23 najlepsze sztuczki z wykorzystaniem SSH

4 lutego 2011, Michał Bielawski

OpenSSH to WOLNA implementacja SSH – narzędzie, na którym polega wielu internautów. Użytkownicy takich programów jak telnet, rlogin czy ftp mogą sobie nie zdawać sprawy, że ich hasła przesyłane są w postaci jawnej, ale takie są fakty. OpenSSH szyfruje cały ruch (włącznie z hasłami), żeby efektywnie wyeliminować podsłuchiwanie, przejmowanie połączenia i inne ataki. Dodatkowo, OpenSSH umożliwia bezpieczne tunelowanie,różne metody uwierzytelniania i obsługuje wszystkie wersje SSH.

SSH to naprawdę potężne narzędzie. Jego możliwości są niemal nieograniczone, a poniżej przedstawiam listę tych, na które najczęściej głosowano w ankiecie:

1. **Kopiowanie kluczy SSH do konta użytkownik@host, aby włączyć logowanie bez podawania hasła**

   ```
   ssh-copy-id użytkownik@host
   ```

   W celu wygenerowania kluczy skorzystaj z narzędzia ssh-keygen.

2. **Tworzenie tunelu z portu 80 komputera zdalnego do lokalnego portu 2001**

   ```
   ssh -N -L2001:localhost:80 komputer_zdalny
   ```

   Od teraz możesz wejść na stronę używając adresu http://localhost:2001

3. **Przekierowanie dźwięku z lokalnego mikrofonu na głośniki komputera zdalnego**

   ```
   dd if=/dev/dsp | ssh -c arcfour -C użytkownik@host dd of=/dev/dsp
   ```

   Powyższe polecenie przekieruje dźwięk z Twojego mikrofonu na wyjście dźwiękowe zdalnego komputera. Jakość dźwięku jest kiepska, więc będziesz słyszał sporo szumów.

   *Od tłumacza: skorzystanie z narzędzi **asound** i **aplay**, uruchomionych z odpowiednimi parametrami powinno poprawić jakość, zakładając wystarczającą przepustowość łącza pomiędzy komputerami.*

4. **Porównanie pliku lokalnego ze zdalnym**

   ```
   ssh użytkownik@host cat /sciezka/do/pliku |
   diff /sciezka/do/lokalnego/pliku -
   ```

   Przydaje się do sprawdzania czy między lokalną i zdalną kopią pliku występują jakieś różnice.

5. **Montowanie katalogu/systemu plików przez SSH**

   ```
   sshfs użytkownik@host:/scieżka/do/katalogu
   /scieżka/do/punktu/montowania
   ```

   SSHFS możesz pobrać ze strony fuse.sourceforge.net

   Komenda pozwala na bezpieczne korzystanie ze zdalnych plików.

6. **Połączenie SSH z wykorzystaniem komputera pośredniczącego**

   ```
   ssh -t osiągalny_host ssh nieosiągalny_host
   ```

   Nieosiągalny_host nie jest dostępny z lokalnej sieci, jednak jest widoczny w sieci, do której jest podłączony osiągalny_host. Powyższa komenda utworzy połączenie do nieosiągalny_host poprzez „ukryte” połączenie do osiągalny_host

7. **Kopiowanie katalogu z jednego komputera zdalnego do drugiego, z wykorzystaniem**

komputera lokalnego

```
ssh użytkownik@host1 'cd /katalog/zrodłowy; tar cf - .' |
ssh użytkownik@host2 'cd /katalog/docelowy; tar xf -'
```

Przydatne gdy tylko Ty masz dostęp do obydwu hostów, ale one nie mają dostępu ani do Ciebie (więc ncat nie zadziała), ani do siebie nawzajem.

8. **Uruchomienie dowolnego programu graficznego zdalnie**

```
ssh -fX użytkownik@host program
```

Wymaga to włączenia następującej opcji w serwerze SSH:

`X11Forwarding yes` – domyślna wartość w Debianie i niektórych innych dystrybucjach.

Przydatne może być również:

```
Compression delayed
```

9. **Utworzenie trwałego połączenia do zdalnego hosta**

```
ssh -MNf użytkownik@host
```

Tworzy w tle (-f) trwałe połączenie do hosta. Najlepiej używać w połączeniu z następującymi opcjami w pliku ~/.ssh/config:

```
Host host
ControlPath ~/.ssh/master-%r@%h:%p
ControlMaster no
```

Od tej chwili wszystkie połączenia do danego hosta będą wykorzystywały trwałe połączenie. Jest to bardzo przydatne, jeżeli często synchronizujesz pliki (wykorzystując rsync, sftp, cvs czy svn), gdyż nie będą otwierane nowe połączenia za każdym razem, gdy chcesz się połączyć z komputerem.

10. **Podłącz zdalną sesję screen**

```
ssh -t użytkownik@host screen -r
```

Bezpośrednio podłącza zdalną sesję screen (nie uruchamia zbędnej sesji bash)

11. **Port knocking**

```
knock host 3000 4000 5000 && ssh -p port użytkownik@host &&
knock 5000 4000 3000
```

„Puka” do portów, w celu otwarcia usługi (na przykład serwera ssh), a następnie aby ją zamknąć. Musisz mieć zainstalowany knockd.

Przyjrzyj się poniższej konfiguracji przykładowej:

```
[options]
logfile = /var/log/knockd.log
[openSSH]
sequence = 3000,4000,5000

seq_timeout = 5
command = /sbin/iptables -A INPUT -i eth0 -s %IP% -p tcp –dport 22
-j ACCEPT
tcpflags = syn
[closeSSH]
sequence = 5000,4000,3000
seq_timeout = 5
```

```
command = /sbin/iptables -D INPUT -i eth0 -s %IP% -p tcp –dport 22
-j ACCEPT
tcpflags = syn
```

12. **Usuwanie hosta z listy zaufanych**

    ```
    ssh-keygen -R kolidujący_host
    ```

    Przydatne gdy np. zmienił się klucz danej maszyny. Lepiej używać dedykowanego narzędzia niż usuwać wiersze ręcznie.

13. **Wykonywanie skomplikowanych poleceń powłoki bez korzystania z sekwencji ucieczki**

    ```
    ssh użytkownik@host $(<lista_komend.txt)
    ```

    Dużo łatwiejszy sposób. Bardziej przenośna metoda: `ssh użytkownik@host "`cat lista_komend.txt`"`

14. **Kopiowanie bazy MySQL między serwerami w jednym poleceniu**

    ```
    mysqldump –add-drop-table –extended-insert –force –
    log-error=error.log -uUŻYTKOWNIK -pHASŁO BAZA |
    ssh -C użytkownik@nowy_serwer 'mysql -uUŻYTKOWNIK -pHASŁO BAZA'
    ```

    Wykonuje zrzut bazy MySQL, przesyła go przez SSH z użyciem kompresji i przekazuje do polecenia **mysql** na serwerze docelowym. Uważam, że jest to najlepszy i najszybszy sposób na przeniesienie bazy danych.

15. **Kopiowanie klucza SSH z systemu bez ssh-copy-id**

    ```
    cat ~/.ssh/id_rsa.pub | ssh użytkownik@host
    'mkdir ~/.ssh; cat >> ~/.ssh/authorized_keys'
    ```

    Jeżeli używasz Mac OS X lub innego systemu uniksopodbnego i nie posiadasz ssh-copy-id, to ten jednolinijkowiec pozwoli Ci na dodawanie swojego klucza SSH do zdalnych komputerów, abyś mógł potem logować się bez wpisywania hasła.

16. **Test przepustowości połączenia SSH**

    ```
    yes | pv | ssh użytkownik@host "cat > /dev/null"
    ```

    Łączy się do hosta i wyświetla aktualną przepustowość połączenia, kierując wszystkie dane do /dev/null.

    Wymaga zainstalowanego programu pv

    W Debianie: `apt-get install pv`

    W Fedorze: `yum install pv` (może wymagać włączenia repozytorium ‚extras')

17. **Tworzenie zdalnej sesji screen, do której można się podłączyć ponownie**

    ```
    ssh -t użytkownik@host screen -xRR
    ```

    Długo przed tym, gdy pojawiły się emulatory terminali obsługujące karty, ludzie korzystali z programu GNU Screen, aby otwierać wiele powłok na pojedynczym terminalu tekstowym. W połączeniu z SSH pozwala to na otwarcie wielu sesji powłoki przy użyciu tylko jednego połączenia. Jeżeli odłączysz screena używając `Ctrl-a d`, bądź sesja screen zostanie przez przypadek zakończona, wszystkie Twoje procesy zostaną nienaruszone i będą czekać aż połączysz się ponownie. Inne przydatne komendy screena to `Ctrl-a c` (otwiera nową sesję powłoki) oraz `Ctrl-a a` (przełącza pomiędzy

powłokami). Żeby poznać więcej komend, przeczytaj krótki opis.

18. **Wznawianie kopiowania dużych plików (scp)**

```
rsync –partial –progress –rsh=ssh
plik/źródłowy użytkownik@host:plik/docelowy
```

Wznawia przerwane kopiowanie przy pomocy rsync (przydatne przy przesyłaniu dużych plików, jak na przykład zrzuty baz danych). Wymaga, aby rsync był zainstalowany na obu komputerach.

19. **Analizowanie ruchu zdalnie przy pomocy Wireshark**

```
ssh użytkownik@host 'tshark -f "port !22" -w -'|wireshark -k -i -
```

Przechwytuje to cały ruch na zdalnym hoście przy pomocy tshark, przesyła surowe dane pcap przez połączenie SSH i wyświetla wyniki w Wiresharku. Wciśnięcie `Ctrl+c` spowoduje przerwanie nasłuchiwania, ale niestety zakończy również Wiresharka. Można to obejść przekazując parametr `-c` do tsharka, wskazujący limit pakietów do przechwycenia, lub przekierowując dane do nazwanej kolejki FIFO zamiast bezpośrednio do Wiresharka. Zalecam stosowanie jak najściślejszych filtrów w tsharku, aby oszczędzić łącze. Tsharka można zastąpić tcpdumpem. Polecenie będzie wtedy wyglądać następująco:

```
ssh użytkownik@host tcpdump -w -- 'port !22' | wireshark -k -i -
```

20. **Utrzymywanie wiecznie otwartego połączenia**

```
autossh -M50000 -t host 'screen -raAd session'
```

Otwórz połączenie SSH na zawsze, świetnie się sprawdza w przypadku laptopów tracących połączenie w momencie przełączania się pomiędzy sieciami bezprzewodowymi.

21. **Lepszy, szybszy, silniejszy klient SSH**

```
ssh -4 -C -c blowfish-cbc
```

Wymuszone IPv4, kompresja strumienia, zdefiniowany algorytm szyfrowania. Przypuszczam, że można również użyć aes256-ctr jako algorytmu szyfrowania. Oczywiście pomijam tutaj takie rzeczy jak sesje zarządzające (ang. Master control sessions) i inne, ponieważ mogą być one niedostępne, chociaż również mają wpływ na szybkość.

22. **Limitowanie łącza przy pomocy cstream**

```
tar -cj /backup | cstream -t 777k | ssh host 'tar -xj -C /backup'
```

Powyższe polecenie kompresuje folder korzystając z bzipa i przesyła go do zdalnego hosta z limitem prędkości 777kbit/s. Cstream potrafi dużo więcej, zerknij na http://www.cons.org/cracauer/cstream.html#usage

Dla przykładu:

```
echo w00t, i\'m 733+ | cstream -b1 -t2
```

23. **Kopiowanie zawartości zdalnego pliku do schowka X11**

```
ssh użytkownik@host cat /scieżka/do/pliku | xclip
```

Czy kiedykolwiek musiałeś skopiować plik ze zdalnego hosta tylko po to, żeby skopiować jego zawartość do pisanego właśnie e-maila? Xclip może Ci w tym pomóc. Kopiuje on standardowe wejście do schowka X11, więc wszystko co musisz zrobić, to wcisnąć środkowy przycisk myszy, aby wkleić zawartość tego długiego pliku.