O'REILLY®

Wydanie II

Linux Receptury

Najważniejsze umiejętności użytkownika i administratora

Carla Schroder

Helion

Tytuł oryginału: Linux Cookbook: Essential Skills for Linux Users and System & Network Administrators, 2nd Edition

Tłumaczenie: Robert Górczyński

ISBN: 978-83-283-8765-2

© 2022 Helion S.A.

Authorized Polish translation of the English edition Linux Cookbook 2E ISBN 9781492087168 © 2021 Carla Schroder
This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/liren2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

Spis treści

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

ROZDZIAŁ 12.

Bezpieczny zdalny dostęp za pomocą OpenSSH

OpenSSH to narzędzie używane podczas zdalnej administracji systemem. Szyfruje operację uwierzytelniania i cały ruch sieciowy podczas sesji pracy, a także gwarantuje spójność przekazywanych danych. Jeżeli cokolwiek spowoduje zmianę przekazywanych pakietów, SSH natychmiast poinformuje o tym fakcie. W tym rozdziale wyjaśnię, jak skonfigurować dostęp SSH do zdalnych hostów, jak zarządzać kluczami szyfrowania SSH, jak skonfigurować loginy do wielu zdalnych hostów i dostosować do własnych potrzeb znak zachęty powłoki bash, aby wskazywał na pracę z sesją SSH. Dowiesz się tu także jeszcze wielu innych rzeczy.

OpenSSH obsługuje wiele silnych algorytmów szyfrowania. Żaden z nich nie jest obciążony patentami, ponieważ zespół tworzący OpenSSH dołożył ogromnych starań, aby kod OpenSSH był wolny od patentów i innych ograniczeń. W recepturze 12.16 wyjaśnię, jak wyświetlić listę wszystkich algorytmów obsługiwanych przez OpenSSH.

OpenSSH to pakiet narzędzi przeznaczonych do zdalnego przekazywania plików, składający się m.in. z wymienionych tutaj poleceń:

- `sshd`. To jest demon serwera OpenSSH.
- `ssh`. To skrót od *secure shell* (pol. *bezpieczna powłoka*), choć tak naprawdę to polecenie nie zawiera powłoki, a zapewnia bezpieczny kanał połączenia z powłoką w zdalnym systemie.
- `scp`. To skrót od *secure copy* (pol. *bezpieczne kopiowanie*). Polecenie `scp` pozwala na kopiowanie plików przez zaszyfrowany kanał komunikacji.
- `sftp`. To skrót od *secure file transfer protocol*. Polecenie `sftp` zapewnia dostęp do plików.
- `ssh-copy-id`. To polecenie pozwala na zainstalowanie klucza publicznego w pliku *authorized_keys* zdalnego serwera SSH.
- `ssh-keyscan`. To polecenie wyszukuje i pobiera klucze publiczne hostów w sieci, a tym samym uwalnia użytkownika od samodzielnego zajmowania się tym zadaniem.
- `ssh-keygen`. To polecenie jest przeznaczone do generowania kluczy uwierzytelniania i zarządzania nimi.
- `ssh-add`. To polecenie dodaje tożsamość do agenta uwierzytelniania, `ssh-agent`.

Kup książkę Poleć książkę

W tym rozdziale dowiesz się więcej o poleceniach `ssh`, `sshd`, `ssh-copy-id`, `ssh-keygen` oraz dwóch użytecznych i powiązanych z nimi narzędziach w postaci poleceń `sshfs` i `ssh-agent`.

Polecenie `sshfs` pozwala na montowanie w lokalnym komputerze PC zdalnych systemów plików, natomiast `ssh-agent` zachowuje hasła do kluczy prywatnych SSH dla wielu loginów SSH w celu zapewnienia funkcjonalności automatycznego uwierzytelniania. Polecenie `ssh-agent` jest dołączane do pojedynczej sesji logowania, więc po wylogowaniu lub otwarciu nowego okna powłoki konfigurację `ssh-agent` trzeba rozpocząć od początku. Znacznie lepszym narzędziem przeznaczonym do zautomatyzowanych operacji jest tzw. pęk kluczy (ang. *keychain*), który jest frontendem dla polecenia `ssh-agent`. Pęk kluczy wielokrotnie używa `ssh-agent` aż do ponownego uruchomienia systemu, więc hasło trzeba podać tylko jednokrotnie (zapoznaj się z recepturą 12.10).

OpenSSH obsługuje trzy odmienne typy uwierzytelniania:

Uwierzytelnianie za pomocą hasła

Do uwierzytelniania są używane login i hasło systemu Linux. To jest najprostszy i zapewniający największą elastyczność sposób, ponieważ zalogować można się z poziomu dowolnego komputera. Trzeba zachować ostrożność i nie rozpoczynać sesji SSH z poziomu komputera niegodnego zaufania, np. stojącego w bibliotece bądź kawiarni internetowej. Jeżeli w komputerze znajduje się tzw. *keylogger*, przechwyci Twoje dane uwierzytelniające.

Uwierzytelnianie za pomocą klucza publicznego

Uwierzytelnianie odbywa się za pomocą osobistego klucza publicznego, a nie systemowych danych uwierzytelniających. Przygotowanie takiej konfiguracji wymaga więcej pracy, ponieważ konieczne jest utworzenie i udostępnienie klucza publicznego, a logowanie może odbywać się jedynie z poziomu komputerów zawierających klucz prywatny odpowiadający publicznemu. Niektóre usługi komercyjne wymagają od klientów używania pewnych form uwierzytelniania za pomocą klucza publicznego.

Uwierzytelnianie bez użycia hasła

Uwierzytelnianie odbywa się za pomocą osobistego klucza publicznego, ale bez konieczności podawania hasła. Takie rozwiązanie okazuje się użyteczne w przypadku usług zautomatyzowanych, takich jak skrypty i zadania cron. Użytkownik posiadający uprawnienia wystarczające do kradzieży klucza prywatnego będzie mógł się podszyć pod jego prawowitego właściciela. Dlatego też trzeba zachować szczególną ostrożność w przypadku utworzenia klucza prywatnego bez hasła.

Alternatywą dla używania kluczy niechronionych hasłami jest wykorzystanie pęku kluczy, który będzie pamiętał hasła do kluczy prywatnych (zapoznaj się z recepturą 12.10).

Mamy dwa podstawowe zastosowania dla kluczy uwierzytelniania. Pierwszym jest uwierzytelnianie komputerów za pomocą kluczy hosta, a drugim uwierzytelnianie użytkowników za pomocą kluczy publicznych. Klucze SSH są generowane parami: publiczny i prywatny. Komunikacja jest szyfrowana za pomocą klucza publicznego i deszyfrowana za pomocą klucza prywatnego — to jest przykład niezwykle prostego schematu. Można bezpiecznie udostępniać klucz publiczny, a jednocześnie trzeba koniecznie chronić klucz prywatny i nie wolno nikomu go udostępniać.

Kup książkę Poleć książkę

Serwer i klient są zdefiniowane przez kierunek transmisji danych. Serwer ma uruchomionego demona SSH nasłuchującego żądań połączenia, natomiast klientem jest każdy system logujący się w serwerze za pomocą SSH.

12.1. Instalowanie serwera OpenSSH

Problem

- Chcesz zainstalować serwer OpenSSH.

Rozwiązanie

W większości dystrybucji Linuksa klient OpenSSH jest zainstalowany domyślnie, natomiast serwer OpenSSH już niekoniecznie. W poszczególnych dystrybucjach Linuksa mamy odmienne sposoby rozpowszechniania pakietów OpenSSH. Dlatego też należy użyć menedżera pakietów w celu wyświetlenia dostępnych pakietów (zapoznaj się z „Dodatkiem"). Zainstaluj serwer OpenSSH i następnie sprawdź, czy został uruchomiony.

```
$ systemctl status sshd
● sshd.service - OpenSSH Daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled; vendor preset
   Active: inactive (dead)
   [...]
```

Wygenerowane dane wyjściowe pokazują, że serwer nie jest uruchomiony i nie jest włączony. W większości dystrybucji Linuksa serwer OpenSSH nie jest uruchamiany automatycznie po jego zainstalowaniu. To akurat jest dobre podejście, ponieważ trzeba go prawidłowo skonfigurować przed zezwoleniem na otrzymywanie żądań od klientów. Natomiast jeśli serwer został uruchomiony, jeszcze zanim zdążysz zapoznać się z jego konfiguracją, zatrzymaj go lub w zaporze sieciowej zablokuj port, na którym nasłuchuje.

Kolejnym zadaniem jest wygenerowanie kluczy hosta i skonfigurowanie serwera. Zapoznaj się z recepturami 12.2 i 12.3.

Analiza

Pamiętaj, że serwer i klient to nie tylko kwestia sprzętowa, ale również kierunek transmisji danych. Serwer zawiera uruchomionego demona SSH i nasłuchuje żądań. Z kolei klient to każdy system logujący się do serwera za pomocą SSH. Dowolny system Linux może być serwerem, klientem bądź jednym i drugim.

Zobacz również

- Rozdział 14.
- Witryna domowa OpenSSH (*https://www.openssh.com/*).
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 8 sshd`.
- „Dodatek".

Kup książkę Poleć książkę

12.2. Wygenerowanie nowych kluczy hosta

Problem

Twoja dystrybucja Linuksa nie tworzy automatycznie kluczy hosta po instalacji lub chcesz zastąpić istniejące klucze hosta. Ewentualnie podczas klonowania instalacji bądź maszyny wirtualnej powstałe klony wymagają unikatowych kluczy hosta.

Rozwiązanie

Wykorzystaj polecenie `ssh-keygen`. Dostępne są cztery odmienne typy kluczy: RSA, DSA, ECDSA i ED25519. Zacznij od usunięcia dotychczasowych kluczy, o ile takie istnieją.

```
$ sudo rm /etc/ssh/ssh_host*
```

Utwórz nowe klucze hosta za pomocą następującego polecenia:

```
$ sudo ssh-keygen -A
ssh-keygen: generating new host keys: RSA DSA ECDSA ED25519
```

Analiza

Jeżeli kiedykolwiek będziesz się nudzić i zechcesz wypróbować coś nowego, spróbuj w ulubionej wyszukiwarce internetowej wpisać hasło „jakiego formatu należy użyć dla klucza SSH?”. Argumenty za poszczególnymi typami ciągną się w nieskończoność. Krótka odpowiedź brzmi: użyj RSA, ECDSA lub ED25519, a unikaj DSA. Usuń klucz hosta w formacie DSA i zachowaj pozostałe.

Typ RSA jest najstarszy. Zapewnia silne szyfrowanie i największą zgodność.

ECDSA i ED25519 to nowsze typy, bardzo silne i wymagające mniejszej mocy obliczeniowej.

Część starszych klientów SSH nie obsługuje ECDSA i ED25519. Mam nadzieję, że nie musisz używać tak „antycznych” klientów, ponieważ ECDSA i ED25519 zostały wydane razem z OpenSSH 6.5 w 2014 roku. Niezwykle ważne znaczenie ma regularne uaktualnianie usług zapewnienia bezpieczeństwa i niezezwalanie na używanie starych, niebezpiecznych klientów.

Zobacz również

- Witryna domowa OpenSSH (*https://www.openssh.com/*).
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 ssh-keygen`.

12.3. Konfiguracja serwera OpenSSH

Problem

Chcesz skonfigurować serwer OpenSSH w jak najbezpieczniejszy sposób i dokładnie go przetestować.

Kup książkę Poleć książkę

Rozwiązanie

Przede wszystkim trzeba sprawdzić, czy właścicielem kluczy prywatnych serwera jest root i czy uprawnienia kluczy pozwalają jedynie na ich odczyt.

```
$ ls -l /etc/ssh/
-r-------- 1 root root    227 Jun  4 11:30 ssh_host_ecdsa_key
-r-------- 1 root root    399 Jun  4 11:30 ssh_host_ed25519_key
-r-------- 1 root root   1679 Jun  4 11:30 ssh_host_rsa_key
```

Dokładnie tak powinny wyglądać uprawnienia kluczy prywatnych serwera. Następnym zadaniem jest sprawdzenie kluczy publicznych. Ich właścicielem również jest root, który ma uprawnienia odczytu i zapisu kluczy. Natomiast pozostali użytkownicy mogą jedynie odczytywać klucze publiczne serwera.

```
$ ls -l /etc/ssh/
-rw-r--r-- 1 root root    174 Jun  4 11:30 ssh_host_ecdsa_key.pub
-rw-r--r-- 1 root root     94 Jun  4 11:30 ssh_host_ed25519_key.pub
-rw-r--r-- 1 root root    394 Jun  4 11:30 ssh_host_rsa_key.pub
```

Przedstawione tutaj uprawnienia są prawidłowe.

Przechodzimy teraz do pliku */etc/ssh/sshd_config*. Po zakończeniu modyfikacji tego pliku trzeba ponownie uruchomić usługę `sshd`, aby zmiany zostały zastosowane.

```
$ sudo systemctl reload sshd.server
```

Usuń znak komentarza z początku wierszy zawierających opcje, które chcesz zastosować bądź zmodyfikować.

Skonfiguruj usługę `sshd` do sprawdzania poprawności uprawnień pliku, właściciela plików użytkownika i katalogu domowego przed zezwoleniem danemu użytkownikowi na jego zalogowanie.

```
StrictModes yes
```

Jeżeli uprawnienia pliku są nieprawidłowe, ta opcja nie pozwoli użytkownikowi na zalogowanie się.

Jeżeli komputer ma więcej niż tylko jeden adres IP, trzeba zdefiniować, który adres lub adresy mają być nasłuchiwane.

```
ListenAddress 192.168.10.15
ListenAddress 192.168.10.16
```

Istnieje możliwość przypisania niestandardowych portów nasłuchiwanych przez usługę `sshd`. Używaj jedynie portów o numerach wyższych niż 1024 i sprawdzaj zawartość pliku */etc/services* w celu wyszukania nieużywanych portów, a następnie do wymienionego pliku dodaj nowe porty.

```
sshd 2022
sshd 2023
```

Kolejnym krokiem jest podanie tych portów w pliku */etc/ssh/sshd_config*.

```
Port 2022
Port 2023
```

Istnieje możliwość zezwolenia na dostęp jedynie wybranym grupom (te grupy muszą być wymienione w pliku */etc/group*).

```
AllowGroups webadmins backupadmins
```


Kup książkę Poleć książkę

Natomiast w celu odmowy dostępu dla grupy należy użyć polecenia `DenyGroups`.

Nie należy zezwalać na logowanie z uprawnieniami roota. Znacznie bezpieczniejszym rozwiązaniem jest zalogowanie się jako użytkownik nieuprzywilejowany, a dopiero później użycie polecenia `sudo` do zwiększenia uprawnień.

```
PermitRootLogin no
```

Alternatywne rozwiązanie polega na umożliwieniu logowania roota, ale tylko wykorzystującego uwierzytelnianie za pomocą klucza publicznego.

```
PermitRootLogin prohibit-password
```

Dla wszystkich użytkowników można wyłączyć logowanie z użyciem hasła i zezwolić tylko na uwierzytelnianie za pomocą klucza publicznego (zobacz recepturę 12.7).

```
PasswordAuthentication no
```

Jeżeli zachodzi potrzeba uniemożliwienia logowania określonym użytkownikom, należy ich wymienić w postaci listy nazw użytkowników, nazw użytkowników i hostów bądź nazw użytkowników i adresów IP.

```
DenyUsers duchess madmax stash@example.com cagney@192.168.10.25
```

Aby zezwolić na dostęp wybranym użytkownikom, należy użyć polecenia `AllowUsers`. Ewentualnie można użyć jednocześnie obu (`AllowUsers` i `DenyUsers`). Polecenie `DenyUsers` zawsze jest wykonywane jako pierwsze.

Istnieje możliwość ograniczenia czasu oczekiwania przez serwer na zalogowanie użytkownika i dokończenie procedury nawiązania połączenia. Domyślnie ten czas wynosi 120 sekund.

```
LoginGraceTime 90
```

Ograniczyć można również liczbę nieudanych prób połączenia, wartością domyślną jest 6.

```
MaxAuthTries 4
```

Analiza

Dowolne narzędzie przeznaczone do skanowania portów znajduje otwarte porty, na które później atakujący może przeprowadzić atak typu brute force w celu złamania hasła. Najczęściej atakujący wybierają domyślny port SSH, czyli 22. Zmiana tego portu nie zmniejsza w znaczący sposób niebezpieczeństwa, choć może ograniczyć w dziennikach zdarzeń ilość wpisów dotyczących prób włamania. Gdy chcesz użyć alternatywnego numeru portu, najpierw zajrzyj do pliku */etc/services* i znajdź nieużywany port, a następnie wybrany port zapisz w tym pliku.

Uwierzytelnianie z użyciem klucza publicznego jest bardzo silnie chronione i nie da się go złamać za pomocą ataku typu brute-force jak w przypadku loginów chronionych hasłami (zapoznaj się z recepturą 12.7). W takim przypadku kompromisem jest mniejsza wygoda, ponieważ zalogować można się jedynie z poziomu systemów zawierających klucz prywatny logującego się użytkownika.

Zobacz również

- Witryna domowa OpenSSH (*https://www.openssh.com/*).
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 5 sshd_config`.

Kup książkę Poleć książkę

- Receptura 12.5.
- Receptura 12.7.

12.4. Sprawdzanie składni konfiguracji OpenSSH

Problem

Każdy popełnia błędy i chcesz mieć możliwość sprawdzenia składni pliku */etc/ssh/sshd_config*.

Rozwiązanie

Istnieje taka możliwość. Po wprowadzeniu modyfikacji w pliku */etc/ssh/sshd_config* należy wydać następujące polecenie:

```
$ sudo sshd -t
```

Jeżeli nie zostaną znalezione żadne błędy w składni, polecenie zakończy działanie bez wygenerowania jakiegokolwiek komunikatu. Natomiast w przypadku błędów zostaną wyświetlone odpowiednie komunikaty.

```
$ sudo sshd -t
/etc/ssh/sshd_config: line 9: Bad configuration option: Porotocol
/etc/ssh/sshd_config: terminating, 1 bad configuration options
```

Wymienione polecenie można wydać także wtedy, gdy demon SSH jest uruchomiony. To pozwala na sprawdzenie konfiguracji pod kątem błędów jeszcze przed wydaniem polecenia ponownie uruchamiającego serwer.

Analiza

Opcja `-t` oznacza *test*. Jej użycie nie ma żadnego wpływu na sposób działania demona SSH. Następuje jedynie sprawdzenie pliku */etc/ssh/sshd_config* pod kątem błędów w składni. Dlatego też wymienionej opcji można używać w każdej chwili.

Zobacz również

- Witryna domowa OpenSSH (*https://www.openssh.com/*).
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 5 sshd_config`.

12.5. Konfigurowanie uwierzytelniania na podstawie hasła

Problem

Klienta OpenSSH chcesz skonfigurować do logowania się w zdalnym hoście za pomocą jak najprostszej z obsługiwanych metod.

Kup książkę Poleć książkę

Rozwiązanie

Uwierzytelnianie na podstawie hasła to najprostszy z możliwych sposobów skonfigurowania zdalnego dostępu za pomocą SSH. Takie rozwiązanie ma następujące wymagania:

- Zainstalowany i poprawnie skonfigurowany serwer OpenSSH w komputerze, który ma umożliwiać zdalny dostęp (zapoznaj się z recepturą 12.3).
- Demon SSH uruchomiony w zdalnym komputerze. Port 22 lub inny wybrany dla usługi `sshd` nie powinien być blokowany przez zaporę sieciową.
- Klient SSH zainstalowany w komputerze klienta.
- Konto użytkownika w zdalnym systemie.
- Klucze hosta w serwerze (zobacz recepturę 12.2).

Klucz publiczny hosta musi być przekazany klientom. Najłatwiejszy sposób polega na zalogowaniu się z poziomu klienta i umożliwieniu OpenSSH pobrania klucza.

```
duchess@pc:~$ ssh duchess@server1
  The authenticity of host 'server1 (192.168.43.74)' can't be established.
  ECDSA key fingerprint is SHA256:8iIg9wwFIzLgwiiQ62WNLF5oOS3SL/aTw6gFrtVJTx8.
  Are you sure you want to continue connecting (yes/no)? yes
  Warning: Permanently added 'server1,192.168.43.74' (ECDSA) to the list of
  known hosts.
  Password: hasło
  Last login: Wed Jul  8 19:22:39 2021 from 192.168.43.183
  Have a lot of fun...
```

Teraz użytkownik Duchess może pracować na urządzeniu *server1*, jakby fizycznie siedział przed klawiaturą tego komputera. Uwierzytelnianie i cały ruch sieciowy są szyfrowane.

Pobieranie klucza hosta odbywa się tylko raz, po pierwszym zalogowaniu się do serwera. Użytkownik nigdy nie powinien być ponownie pytany o pobranie klucza, chyba że dotychczasowy klucz został zastąpiony nowym lub został usunięty z pliku *~/.ssh/known_hosts*.

Analiza

Klucz publiczny komputera *server1* jest przechowywany w pliku *~/.ssh/known_hosts* znajdującym się w systemie klienta. Wymieniony plik może przechowywać dowolną liczbę kluczy hostów.

Za niebezpieczne uznaje się logowanie przez SSH jako użytkownik root. Znacznie lepszym rozwiązaniem jest zalogowanie się za pomocą użytkownika nieuprzywilejowanego, a następnie używanie `sudo`, gdy zachodzi potrzeba podniesienia uprawnień. Do zdalnego systemu można się zalogować z użyciem dowolnego użytkownika posiadającego w nim konto. Trzeba jedynie znać hasło tego użytkownika.

```
duchess@pc:~$ ssh madmax@server1
```

Gdy w obu systemach jest używana ta sama nazwa użytkownika, wówczas nie trzeba jej podawać i można zalogować się w następujący sposób:

```
duchess@pc:~$ ssh server1
```

Osobiście wyrobiłam u siebie nawyk podawania nazwy użytkownika za każdym razem i traktuję to jako rodzaj prostego zabezpieczenia przed pomyłkami.

Kup książkę Poleć książkę

Nie należy za bardzo koncentrować się na pojęciach *klient* i *serwer*. W omawianym kontekście nie dotyczą one sprzętu komputerowego. Serwer to system, do którego następuje logowanie przez SSH, natomiast klient to system, z poziomu którego odbywa się logowanie. Usługa `sshd` nie musi działać po stronie klienta.

Istnieje niebezpieczeństwo przechwycenia przekazywanego klucza hosta i zastąpienia go innym, co pozwoliłoby atakującemu uzyskać dostęp do systemu. Masz możliwość weryfikacji odcisku palca klucza publicznego przed wpisaniem słowa `yes` oznaczającego zgodę na dodanie danego klucza do zbioru znanych i akceptowanych. Podczas weryfikacji można wykorzystać staroświecką metodę zapisania odcisków kluczy i ich porównania bądź nowszą i np. smartfonem zrobić zdjęcie klucza hosta i porównać go z dodawanym. Ewentualnie za pomocą wspomnianego telefonu można zadzwonić do osoby posiadającej dostęp do zdalnego systemu i poprosić ją o odczytanie odcisku palca klucza.

W recepturze 12.6 wyjaśniłam, jak pobierać odcisk palca klucza.

Zobacz również

- Receptura 12.6.
- Witryna domowa OpenSSH (*https://www.openssh.com/*).
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 ssh`.
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 ssh-keygen`.
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 8 sshd`.

12.6. Pobieranie odcisku palca klucza

Problem

Potrzebujesz odcisk palca klucza hosta, aby móc zweryfikować klienta.

Rozwiązanie

Po stronie serwera skorzystaj z polecenia `ssh-keygen` razem z kluczem hosta, który chcesz sprawdzić.

```
duchess@server1:~$ ssh-keygen -lf /etc/ssh/ssh_host_rsa_key
4096 SHA256:32Pja4+F2+MTdla9cs4ucecThswRQp6a4xZ+5sC+BfO backup server1 (RSA)
```

Analiza

W omawianym przypadku dobrze sprawdzają się staroświeckie metody komunikacji, takie jak telefon lub tzw. sneakernet (pol. *sieć piesza*). Nie korzystaj z poczty elektronicznej, o ile nie jest ona szyfrowana z wykorzystaniem innej metody szyfrowania i uwierzytelniania, ponieważ wiadomość pocztowa w postaci zwykłego tekstu jest łatwa do przechwycenia i odczytania.

Zobacz również

- Witryna domowa OpenSSH (*https://www.openssh.com/*).
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 ssh-keygen`.

Kup książkę Poleć książkę

12.7. Uwierzytelnianie za pomocą klucza publicznego

Problem

Chcesz wykorzystać uwierzytelnianie za pomocą klucza publicznego, ponieważ jest to metoda bezpieczniejsza niż uwierzytelnianie za pomocą hasła, a ponadto nie zamierzasz używać hasła do swojego konta systemu Linux. Chcesz mieć możliwość użycia pojedynczego klucza publicznego podczas uzyskiwania dostępu do wielu systemów bądź też utworzenia oddzielnych kluczy publicznych do poszczególnych systemów zdalnych.

Rozwiązanie

Linux może spełnić Twoje oczekiwania w podanym zakresie. Możesz więc utworzyć dowolną liczbę kluczy SSH i używać ich w dowolny sposób. W omawianym tutaj przykładzie zamieściłam moje ulubione polecenie przeznaczone do tworzenia nowej pary kluczy RSA. Oczywiście możesz podać własny komentarz i nazwę dla kluczy. (W sekcji „Analiza" przekazałam więcej informacji na temat używania hasła do zabezpieczania klucza prywatnego).

```
duchess@pc:~/.ssh $ ssh-keygen -C "backup server2" -f id-server2 -t rsa -b 4096
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id-server2.
Your public key has been saved in id-server2.pub.
The key fingerprint is:
SHA256:32Pja4+F2+MTdla9cs4ucecThswRQp6a4xZ+5sC+BfO backup server2
The key's randomart image is:
+---[RSA 4096]----+
|          ..     |
|          ....   |
|           o. . .|
|          +  .  o|
|        S* .o o o|
|        +.+..Bo*+|
|         *.+*EX=o|
|        o *o.Oo+.|
|         o.o=+*+.|
+----[SHA256]-----+
```

Następnym krokiem jest skopiowanie nowego klucza do zdalnego systemu, którym w omawianym przykładzie jest lokalny serwer kopii zapasowej, *server1*. Trzeba mieć dostęp SSH do zdalnego systemu, np. dzięki użyciu uwierzytelniania za pomocą klucza hosta, a następnie można użyć `ssh-copy-id` w celu skopiowania klucza publicznego do serwera.

```
duchess@pc:~/.ssh $ ssh-copy-id -i id-server1 duchess@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/id-server1"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys

Number of key(s) added: 1
Now try logging into the machine, with:   "ssh 'duchess@server1'"
and check to make sure that only the key(s) you wanted were added.
```


Kup książkę Poleć książkę

Teraz można spróbować się zalogować.

```
duchess@pc:~/.ssh $ ssh -i id-server1 duchess@server1
Enter passphrase for key 'id-server1':
Last login: Sat Jul 11 11:09:53 2021 from 192.168.43.234
Have a lot of fun...
duchess@server1:~$
```

Nowy klucz można wykorzystać w celu uzyskania dostępu do wielu zdalnych systemów. Ewentualnie można utworzyć oddzielne klucze do poszczególnych zdalnych hostów. Wykorzystanie tego samego klucza ułatwia pracę z wieloma zdalnymi hostami i jednocześnie jest utrudnieniem w przypadku zmiany klucza — trzeba ją będzie przeprowadzić w wielu hostach. Jeżeli unikatowy klucz zostanie złamany lub utracony, wówczas trzeba go będzie zastąpić tylko w jednym miejscu.

Analiza

Zawsze należy definiować hasło w kluczach SSH, z których mają korzystać ludzie. Każdy, kto uzyska dostęp do Twojego klucza prywatnego, będzie mógł z łatwością się pod Ciebie podszywać, jeśli nie zostało zdefiniowane hasło dla tego klucza.

Polecenie `ssh-copy-id` zapewnia dostęp do użytecznego narzędzia gwarantującego, że klucz publiczny został skopiowany w odpowiednie miejsce (w zdalnym systemie to będzie plik *~/.ssh/authorized_keys*), w odpowiednim formacie oraz z odpowiednimi uprawnieniami. Ponadto wymienione narzędzie gwarantuje, że klucz prywatny nie został skopiowany przez pomyłkę.

Oto wybrane opcje dostępne dla polecenia `ssh-copy-id`:

- `-C`. Pozwala na dodanie do klucza komentarza, który pomoże przypomnieć przeznaczenie danego klucza.
- `-f`. Pozwala na nadanie nazwy klucza, może nią być cokolwiek. Pamiętaj o bieżącym katalogu roboczym. Jeżeli nie jest nim *~/.ssh/*, podaj ścieżkę dostępu.
- `-t`. Pozwala na podanie typu klucza. Dozwolone wartości to `rsa`, `ecdsa` i `ed25519`.
- `-b`. Pozwala na określenie siły klucza, opcja używana jedynie w przypadku typu `rsa`. Wartością domyślną jest `2048`, zaś maksymalną `4096`. Im więcej bitów, tym większa moc obliczeniowa wymagana do obsługi klucza. Wątpię, że zauważysz jakąkolwiek różnicę po użyciu 4096 bitów, z wyjątkiem słabego komputera lub bardzo obciążonego serwera.
- `-i`. Pozwala wskazać klientowi SSH klucz do użycia. Jeżeli masz więcej niż tylko jeden klucz, musisz skorzystać z tej opcji. Gdy masz wiele kluczy publicznych i wyraźnie nie wybierzesz konkretnego do użycia, wówczas mogą zacząć się pojawiać komunikaty błędów typu `too many authentication failures`, ponieważ podczas nawiązywania połączenia SSH zostaną wypróbowane wszystkie te klucze.

Zobacz również

- Witryna domowa OpenSSH (*https://www.openssh.com/*).
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 ssh`.
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 ssh-keygen`.

Kup książkę Poleć książkę

12.8. Zarządzanie wieloma kluczami publicznymi

Problem

Chcesz wykorzystać odmienne klucze dla poszczególnych serwerów. Jak można zarządzać kluczami o różnych nazwach?

Rozwiązanie

Podczas tworzenia nowej pary kluczy użyj opcji `-f` polecenia `ssh-keygen`, aby w ten sposób nadać kluczom unikatową nazwę.

```
duchess@pc:~/.ssh $ ssh-keygen -t rsa -f id-server2
```

Następnie za pomocą opcji `-i` wskaż klucz, który ma zostać użyty podczas logowania do zdalnego hosta.

```
duchess@pc:~/.ssh $ ssh -i id-server2 duchess@server2
```

W celu łatwiejszego zarządzania wieloma kluczami publicznymi utwórz plik *~/.ssh_config*. Pozwala on na skonfigurowanie loginów do wielu zdalnych hostów. Dzięki temu można zalogować się za pomocą np. `ssh foo`, zamiast wpisywać długi ciąg tekstowy połączenia. W kolejnym fragmencie kodu pokazałam konfigurację prostego loginu umożliwiającego użytkownikowi Duchess uzyskanie dostępu do hosta *server2*.

```
Host server2
  HostName server2
  User duchess
    IdentityFile ~/.ssh/id-server2
    IdentitiesOnly yes
```

Po przygotowaniu przedstawionej konfiguracji użytkownik Duchess może zalogować się, podając w poleceniu `ssh` wartość zdefiniowaną w wierszu `Host`, jak pokazałam w kolejnym poleceniu.

```
$ ssh server2
```

Do pliku *~/.ssh_config* należy dodać dane logowania używane z pozostałymi kluczami publicznymi, np.:

```
Host server3
  HostName server3
  User duchess
    IdentityFile ~/.ssh/id-server3
    IdentitiesOnly yes

Host server3
  HostName server3
  User madmax
    IdentityFile ~/.ssh/id-server3
    IdentitiesOnly yes
```

Analiza

W przedstawionym wcześniej fragmencie kodu:

- Wiersz `Host` wskazuje na początek danej konfiguracji. To jest etykieta używana podczas logowania, jej wartość może być dowolna.

Kup książkę Poleć książkę

- `HostName` to nazwa zdalnego hosta, w pełni kwalifikowana nazwa domeny lub adres IP.
- `User` to nazwa użytkownika w zdalnym systemie.
- `IdentityFile` to pełna ścieżka dostępu do klucza publicznego.
- `IdentitiesOnly yes` nakazuje SSH użycie ustawień zdefiniowanych w pliku *~/.ssh/config* lub przekazanych w powłoce, ale już nie od innych dostawców, o ile tacy istnieją.

Portem domyślnym dla SSH jest 22. Gdy zachodzi potrzeba nawiązania połączenia z portem niestandardowym, np. 2022, należy go podać w sekcji `Port`.

```
Port 2022
```

Kluczom można nadawać dowolne nazwy. Osobiście lubię nazwy opisowe, dzięki którym od razu wiem, jakie jest przeznaczenie danego klucza.

Pamiętaj, aby zawsze definiować hasła chroniące osobiste klucze prywatne.

Zobacz również

- Witryna domowa OpenSSH (*https://www.openssh.com/*).
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 ssh_config`.
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 ssh`.

12.9. Zmiana hasła chroniącego klucz

Problem

Chcesz zmienić hasło jednego z Twoich kluczy prywatnych.

Rozwiązanie

Skorzystaj z opcji `-p` polecenia `ssh-keygen`.

```
$ ssh-keygen -p -f ~/.ssh/id-server2
Enter old passphrase:
Key has comment 'backup server2'
Enter new passphrase (empty for no passphrase):
Enter same passphrase again: passphrase
Your identification has been saved with the new passphrase.
```

Analiza

Hasła nie można odzyskać. Jeżeli zapomnisz hasła, jedyną możliwością będzie utworzenie nowego klucza z nowym hasłem.

Zobacz również

- Witryna domowa OpenSSH (*https://www.openssh.com/*).

Kup książkę Poleć książkę

- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 ssh_config`.
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 ssh-keygen`.

12.10. Automatyczne zarządzanie hasłami za pomocą pęku kluczy

Problem

Chcesz coś wykorzystać do zapamiętania haseł Twoich kluczy prywatnych, aby były używane, gdy zajdzie potrzeba.

Rozwiązanie

Do tego celu zostało opracowane narzędzie w postaci pęku kluczy (ang. *keychain*). Zainstaluj pakiet `keychain`, a następnie przedstawione nieco dalej polecenie skopiuj do swojego pliku *.bashrc*.

W następnym przykładzie pokazałam, jak można uzyskać dostęp do hostów *server1*, *server2* i *server3* bez wprowadzania hasła za każdym razem. Skopiuj to polecenie w podanej postaci, przy czym nazwy kluczy zmień na swoje.

```
$ keychain ~/.ssh/id-server1 ~/.ssh/id-server2 \
 ~/.ssh/id-server3 . ~/.keychain/$HOSTNAME-sh
```

Pęk kluczy będzie przechowywał klucz prywatny aż do zamknięcia systemu, więc hasło trzeba podawać po każdym uruchomieniu systemu.

W przypadku uruchomienia środowiska graficznego może nie pojawiać się okno dialogowe pozwalające na podanie hasła. Spróbuj uruchomić narzędzie Terminal, a jeśli wciąż nie pojawia się okno dialogowe umożliwiające podanie hasła do pęku kluczy, musisz skorzystać z konsoli Linuksa. Naciśnij klawisze *Ctrl+Alt+F2* i zaloguj się, a zobaczysz wygenerowane komunikaty podobne do następujących:

```
* keychain 2.8.5 ~ http://www.funtoo.org
 * Found existing ssh-agent: 2016
 * Adding 3 ssh key(s): /home/duchess/.ssh/id-server1
/home/duchess/.ssh/id-server2 /home/duchess/.ssh/id-server3
Enter passphrase for /home/duchess/.ssh/id-server1:
Enter passphrase for /home/duchess/.ssh/id-server2:
Enter passphrase for /home/duchess/.ssh/id-server3:
 * ssh-add: Identities added: /home/duchess/.ssh/id-server1
/home/duchess/.ssh/id-server2 /home/duchess/.ssh/id-server3
```

Analiza

Kropka znajdująca się na początku polecenia `. ~/.keychain/$HOSTNAME-sh` to skrót wskazujący na wywołanie polecenia `source`, co oznacza użycie podanego pliku.

`$HOSTNAME` nakazuje pękowi kluczy przeszukanie zmiennych środowiskowych użytkownika w celu pobrania nazwy hosta. Możesz to zrobić samodzielnie przez wydanie następującego polecenia:

Kup książkę Poleć książkę

```
$ echo $HOSTNAME
pc
```

Pęk kluczy jest menedżerem przeznaczonym zarówno dla `ssh-agent`, jak i `gpg-agent`, a jego zadaniem jest buforowanie haseł SSH i GPG przez cały czas działania systemu. Można się wylogować i zalogować ponownie, a mimo tego hasło trzeba będzie ponownie podać jedynie po ponownym uruchomieniu.

Dobrą alternatywą dla pęku kluczy jest `gnome-keyring`, czyli narzędzie działające w środowisku graficznym. Oferuje ono interfejs graficzny przeznaczony do wyświetlania kluczy SSH i GPG i zarządzania nimi, a także zawiera menedżera haseł. W większości dystrybucji to narzędzie nosi nazwę *Hasła i klucze*. Wymienione narzędzie ma dwie wady: nie nadaje się do stosowania w systemie typu headless oraz nie udostępnia haseł mechanizmowi cron (zapoznaj się z recepturą 12.11).

Zobacz również

- Strona domowa projektu Funtoo Keychain (*https://www.funtoo.org/Keychain*).

12.11. Używanie pęku kluczy w celu udostępniania haseł mechanizmowi cron

Problem

Musisz użyć mechanizmu cron w celu automatyzacji zadań, takich jak wykonywanie kopii zapasowej do zdalnego hosta za pomocą polecenia `rsync`. Jednak wszystkie podejmowane próby kończą się niepowodzeniem z powodu błędów generowanych podczas uwierzytelniania.

Rozwiązanie

W celu skonfigurowania pęku kluczy do zarządzania kluczami prywatnymi na potrzeby zadań mechanizmu cron należy utworzyć skrypt przeznaczony do wykonywania przez cron. Spójrz na przedstawiony tutaj przykład skryptu o nazwie *duchess-backup-server1* przeznaczonego do wykonania kopii zapasowej za pomocą polecenia `rsync`.

```
#!/bin/bash
source $HOME/.keychain/${HOSTNAME}-sh
/usr/bin/rsync -ae "ssh -i /home/duchess/.ssh/id-server3" /home/duchess/ \
duchess@server1:/backups/
```

Skrypt musi mieć uprawnienia wykonywalności, które można nadać za pomocą polecenia `chmod`.

```
$ chmod +x duchess-backup-server1
```

Oto wiersz, który trzeba dodać do pliku crontab. To zadanie spowoduje wykonywanie skryptu *duchess-backup-server1* codziennie o godzinie 22.15.

```
15 22 * * * /home/duchess/duchess-backup-server1
```


Kup książkę Poleć książkę

Analiza

W omawianym przykładzie wiersz skryptu rozpoczynający się od */usr/bin/rsync* musi znajdować się w jednym wierszu.

Mechanizm cron działa w swoim własnym ograniczonym środowisku i wymaga, aby pęk kluczy dostarczył niezbędne klucze i zmienne środowiskowe.

Zobacz również

- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 crontab`.
- Strona domowa projektu Funtoo Keychain (*https://www.funtoo.org/Keychain*).

12.12. Bezpieczne tunelowanie sesji środowiska graficznego za pomocą SSH

Problem

Chcesz uruchamiać aplikacje graficzne w zdalnym hoście. Doskonale wiesz, że system X Window ma wbudowane możliwości dotyczące pracy w sieci. Jednak cały ruch sieciowy jest przekazywany w postaci zwykłego tekstu, co jest niebezpieczne, a Ty chcesz pracować zdalnie, ale bezpiecznie.

Rozwiązanie

Tunelowanie sesji X przez SSH nie wymaga żadnego dodatkowego oprogramowania. Należy zacząć od wydania przedstawionych tutaj poleceń w celu sprawdzenia, czy w komputerze klienta działa X11 bądź protokół Wayland. Zamieszczone tutaj przykłady pokazują wyniki obu sprawdzeń.

```
$ echo $XDG_SESSION_TYPE
x11
$ echo $XDG_SESSION_TYPE
wayland
$ loginctl show-session "$XDG_SESSION_ID" -p Type
Type=x11
$ loginctl show-session "$XDG_SESSION_ID" -p Type
Type=wayland
```

Polecenie `loginctl` jest częścią menedżera systemd.

Jeżeli system używa Waylanda, nie możesz zastosować tunelowania przez SSH z powodu braku obsługi sieci przez Waylanda.

Natomiast jeśli system używa X11, trzeba skonfigurować przekazywanie danych X11 za pomocą pliku */etc/ssh/sshd_config* w zdalnym hoście.

```
X11Forwarding yes
```

Kolejne polecenie powoduje zainicjowanie tunelowania X poprzez SSH, używając do tego opcji `-Y`.

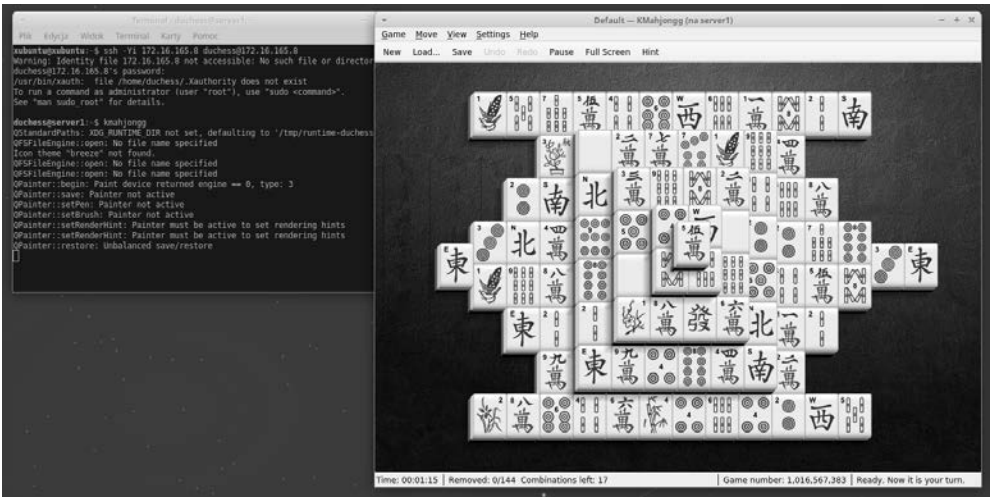
```
duchess@pc:~$ ssh -Yi id-server1 duchess@server1
Last login: Thu Jul  9 09:26:09 2021 from 192.168.43.80
```


Kup książkę Poleć książkę

```
Have a lot of fun..
duchess@server1:~$
```

Teraz można uruchamiać aplikacje graficzne, choć tylko pojedynczo. Na rysunku 12.1 pokazałam uruchomioną grę KMahjongg.

```
duchess@server1:~$ kmahjongg
```



Rysunek 12.1. Gra KMahjongg uruchomiona w zdalnym hoście

Analiza

Działanie serwera X jest przesunięte o wartość podaną w pliku */etc/ssh/sshd_config*, a dokładnie w opcji `X11DisplayOffset`, której wartością domyślną jest 10. To pozwala uniknąć kolizji z istniejącymi sesjami X. Zwykła lokalna sesja X ma wartość przesunięcia `:0.0`, więc pierwsza zdalna sesja X ma wartość przesunięcia `:10.0`. Możesz się o tym przekonać na własne oczy. Przedstawione tutaj polecenie wydaj w komputerze lokalnym. Po raz pierwszy w powłoce lokalnej.

```
duchess@pc:~$ echo $DISPLAY
:0.0
```

Po raz drugi wydaj je w powłoce SSH.

```
duchess@server1:~ssh $ echo $DISPLAY
localhost:10.0
```

Zdalny system musi być włączony. Nie ma konieczności logowania się w nim jakiegokolwiek użytkownika lokalnego ani nawet uruchamiania serwera X. Serwer X musi działać całkowicie w komputerze klienta.

Zobacz również

- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 sshd`.
- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 ssh_config`.

Kup książkę Poleć książkę

12.13. Uruchomienie sesji SSH i wydanie polecenia w jednym wierszu

Problem

W zdalnym hoście chcesz wydać pojedyncze polecenie i uważasz, że byłoby dobrze móc je wykonać bez konieczności logowania się, wykonania właściwego polecenia i następnie wylogowania się. W końcu czy to nie lenistwo jest cnotą administratorów systemu?

Rozwiązanie

OpenSSH pozwala osiągnąć żądany efekt. Spójrz na polecenie pokazujące, jak można ponownie uruchomić serwer poczty Postfix.

```
$ ssh mailadmin@server2.example.com sudo systemctl restart postfix
```

Wprawdzie pojawi się konieczność podania hasła, związana z użyciem polecenia `sudo`, ale mimo tego i tak wykonasz o jeden krok mniej.

Kolejne polecenie pokazuje, jak można otworzyć grę GNOME Sudoku, która wymaga systemu X Window.

```
$ ssh -Y duchess@laptop /usr/games/gnome-sudoku
```

Analiza

Rozwiązaniem alternatywnym jest zastosowanie dla użytkownika root uwierzytelniania z użyciem klucza publicznego, co wyeliminuje konieczność wykorzystania `sudo` (zapoznaj się z recepturą 12.7).

Zobacz również

- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 ssh`.

12.14. Montowanie całego zdalnego systemu plików za pomocą polecenia sshfs

Problem

OpenSSH to oprogramowanie działające szybko i efektywnie, nawet tunelowane aplikacje X Window działają z satysfakcjonującą wydajnością. Chcesz jednak mieć możliwość szybszej edycji większej liczby plików zdalnego hosta, bez konieczności uruchamiania przez SSH graficznego menedżera plików.

Rozwiązanie

Odpowiednim narzędziem dla Ciebie jest polecenie `sshfs`. Służy ono do montowania całych zdalnych systemów plików, a następnie uzyskiwania do nich dostępu, jakby były lokalnymi systemami plików, bez konieczności zajmowania się konfiguracją serwera NFS lub Samby.

Kup książkę Poleć książkę

Zainstaluj pakiet `sshfs`, który powinien zainstalować także FUSE, czyli obsługę systemu plików w przestrzeni użytkownika. Potrzebny jest katalog lokalny z uprawnieniami zapisu, który posłuży jako punkt montowania.

```
duchess@pc:~$ mkdir sshfs
```

Następnie wybrany katalog zdalnego hosta trzeba zamontować w lokalnym katalogu *sshfs*. W omawianym przykładzie katalog domowy *duchess@server2* zostanie zamontowany w katalogu *sshfs* hosta *duchess@pc*.

```
duchess@pc:~$ sshfs duchess@server2: sshfs/
```

Teraz zdalny system plików jest dostępny, jakby był lokalnym systemem plików.

```
duchess@pc:~$ ls sshfs
Desktop
Documents
Downloads
[...]
```

Dostęp do tych plików może odbywać się z poziomu powłoki bądź też graficznego menedżera plików, podobnie jak w przypadku plików znajdujących się lokalnie.

Znak zachęty nie ulegnie zmianie na zdalny znak zachęty.

Po zakończeniu pracy ze zdalnym systemem plików należy go odmontować.

```
duchess@pc:~$ fusermount -u sshfs/
```

We wcześniejszym przykładzie został zamontowany cały katalog domowy użytkownika Duchess. Istnieje możliwość zamontowania jedynie wybranego podkatalogu, np.:

```
duchess@pc:~$ sshfs duchess@server2:/home/duchess/arias sshfs/
```

Nie można użyć tyldy ~ jako skrótu dla */home/użytkownik*, ponieważ narzędzie sshfs nie obsługuje takiej możliwości.

Jeżeli połączenie sieciowe jest zawodne, dobrze jest nakazać narzędziu *sshfs* automatyczne ponowne nawiązywanie połączenia po jego zerwaniu.

```
duchess@pc:~$ sshfs duchess@server2:/home/duchess/arias sshfs/ -o reconnect
```

Analiza

Użytkownicy dopiero rozpoczynający pracę z poleceniem `sshfs` zawsze zadają następujące pytania: dlaczego nie można po prostu uruchomić sesji X poprzez SSH lub dlaczego nie można użyć udziału NFS? Odpowiedzi są proste: polecenie `sshfs` zapewnia szybszą wydajność działania niż sesja X Window przez SSH, a przy tym jest łatwiejsze do skonfigurowania niż udział NFS. Oczywiście możesz używać NFS, Samby i innego dowolnego rozwiązania.

Zobacz również

- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 sshfs`.

Kup książkę Poleć książkę

12.15. Dostosowanie do własnych potrzeb znaku zachęty bash podczas pracy z SSH

Problem

Doskonale wiesz, że znak zachęty zmienia się i wyświetla nazwę zdalnego hosta, gdy zalogujesz się do niego za pomocą SSH. To jednak jest zwykły znak zachęty i łatwo popełnić błąd. Dlatego też chcesz otrzymać niestandardowy i kolorowy znak zachęty, który będzie wyraźnie wskazywał na aktywne użycie loginu SSH.

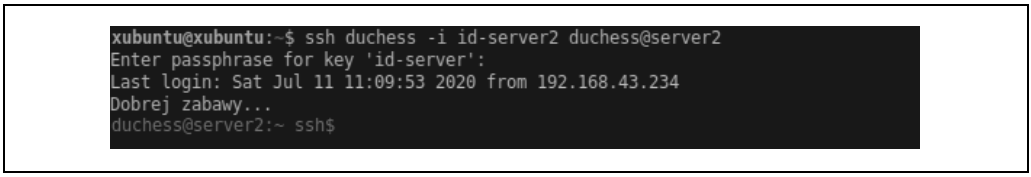
Rozwiązanie

Dostosowanie do własnych potrzeb znaku zachęty bash w zdalnym systemie. Przedstawiony tutaj przykład powoduje zmianę koloru na fioletowy i dodanie ciągu tekstowego `ssh`.

Zamieszczony tutaj fragment kodu umieść w pliku *.bashrc* na zdalnym koncie użytkownika, do którego chcesz się zalogować.

```
if [ -n "$SSH_CLIENT" ]; then text=" ssh"
fi
export PS1='\[\e[0;36m\]\u@\h:\w${text}$\[\e[0m\] '
```

Po zalogowaniu się do tego systemu znak zachęty zostanie odpowiednio zmieniony, jak pokazałam na rysunku 12.2.



Rysunek 12.2. Znak zachęty dostosowany do własnych potrzeb w przypadku używania SSH

Kolor zielony ma tylko znak zachęty, pozostały tekst zostanie wyświetlony z użyciem standardowych kolorów powłoki.

Analiza

Dostosowanie do własnych potrzeb znaku zachęty powłoki bash to temat na oddzielną książkę. Omówiony tutaj przykład można zmodyfikować i zmienić wedle własnych upodobań. Nie musisz używać ciągu tekstowego `ssh` lub zmiennej o nazwie `text` — te dane możesz zmienić na dowolne. Jeżeli chcesz, ciągiem tekstowym może być `super zaszyfrowana sesja`, a nazwą zmiennej może być `ukryta-wiewiórka`.

`[\e[0;31m\]` to blok kodu odpowiedzialny za ustalenie koloru tekstu. Trzeba jedynie zmienić liczbę na określającą żądany kolor.

Blok kodu `[\e[0m\]` wyłącza kolor niestandardowy, więc polecenia i ich dane wyjściowe będą wyświetlane z użyciem zwykłych kolorów powłoki. Spójrz na kody wybranych kolorów:

Kup książkę Poleć książkę

- czarny — `0;30`,
- niebieski — `0;34`,
- zielony — `0;32`,
- cyjanowy — `0;36`,
- czerwony — `0;31`,
- fioletowy — `0;35`,
- brązowy — `0;33`,
- jasnoszary — `0;37`,
- ciemnoszary — `1;30`,
- jasnoniebieski — `1;34`,
- jasnozielony — `1;32`,
- jasnocyjanowy — `1;36`,
- jasnoczerwony — `1;31`,
- jasnofioletowy — `1;35`,
- żółty — `1;33`,
- biały — `1;37`.

Dostosowanie do własnych potrzeb następuje przez sprawdzenie istnienia zmiennej środowiskowej `SSH_CLIENT`, która jest dostępna jedynie w przypadku aktywnego połączenia SSH. Możesz się o tym przekonać samodzielnie przez wydanie w zdalnym hoście następującego polecenia:

```
$ echo $SSH_CLIENT
192.168.43.234 51414 22
```

Powłoka bash „wie”, jak używać niestandardowego znaku zachęty SSH zamiast domyślnego. Po wydaniu przedstawionego polecenia w systemie niezawierającym aktywnej sesji SSH wynikiem jego wykonania będzie pusty wiersz.

Zobacz również

- Strona podręcznika systemowego wyświetlona po wydaniu polecenia `man 1 bash`.
- Sekcja *Colours* w dokumencie *Bash Prompt HOWTO* (*https://tldp.org/HOWTO/Bash-Prompt-HOWTO/x329.html*).

12.16. Wyświetlenie obsługiwanych algorytmów szyfrowania

Problem

Musisz stosować się do pewnych reguł zgodności i chcesz dowiedzieć się, które algorytmy szyfrowania są obsługiwane przez OpenSSH.

Kup książkę Poleć książkę

Rozwiązanie

OpenSSH zawiera polecenia umożliwiające wyświetlenie listy wszystkich obsługiwanych algorytmów — `ssh -Q <opcja>`. Wspomniane algorytmy można wyświetlić za pomocą opcji `help`.

```
$ ssh -Q help
cipher
cipher-auth
compression
kex
kex-gss
key
key-cert
key-plain
key-sig
mac
protocol-version
sig
```

Przedstawione tutaj polecenie powoduje wyświetlenie listy sygnatur algorytmów.

```
$ ssh -Q sig
ssh-ed25519
sk-ssh-ed25519@openssh.com
ssh-rsa
rsa-sha2-256
rsa-sha2-512
ssh-dss
ecdsa-sha2-nistp256
ecdsa-sha2-nistp384
ecdsa-sha2-nistp521
sk-ecdsa-sha2-nistp256@openssh.com
```

Analiza

Oto krótkie omówienie poszczególnych opcji:

- *`cipher`*. Wyświetla listę obsługiwanych algorytmów szyfrowania symetrycznego.
- *`cipher-auth`*. Wyświetla listę obsługiwanych algorytmów szyfrowania symetrycznego razem z szyfrowaniem uwierzytelniania.
- *`compression`*. Wyświetla listę obsługiwanych typów kompresji.
- *`mac`*. Wyświetla listę obsługiwanych kodów spójności wiadomości. Umożliwiają one ochronę spójności i autentyczności danych wiadomości.
- *`kex`*. Wyświetla listę obsługiwanych algorytmów wymiany kluczy.
- *`kex-gss`*. Wyświetla listę obsługiwanych algorytmów typu GSSAPI (ang. *generic security service application program interface*) wymiany kluczy.
- *`key`*. Wyświetla listę typów kluczy.
- *`key-cert`*. Wyświetla listę typów certyfikatów kluczy.
- *`key-plain`*. Wyświetla listę niebędących certyfikatami typów kluczy.
- *`key-sig`*. Wyświetla listę obsługiwanych algorytmów typów i sygnatur kluczy.

Kup książkę Poleć książkę

- *protocol-version*. Wyświetla listę obsługiwanych wersji protokołu SSH. W chwili powstawania książki to tylko wersja 2.
- *sig*. Wyświetla listę obsługiwanych algorytmów sygnatur.

Zobacz również

- Witryna domowa OpenSSH (*https://www.openssh.com/*).
- Książka *Nowoczesna kryptografia. Praktyczne wprowadzenie do szyfrowania*, autor Jean-Philippe Aumasson, Helion.

Kup książkę Poleć książkę


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

...a więc twierdzisz, że dobrze znasz swojego Linuksa?

Dawny Linux był niezawodny w działaniu, ale nie miał efektownego wyglądu i wymagał mozolnej konfiguracji, aby dostosować go do potrzeb użytkowników. Trzeba było znać wiele poleceń, skryptów i plików konfiguracyjnych. Zarządzanie systemem i siecią kosztowało sporo ręcznej pracy. Dzisiejszy Linux jest o wiele łatwiejszy w użytkowaniu. Poszczególne elementy zostały zmienione i usprawnione bez utraty tego, z czego Linux słynął od dawna: niezawodności, wydajności i bezpieczeństwa.

Ta książka przyda się początkującym i średnio zaawansowanym użytkownikom tego systemu. Dzięki niej nauczysz się korzystać z narzędzi graficznych i tych działających w powłoce. Poznasz również podstawy administrowania systemami linuksowymi i przygotujesz się do tego, by sprawnie rozpocząć z nimi pracę. W poszczególnych rozdziałach znajdziesz ponad 250 gotowych receptur, które pomagają poradzić sobie z większością wyzwań stojących przed użytkownikami i administratorami systemów Linux. Plusem publikacji jest to, że zawiera zarówno podstawowe zadania — takie jak instalacja i uruchamianie systemu czy zarządzanie usługami, plikami i katalogami — jak i operacje związane z konfiguracją i zabezpieczaniem sieci. Dodatkowo umieszczono tu rozdział poświęcony instalacji Linuksa na płytce Raspberry Pi, a także szeroki wybór receptur poświęconych rozwiązywaniu typowych problemów z tym systemem.

W książce między innymi:

- korzystanie z systemd
- tworzenie i konfiguracja zapór sieciowych
- zarządzanie użytkownikami, grupami i kontrola dostępu do plików
- sprawdzanie komponentów komputera i monitorowanie jego stanu
- instalacja Linuksa i Windowsa na jednym komputerze
- zarządzanie systemami plików i partycjonowanie dysków

Carla Schroder od połowy lat 90. zeszłego stulecia pracowała jako administrator sieci. Napisała ponad 1000 dokumentów typu HOWTO przeznaczonych do różnych publikacji. Obecnie pisze podręczniki dla użytkowników oprogramowania korporacyjnego dla Linuksa. Słynie z umiejętności przystępnego wyjaśniania trudnych zagadnień.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
Sprawdź nasze szkolenia!
SZKOLENIA
AKADEMIA IT & BUSINESS
HELIONSZKOLENIA.PL
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-283-8765-2
9 788328 387652
INFORMATYKA W NAJLEPSZYM WYDANIU
Cena: 109,00 zł