dilbert/.ssh/id_dsa.pub /var/www/html/pubkeys/dilbert.pub
```

6. Przekazałeś w tym momencie serwerowi Apache klucz publiczny. Sprawdź, czy został on zapisany — uruchom przeglądarkę WWW (jak chociażby lynx lub Netscape Navigator) i wpisz:

```
http://localhost/pubkeys/
```

   Domyślnie zostanie wyświetlona zawartość katalogu. Powinieneś ujrzeć plik *dilbert.pub*. Jeśli jednak go nie zobaczysz, sprawdź domyślny katalog ze stronami WWW używany przez serwer Apache i upewnij się, że przekopiowałeś klucz publiczny we właściwe miejsce.

7. Komputer zdalny: musisz użyć drugiego systemu w charakterze zdalnego serwera. Użyjemy go do odwołania się do pierwszego systemu (właśnie skonfigurowanego) w charakterze klienta. Zdalny system w tym przykładzie wykorzystuje system Red Hat Linux 7 i jest umieszczony w tej samej sieci. Zaloguj się do zdalnego serwera jako administrator (`root`).

8. Komputer zdalny: musisz utworzyć konto użytkownika o nazwie `dilbert`. Użyj tego samego hasła co w systemie klienta. Wpisz w tym celu:

```
useradd dilbert
```

9. Komputer zdalny: dla użytkownika `dilbert` utwórz hasło przez wpisanie:

```
passwd dilbert
Changing password for user dilbert
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
```

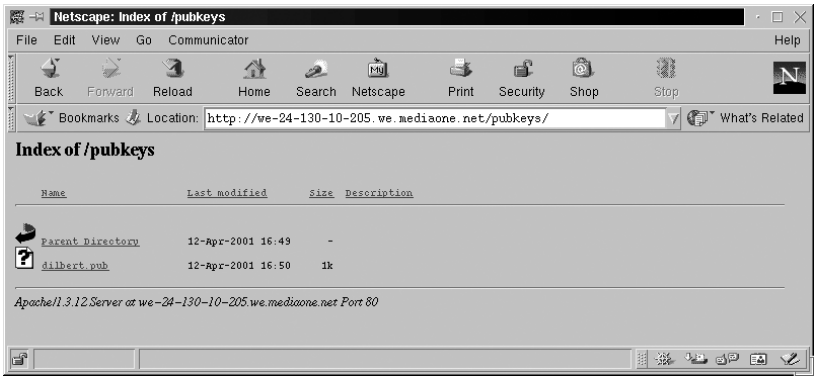
10. Komputer zdalny: w katalogu domowym Dilberta za pomocą następującego polecenia utwórz katalog o nazwie *.ssh*:

```
cd /home/dilbert
mkdir .ssh
```

11. Komputer zdalny: uruchom przeglądarkę internetową i uzyskaj dostęp do klucza publicznego Dilberta umieszczonego na serwerze Apache komputera używanego jako klient. Podaj adres URL serwera i katalog *pubkey*. Jeśli na przykład skonfigurowałeś serwer Apache na *we-24-130-10-205.we.mediaone.net*, wpiszesz:

```
http://we-24-130-10-205.we.mediaone.net/pubkey
```

   Okno przeglądarki będzie przypominać to przedstawione na rysunku 7.17.



Rysunek 7.17. *Dostęp do klucza publicznego z witryny internetowej*

12. Komputer zdalny: pobierz klucz publiczny na komputer zdalny. Na przykład zapisz go w głównym katalogu użytkownika.

13. Komputer zdalny: następnie musisz skopiować zawartość pliku klucza publicznego Dilberta do pliku */home/dilbert/.ssh/suthorized_keys2*. W tej chwili plik ten nie istnieje. Najprostszym sposobem na przeniesienie klucza publicznego do wspomnianego pliku jest przekopiowanie *dibert.pub* i nazwanie go *authorized_keys2*. Jeśli na przykład pobrałeś klucz publiczny Dilberta i zapisałeś go w głównym katalogu użytkownika, napiszesz (znajdując się w głównym katalogu użytkownika):

    ```
    cp dilbert.pub /home/dilbert/.ssh/authorized_keys2
    ```

    Plik o nazwie *authorized_keys2* zawiera klucze publiczne DSA, które mogą być wykorzystane do zalogowania użytkownika. Każdy klucz publiczny musi być wymieniony jako jeden wiersz w pliku. W przypadku przeglądania pliku *id_dsa.pub* w edytorze tekstowym, jest on zapisany w postaci jednego wiersza. Dlatego też bardzo ważne jest, aby podczas umieszczania w pliku *authorized_keys2* klucz znajdował się w jednym wierszu.

14. Komputer klienta: uzyskaj fizyczny dostęp do komputera klienta.

15. Komputer klienta: zaloguj się jako `dilbert`.

16. Komputer klienta: zaloguj się do zdalnego komputera za pomocą ssh. Jeśli zdalny serwer miał adres *we-24-130-8-170.we.mediaone.net*, wpiszesz:

    ```
    ssh we-24-130-8-170.we.mediaone.net
    ```

17. Komputer klienta: uzyskasz odpowiedź podobną do poniższej:

    ```
    The authenticity of host 'we-24-130-8-170.we.mediaone.net' can't be
    established.
    DSA key fingerprint is
    9a:e6:64:34:d5:fa:f7:e4:e9:fd:b7:e5:95:b0:1e:40.
    Are you sure you want to continue connecting (yes/no)?
    ```

    Komunikat `The authenticity of host ' we-24-130-8-170.we.mediaone.net ' can't be established` jest standardowym komunikatem informującym o tym, że zaufane połączenie pomiędzy serwerami nie zostało jeszcze utworzone. Jest to zwyczajny komunikat uzyskiwany za pierwszym razem

podczas tworzenia zaufanego połączenia, zarówno w wersji komercyjnej programu SSH, jak i w wersji OpenSource.

18. Komputer klienta: w celu kontynuacji połączenia wpisz `Yes`. Ukaże się następujący komunikat z ostrzeżeniem, informujący o tym, że do pliku *$HOME/.ssh/known_hosts* na komputerze klienta dodawany jest klucz publiczny pochodzący z komputera zdalnego. Podczas logowania do komputera zdalnego w przyszłości te ostrzeżenia już się nie pojawią, ponieważ zostało już utworzone bezpieczne, zaufane połączenie. Oto wspomniany komunikat zakończony znakiem zachęty (hasło nie jest wymagane dzięki parze kluczy):

```
Warning: Permanently added ' we-24-130-8-170.we.mediaone.net ,24.130.8.170'
➥(DSA) to the list of known hosts.
[dilbert@we-24-130-8-170.we.mediaone.net dilbert]$
```

19. Komputer klienta: otrzymasz znak zachęty używany na zdalnym komputerze dla użytkownika `dilbert`. Wszystkie dane przesyłane pomiędzy Twoim systemem a zdalnym komputerem będą zaszyfrowane.

20. Komputer klienta: zakończ sesję przez wpisanie w tym celu polecenia `exit`.

Jeśli autoryzacja za pomocą klucza publicznego nie powiedzie się lub będziesz zalogowany na komputerze klienta jako użytkownik inny niż `dilbert`, ssh poprosi o podanie hasła:

```
dilbert@we-24-130-8-170.we.mediaone.net's password:
```

Dilbert może bezpiecznie podać swoje hasło dla zdalnego komputera, ponieważ posiada tam swoje konto. Jego nazwa użytkownika oraz hasło zostaną zaszyfrowane za pomocą SSH, tak więc przesyłane dane będą wciąż bezpieczne. Jeśli zostaniesz zapytany o hasło, musisz ponownie prześledzić wszystkie kroki. W przypadku poprawnego skonfigurowania autoryzacji za pomocą klucza publicznego nie zostaniesz zapytany o hasło.

Przechwytywanie i analiza zaszyfrowanych pakietów w sieci

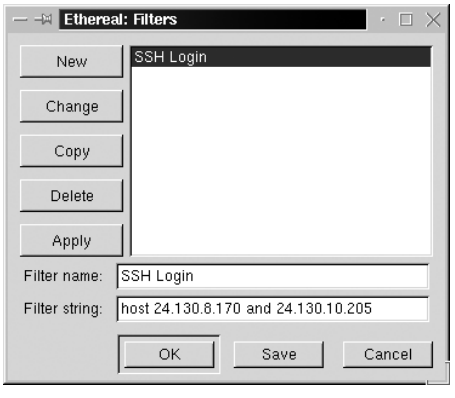
Teraz, kiedy posiadamy już połączenie SSH pomiędzy dwoma komputerami, musimy udowodnić, że sesja jest rzeczywiście zaszyfrowana. Spróbujemy zlokalizować jakiekolwiek dane logowania, jak również dane sesji, a następnie prześledzić strumień TCP.

1. Zaloguj się do zdalnego komputera za pomocą SSH jako administrator (`root`).

2. Komputer zdalny: w celu dodania filtrów do programu `Ethereal` bez używania nazw komputerów, uruchom interpreter poleceń i wpisz:

```
ethereal -n
```

3. Komputer zdalny: wybierz menu *Edit*, a następnie pozycję *Filters*. Pojawi się okno z filtrami programu `Ethereal`.
4. Komputer zdalny: w celu utworzenia filtra dopuszczającego jedynie wymianę danych pomiędzy serwerami używającymi SSH, musisz podać nazwę filtra i definiujący go łańcuch. I tak, w celu utworzenia filtra pomiędzy komputerem zdalnym a komputerem klienta o adresie 24.130.10.205 podaj nazwę „SSH Login" i łańcuch filtra pokazany na rysunku 7.18. Zauważ, że Twoje adresy IP nie będą takie same. Musisz podać adres IP używanych przez Ciebie komputerów klienta i komputera zdalnego używających SSH. Po uzupełnieniu tych dwóch pól, musisz nacisnąć przycisk *Save*, a następnie *New*. Ekran, który się pojawi będzie przypominać ten przedstawiony na rysunku 7.18.



Rysunek 7.18. *Tworzenie filtra pomiędzy dwoma komputerami wykorzystującymi SSH*

5. Komputer zdalny: naciśnij *OK* w celu zamknięcia okna filtrów.
6. Komputer zdalny: w celu rozpoczęcia przechwytywania pakietów, wybierz menu *Capture*, a następnie pozycję *Start*. Pojawi się ekran z właściwościami przechwytywania. Naciśnij *Filter* i wybierz przed chwilą utworzony filtr o nazwie *SSH Login*. Dwukrotnie naciśnij przycisk *OK* w celu rozpoczęcia przechwytywania.
7. Komputer klienta: uzyskaj fizyczny dostęp do komputera klienta i zaloguj się jako użytkownik `dilbert`.
8. Komputer klienta: w celu utworzenia pakietów logowania za pomocą SSH, zaloguj się do zdalnego komputera udostępniającego usługę SSH. Na przykład, w celu zalogowania się do komputera o adresie 24.130.8.170 musisz wpisać:

```
ssh 24.130.8.170
```

9. Komputer klienta: po zalogowaniu się na zdalnym komputerze jako użytkownik o nazwie `dilbert` (dzięki szyfrowaniu za pomocą klucza publicznego nie jest wymagane hasło) wpisz proste polecenie w celu utworzenia danych przesyłanych przez utworzone połączenie. Na przykład napisz:

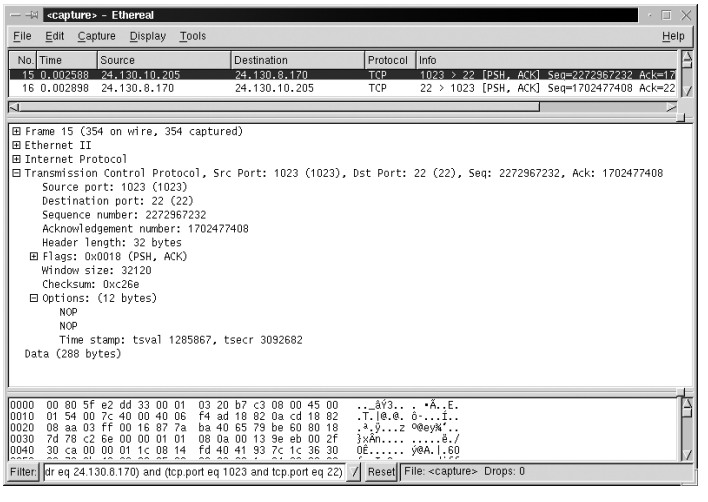
```
/sbin/ifconfig
```

10. Komputer klienta: zakończ sesję SSH przez wydanie polecenia `exit`.

11. Komputer zdalny: uzyskaj fizyczny dostęp do zdalnego komputera. Zatrzymaj przechwytywanie pakietów w programie `Ethereal` naciśnięciem przycisku *Stop*.

12. Komputer zdalny: przechwycone pakiety ukażą się w programie `Ethereal`. Spróbuj zlokalizować pakiet z danymi SSH zawierający pole z hasłem (*password field*). Przypomnij sobie łatwo identyfikowalne pole z hasłem w pakiecie wysłanym przez program `Telnet`, umieszczone na rysunku 7.2. W przechwyconym pakiecie nie powinieneś być w stanie znaleźć żadnych danych z warstwy aplikacji. Ekran, który się pojawi będzie podobny do tego przedstawionego na rysunku 7.19.

Rysunek 7.19. *Próba zlokalizowania sesji logowania programu SSH*



13. Przewiń przechwycony pakiet. Jedyne informacje możliwe do analizy należą do warstw modelu OSI zawartych pomiędzy warstwą 1. (fizyczna) a warstwą 4. (transportowa). Odkryliśmy, że komputer zdalny używający SSH nasłuchuje i przesyła dane na porcie TCP o numerze 22. Klient SSH używa portu TCP o numerze 1023. Nie istnieją żadne hasła, nazwy użytkowników ani inne użyteczne dane. Rysunek 7.20 pokazuje olbrzymią tablicę bezużytecznych pakietów TCP.

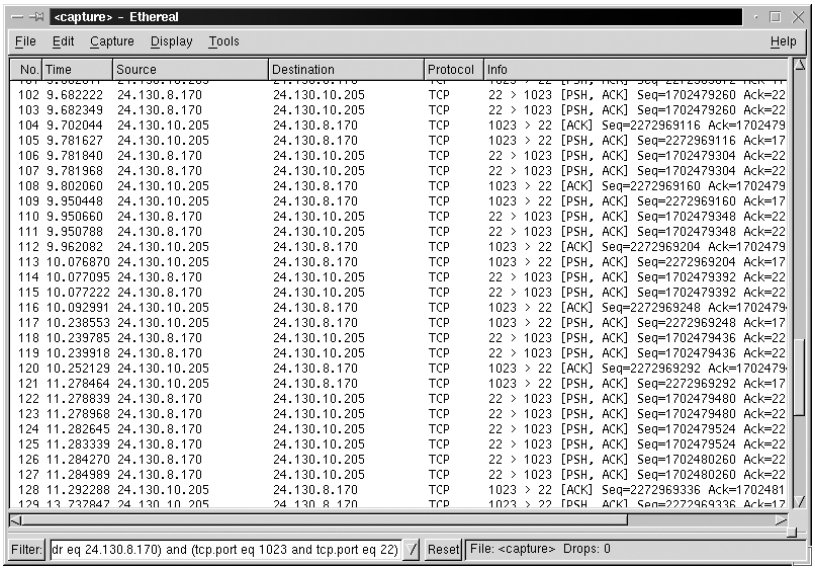
14. Prostszym sposobem na poznanie przydatnych danych jest śledzenie strumienia TCP. W tym celu zaznacz dowolny pakiet uczestniczący w tym połączeniu SSH.

15. Po zaznaczeniu pakietu wybierz menu *Tools*, a następnie *Follow TCP Stream*. Pojawi się zawartość strumienia TCP, tak jak na rysunku 7.21.

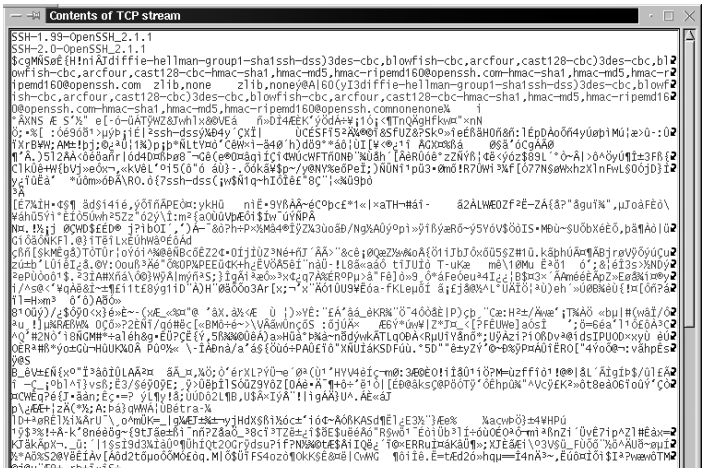
16. Większość danych zaszyfrowana jest za pomocą algorytmu 3DES (domyślny symetryczny algorytm szyfrowania używany w przypadku danych sesji). Program podsłuchujący pakiety, śledząc strumień TCP nie odkrył żadnych użytecznych ani przydatnych danych.

17. Zapisz przechwycony pakiet pod nazwą *secssh* i zakończ pracę programu `Ethereal`.

Rysunek 7.20. *Przechwycone pakiety sesji SSH*



Rysunek 7.21. *Śledzenie strumienia TCP w SSH*



Podsumowanie

W tym rozdziale omówione zostało szyfrowanie sieci oraz powody, które sprawiają, że jest ono tak istotne dla jej bezpieczeństwa. Szyfrowanie sieci zapewnia bezpieczeństwo danych przekazywanych przez sieć z jednego komputera do drugiego. Jeśli program podsłuchujący pakiety przechwyci dane, są one bezużyteczne, ponieważ są zaszyfrowane. Dlatego też haker nie może podejrzeć żadnych nazw użytkownika lub haseł, a jakiekolwiek informacje przesyłane przez sieć są bezpieczne.

W celu zaprezentowania problemów związanych z przekazywaniem niezaszyfrowanych danych, zostały przechwycone niezaszyfrowane pakiety sieciowe, a następnie przeanalizowane w celu pokazania słabych punktów. Nauczyłeś się, że `rlogin`, `rsh` oraz `Telnet` to trzy niebezpieczne protokoły. Nie używają one szyfrowania w przypadku zdalnych logowań lub jakichkolwiek operacji przesyłania danych. Odkryłeś, że `Telnet` przesyła każdy znak hasła w oddzielnym pakiecie. Jeśli będziesz przewijać przechwycone pakiety i przeglądać każdy z pakietów z danymi programu `Telnet`, odkryjesz całe hasło. Prostszą metodą na poznanie hasła jest śledzenie strumienia TCP. Nazwa użytkownika oraz hasło są wyświetlone w postaci jawnego tekstu, podobnie jak wszystkie przesyłane dane.

W celu rozwiązania problemu, dowiedziałeś się, w jaki sposób za pomocą OpenSSH zaszyfrować dane przesyłane w sieci. OpenSSH (*www.openssh.org*) jest programem udostępnianym na zasadach open source, szyfrującym cały ruch pomiędzy serwerami. To bezpieczny program, który może być wykorzystywany w zastępstwie powszechnych programów internetowych, używanych do zdalnych połączeń, takich jak `Telnet`, `rlogin` oraz `FTP`. Ponieważ szyfruje on wszystkie dane, ukrywa zawsze nazwy użytkowników oraz hasła wykorzystywane podczas zdalnego logowania. Po udanym logowaniu, szyfruje on dalej wszystkie dane przesyłane pomiędzy serwerami.

Za pomocą programu OpenSSH dokonałeś bezpiecznej transmisji danych w niezabezpieczonej sieci. Taka operacja wymaga upewnienia się, że OpenSSH został zainstalowany na dwóch różnych serwerach. Jeden z nich był zdalnym serwerem SSH, podczas gdy drugi był jego klientem. SSH udostępnia autoryzację, tworząc dla użytkownika parę kluczy (prywatny i publiczny) za pomocą polecenia `ssh-keygen`. W SSH 2.0 prywatny klucz DSA przechowywany jest w pliku *$HOME/.ssh/id_dsa*. Klucz publiczny powinien być przekopiowany i umieszczony w pliku *$HOME/.ssh/authorized_ keys2*, znajdującym się na komputerze zdalnym. Plik *authorized_keys2* zawiera w każdym wierszu jeden klucz publiczny. Sesje programu SSH 2.0 są zaszyfrowane za pomocą algorytmów ArcFour, CAST128, Blowfish lub 3DES. Integralność danych zapewniają *hmac-md5* oraz *hmac-sha1*. SSH 2.0 jest lepszy i powinien być używany gdy tylko jest to tylko możliwe.

Na koniec sesja SSH została przechwycona i przeanalizowana pod kątem słabości zabezpieczeń. Jedyną informację możliwą do przeanalizowania stanowiły dane zawarte w warstwach od 1. (warstwa fizyczna) do 4. (warstwa transportowa) modelu referencyjnego OSI. Dane logowania nie były możliwe do odczytania, a większość danych była zaszyfrowana przy użyciu algorytmu 3DES (domyślny symetryczny algorytm szyfrowania dla danych sesji). Program przechwytujący pakiety, śledząc strumień TCP nie odkrył żadnych przydatnych danych. Jak widzisz, OpenSSH jest niezbędny i powinien być stale używany w zastępstwie programów `Telnet`, `rsh` oraz `rlogin`.

Szybki przegląd rozwiązań

Podstawy szyfrowania sieci

- Szyfrowanie na poziomie sieci używane jest w przypadku przesyłania jakichkolwiek danych wymagających zachowania poufności. Szyfrowanie zapewnia, że dane przesyłane przez sieć z jednego komputera do drugiego są nieczytelne dla innych osób.

- Programy `rlogin`, `rsh` (ang. *remote shell*, czyli zdalna powłoka systemowa) oraz `Telnet` należą do protokołów niezabezpieczonych. Do operacji zdalnego logowania oraz przekazywania dowolnego rodzaju danych nie wykorzystują one szyfrowania. Jeśli haker przechwyciłby te dane, mógłby wyświetlić w postaci jawnego tekstu nazwy użytkowników lub hasła.

Przechwytywanie i analiza niezaszyfrowanych pakietów w sieci

- Możesz przechwycić pakiety podczas sesji logowania programu `Telnet` przy użyciu programu przechwytującego pakiety o nazwie `Ethereal`, rozpowszechnianego na zasadach open source. Po przechwyceniu sesji, możesz odnaleźć pakiet danych programu `Telnet`, zawierający pole *data:password*.
- Innym sposobem na poznanie hasła programu `Telnet` jest śledzenie strumienia TCP. W tym celu zaznacz po prostu dowolny pakiet uczestniczący w połączeniu za pomocą programu `Telnet`, następnie w programie `Ethereal` wybierz menu *Tools*, a potem pozycję *Follow TCP Stream*. Nazwa użytkownika oraz hasło zostaną wyświetlone w postaci jawnego tekstu.

Wykorzystanie OpenSSH w celu zaszyfrowania pakietów przesyłanych pomiędzy dwoma serwerami

- OpenSSH szyfruje wszystkie dane przesyłane pomiędzy dwom komputerami używającymi SSH (ang. *secure shell*, czyli bezpiecznej powłoki systemowej). Zastępuje on w sposób bezpieczny powszechne programy internetowe wykorzystywane do zdalnych połączeń: `Telnet`, `rlogin` oraz `rsh`.
- Używa on silnych algorytmów szyfrowania: 3DES (ang. *Triple Data Encryption Standard*, czyli standard potrójnego szyfrowania danych) oraz Blowfish, jak również silnej autoryzacji za pomocą kluczy publicznych, haseł jednorazowych (OTP) oraz autoryzacji Kerberos.

Instalacja i konfiguracja SSH na dwóch komputerach sieciowych

- Implementacja OpenSSH różni się znacząco w przypadku różnych systemów operacyjnych. Zespół zajmujący się przenośnością programu OpenSSH wykorzystuje jego wersję przeznaczoną dla systemu OpenBSD do tworzenia przenośnych wersji dla innych systemów operacyjnych. Na stronie *www.openssh.org* musisz upewnić się, czy istnieje określona wersja przeznaczona dla Twojego systemu operacyjnego.
- Metoda używana podczas implementacji SSH łączy w sobie idee podobne do używanych w celu zdalnego wykonywania poleceń z metodami opartymi na kluczach publicznych i prywatnych.

- Za pomocą polecenia `ssh-keygen -d` SSH może utworzyć dla użytkownika parę kluczy DSA (prywatny oraz publiczny). W SSH 2.0 prywatny klucz DSA umieszczony jest w pliku *$HOME/.ssh/id_dsa*. Klucz publiczny natomiast umieszczony jest w pliku *$HOME/.ssh/id_dsa.pub*. Jego nazwa powinna być zmieniona, a klucz powinien być przekopiowany do pliku *$HOME/.ssh/authorized_keys2* umieszczonego w zdalnym systemie.

Wykorzystanie SSH do zabezpieczania transmisji danych poprzez niezabezpieczoną sieć komputerową

- W celu przekazywania danych w sposób bezpieczny, oba komputery muszą posiadać zainstalowany program SSH.
- W pierwszej kolejności musisz użyć polecenia `ssh-keygen` w celu utworzenia na każdym z komputerów klucza publicznego i prywatnego wykorzystującego autoryzację RSA lub DSA. Następnie musisz umieścić klucz publiczny na komputerze, z którym chcesz wymieniać dane w sposób bezpieczny.
- W celu utworzenia połączenia wykorzystującego SSH, używane jest polecenie `ssh` w postaci `ssh zdalny komputer`. Zdalny komputer jest nazwą komputera, z którym za pomocą SSH chcesz nawiązać połączenie.

Przechwytywanie i analiza zaszyfrowanych pakietów w sieci

- W celu określenia czy przesyłane dane są bezpieczne, możesz przechwycić pakiety transmitowane pomiędzy dwoma komputerami używającymi sesji SSH. Możesz spróbować określić jakiekolwiek dane używane podczas logowania, jak również dane sesji.
- Przy użyciu programu `Ethereal` lub dowolnego innego programu przechwytującego pakiety odkryjesz, że wszystkie dane na poziomie aplikacji są zaszyfrowane. Nie zostaną wyświetlone żadne hasła, nazwy użytkowników ani inne użyteczne dane. Śledzenie strumienia TCP jest bezowocne. Podczas przechwytywania danych zostaną wyświetlone jedynie porty TCP.

Najczęściej zadawane pytania (FAQ)

Następujące często zadawane pytania wraz z odpowiedziami autorów tej książki, mają zarówno służyć sprawdzeniu czy zrozumiałeś pojęcia zaprezentowane w tym rozdziale, jak i pomagać Ci w ich stosowaniu. W celu uzyskania odpowiedzi autorów na pytania dotyczące tego rozdziału, wypełnij formularz *Ask the Author* znajdujący się na stronie pod adresem *www.syngress.com/solutions*.

P: Otrzymuję komunikaty z ostrzeżeniem dotyczącym długości kluczy. Co oznaczają te komunikaty i w jaki sposób mogę im zapobiec?

O: Komunikaty z ostrzeżeniami, które obserwujesz, wysyłane są przez program OpenSSH w przypadku, gdy napotka on pewne szczególne, wadliwe klucze RSA oraz DSA, tworzone z powodu błędu w programie `ssh-keygen` (w komercyjnej wersji SSH). Tymi wadliwymi kluczami są klucze publiczne, służące do autoryzacji, których najbardziej znaczący bit (ang. *Most Significant Bit* — MSB) nie jest ustawiony. Z tego powodu klucze te często posiadają połowę poszukiwanej długości (przedstawiają się jako klucze pełnej długości). Komunikat ostrzega Cię o tym, że program OpenSSH wykrył tego rodzaju wadliwy klucz.

Tego rodzaju komunikatom z ostrzeżeniem możesz zapobiec poprzez modyfikację pliku *known_hosts*. Znajdź wpis zawierający długość wadliwego klucza (często 1024) i zmień go na poprawny (zazwyczaj 1023).

Innym rozwiązaniem jest stworzenie nowych kluczy. To podejście jest zalecane, ponieważ nawet po poprawieniu pliku *known_hosts* zmienione klucze są, ogólnie mówiąc, mniej bezpieczne.

P: Dlaczego po uaktualnieniu programu OpenSSH do wersji 2.5.1 utraciłem możliwość obsługi SSH2?

O: Po uaktualnieniu wersji programu OpenSSH, pliki *sshd_config* oraz *ssh_config* mogą zostać zmodyfikowane. Zaleca się sprawdzenie ustawień w tych plikach podczas każdego uaktualniania programu OpenSSH. W przypadku uaktualnienia z wersji 2.3.0 do 2.5.1 możesz do pliku *sshd_config* dodać wiersz:

```
HostKey /etc/ssh_host_dsa_key
```

Ta zmiana spowoduje zachowanie obsługi SSH2.

P: Dlaczego połączenie za pomocą SSH do systemu Linux z *glibs 2.1* trwa tak długo?

O: Implementacja *glibc* użyta w systemie Red Hat Linux 6.1 umożliwia uniwersalną obsługę „IPv6 oraz IPv4". Chociaż ta cecha może być wygodna, rozwiązanie adresów IP z nazw domenowych zajmuje więcej czasu, ponieważ musi zostać ustalona wersja IP.

W celu przyspieszenia tej operacji, możesz użyć opcji `--with-ipv4-default` programu `configure`. Po wpisaniu tej opcji OpenSSH będzie rozwiązywać jedynie adresy IPv4. Podobnie możesz wykorzystać opcję `-6` w celu powiadomienia OpenSSH, aby rozwiązywał on jedynie adresy IPv6.

P: Dlaczego programy `SSHD` lub `configure` stwierdzają czasem, że nie obsługują RSA lub DSA?

O: Biblioteki OpenSSH muszą być zbudowane tak, aby wspierać tę obsługę. Możesz sprawdzić czy program wspiera RSA i DSA w sposób wewnętrzny lub za pomocą RSAref.

P: W mojej dystrybucji nie ma pliku konfiguracyjnego i uruchomienie polecenia `make` nie udaje się. Dlaczego?

O: Jeśli otrzymasz komunikat o braku znaku rozdzielającego, w momencie gdy uruchomienie polecenia `make` nie powiedzie się lub w pobranym przez Ciebie archiwum *tar.gz* brakuje pliku konfiguracyjnego, prawdopodobnie napotykasz ten sam problem. Być może próbujesz skompilować dystrybucję OpenBSD programu OpenSSH w innym systemie operacyjnym niż ten, dla którego był on przeznaczony. W celu bezbłędnego skompilowania programu musisz użyć przenośnej wersji OpenSSH.

P: OpenSSH zawiesza się podczas kończenia pracy z SSH. Dlaczego?

O: Zostały odnotowane przypadki zawieszania się systemów Linux oraz HP-UX w momencie zamykania OpenSSH. Ten błąd pojawia się w obecnych wersjach programu OpenSSH i zdarza się głównie wtedy, kiedy aktywny jest proces drugoplanowy. W celu sprawdzenia występowania tego problemu możesz wpisać `sleep 20&exit`. Strona podręcznika dla używanej przez Ciebie powłoki systemowej powinna zawierać opcję, której możesz użyć w celu wysłania sygnału HUP do aktywnego procesu przy wyjściu. Użytkownicy wykorzystujący powłokę `bash` mogą zastosować następujący wpis w */etc/bashrc* lub *~/.bashrc*:

```
shopt -s huponexit
```