Co to jest SSH?

Autor: Kamil Porembiński (paszczak@thecamels.org)

SSH (ang. secure shell) czyli tłumacząc na polski "bezpieczna powłoka" jest standardem protokołów komunikacyjnych wykorzystywanych w sieciach komputerowych TCP/IP, w architekturze klient - serwer. W wąskim tego słowa znaczeniu SSH jest zdecydowanie lepszym następcą słynnego protokoły **telnet**. SSH podobnie jak telnet służą do łączenia się ze zdalnym komputerem. Jednakże SSH zapewnia szyfrowanie oraz umożliwia rozpoznawanie użytkownika na wiele różnych sposobów. W szerszym znaczeniu SSH jest wspólną nazwą dla całej rodziny protokołów. Obejmuje ona nie tylko podstawowe protokoły służące do zadań terminalowych, ale również do:

- przesyłania plików (SCP - Secure Copy Protocol, SFTP - Secure File Transfer Protocol)
- zdalnej kontroli zasobów komputera
- tunelowania
- forwardowania
- i wiele innch zastosowań

Wspólną cechą tych wszystkich protokołów jest identyczna z SSH technika szyfrowania połączenia, przesyłu danych oraz metoda rozpoznawania użytkownika. Na chwilę obecną SSH praktycznie wyparło inne "bezpieczne" protokoły jak Rlogin i RSH.

Ogólne założenia protokołu SSH powstały w grupie roboczej IETF. Istnieją jego dwie wersje SSH1 i SSH2. W jego wersji 2, możliwe jest użycie dowolnych sposobów szyfrowania danych i 4 różnych sposobów rozpoznawania użytkownika, podczas gdy SSH1 obsługiwał tylko stałą listę kilku sposobów szyfrowania i 2 sposoby rozpoznawania użytkownika (klucz RSA i zwykłe hasło). Najczęściej współcześnie stosowany sposób szyfrowania to AES, choć nadal część serwerów używa szyfrowania Blowfish i technik z rodziny DES. Rozpoznawanie użytkownika może się opierać na tradycyjnym pytaniu o hasło, klucz (RSA lub DSA) lub z użyciem protokołu Kerberos. Domyślnie ustawiony serwer SSH nasłuchuje na porcie 22. Niektóre serwery czekają na pakiety na innych portach w celu zwiększenia bezpieczeństwa.

Historia SSH
W 1995, Tatu Ylönen badacz Uniwersytetu Technologii w Helisnkach stworzył pierwszą wersję protokołu (obecnie nazywana SSH-1), której celem była ochrona przed podsłuchiwaniem haseł w uniwersyteckiej sieci. Ylönen wydał swoją implementację jako Open Source w lipcu 1995 roku. Protokół bardzo szybko stawał się popularny i już pod koniec 1995, SSH było używane przez ponad 20,000 użytkowników w 15 karajach na świecie.

W grudniu 1995, Ylönen założył SSH Communications Security w celu rozwoj SSH. Oryginalna wersja protokoły wykorzystywała wiele części pochodzących z darmowego oprogramowania jak na przykład GNU libgmp. Poźniejsza wersja wydana przez SSH Communications Security ewoluowała coraz bardziej do własnościowego oprogramowania. SSH Communications Security obieło licencją SSH.

W 1996 roku została wydana nowsza wersja protokołu określana mianem SSH-2. Była ona niekompatybilna z SSH-1. Grupa robocza IETF zajęła się standaryzacją SSH-2. Obecnie SSH-2 jest wykorzystywane prawie wszędze. Zapewnia znacznie większe bezpieczeństwo oraz oferuje więcej możliwości. Pod koniec roku 2000 liczba użytkowników SSH przekroczyła 2,000,000.

Architektura SSH
Protokół składa się z 3 warst:

- Transportowej
- Autentykacji (hasła, PKI, etc)
- Połączenia (zwielokrotnianie)

Warstwa transportowa odpowiedzialna jest za ustalenie szyfrowania, kompresji (opcjonalnie) oraz integralności danych. Po przesłaniu za pomocą protokołu 1GB danych lub jeśli sesja trwa ponad godzinę następuje ponowne ustalenie wszystkich parametrów połączenia.

Głównym zadaniem warstwy autentykacji jest wybór metody uwierzytelnienia użytkownika. Do wyboru jest kilka możliwości:

- **password** - najprostsze i najbardziej popularne uwierzytelnianie użytkownika za pomocą hasła
- **publickey** - metoda uwierzytelniania za pomocą kluczy publicznych i prywatnych. Najczęściej wykorzystywane są klucze typu DSA lub RSA. SSH wspiera również certyfikaty X.509.
- **GSSAPI** - metoda, w której wykorzystywane są mechanizmy typu Kerberos lub NTLM. Metody te są używane przez komercyjną wersję SSH.
- **keyboard-interactive** - jest to metoda uwierzytelniania, w której mieszczą się wszystkie inne metody autentykacji, które nie zostały tutaj wymienione. Może być to między innymi metoda uwierzytelniania za pomocą jednorazowego hasła pobieranego z tokena.

Warstwa połączenia jest odpowiedzialna za ustalenie własności kanału w jakim się łączymy:

- shell - łączenie się w trybie terminalu
- direct-tcpip - kanał przekazujący połączenia typu client-to-server
- forwarded-tcpip - kanał przekazujący połączenia typu server-to-client

Lista implementacji SSH
Wieloplatformowe

- PuTTY - klient wspomgający SSH, SFTP, SCP i telnet
- Ganymed SSH2 - client napisany w Javie obsługujący SSH-2
- JavaSSH - oparty na Javie klient SSH

Windows

- WinSCP - darmowy klient SFTP oraz SCP
- freeSSHd - darmowy serwer SSH oferujący SFTP, SCP, forwaroding oraz telnet
- OpenSSH for Windows
- SSHDOS
- Tunnelier - SSH/SFTP ksmslient
- Whitehorn Secure Terminal - darmowy SSH/telnet klient

Macintosh

- MacSSH - klient SSH w terminalu

Uniksopodobne

- Lsh - klient i serwer. projekt GNU
- OpenSSH - bardzo popularny klient i serwer SSH rozwijany przez, developed by OpenBSD
- Dropbear - klient oraz serwer
- libssh - bliblioteka klient-serwer

Instalacja SSH
Instalacja serwera jaki i klienta SSH nie różni się niczym innym od zwykłej instalacji. Jak zwykle mamy do wyboru w zależności kilka metod instlacji.
Źródła: ftp://sunsite.icm.edu.pl/pub/OpenBSD/OpenSSH/portable/
Do instalacji oprogramowania dostarczonego w formie źródłowej niezbędny jest kompilator C, oraz podstawowe narzędzia dostarczane z większością systemów uniksowych.
Na początku rozpakowywujemy źródła i kolejno wydajemy polecenia:
./configure

Wywołując skrypt configure z dodatkowymi opcjami można np. zmienić miejsce instalacji plików konfiguracyjnych lub binariów SSH, jednak w przeważającej większości przypadków ustawienia domyślne są poprawne, i takie będą stosowane w przykładach.
Po skonfigurowaniu pakietu należy go skompilować i zainstalować.
make

a następnie, jako root
make install

Po instalacji zostaną utworzone następujące pliki:
W katalogu **/etc/ssh**:

- ssh_config - plik konfiguracyjny klienta SSH
- ssh_host_key - klucz prywatny serwera SSH
- ssh_host_key.pub - klucz publiczny serwera SSH
- sshd_config - plik konfiguracyjny serwera SSH

W katalogu **/usr/local/bin** :

- make-ssh-known-hosts - skrypt perlowy do generacji plików /etc/ssh_known_hosts
- ssh-askpass - prosty program dla X-Window służący do wczytywania haseł
- scp - secure copy - bezpieczny zamiennik dla komendy rcp
- ssh-add - program służący do wprowadzania kluczy publicznych do agenta autentykacji.
- ssh-agent - agent autentykacyjny
- ssh-keygen - generator kluczy prywatnych i publicznych

- slogin - secure login - link do ssh
- ssh - secure shell - bezpieczny zamiennik rsh

W katalogu **/usr/local/sbin** :

- sshd - serwer ssh

Dodatkowo do podkatalogów w hierarchii **/usr/local/man** zostaną zainstalowane strony dokumentacji do poszczególnych komend plików konfiguracyjnych.
Jeżeli posiadamy dystrybucje typu Debian, Mandriva możemy zawsze zainstalować serwer SSH za pomocą paczek. Wydajemy wtedy odpowiednią dla naszej dystrybucji komendę i czekamy na zainstalowanie binarek. Instalacja OpenSSH pod Debianem:

(Kliknij, żeby powiększyć)

Konfiguracja serwera SSH:
Głównym elementem serwera SSH jest demon sshd. Zastępuje on dwa programy, które znajdują się w większości dystrybucji: rlogin i rsh. Służy do prowadzenia bezpiecznej (kodowanej) komunikacji pomiędzy dwoma komputerami w sieci.

Sshd odczytuje konfigurację z **/etc/ssh/sshd_config** (lub z pliku określonego w linii poleceń, w opcji -t). Plik ten zawiera pary: klucz - wartość. Każda z nich jest zapisana w jednej linii. Linie puste i zaczynające się od znaku '#' są traktowane jako komentarz i są pomijane.
Przykład pliku konfiguracyjnego: sshd_config.
Opis niektórych elementów konfiguracyjnych serwera:

`Port 22`	Ustawienie portu dla serwera. Domyślny portem jest port 22.
`ListenAddress 0.0.0.0` `ListenAddress ::`	Adres IP, na którym serwer SSH ma nasłuchiwać. Ustawienie ma sens, jeśli mamy w komputerze kilka kart sieciowych. Można wtedy wpisać IP karty, która jest np. mniej obciążona.

`# HostKey for protocol version 1`
`HostKey /etc/ssh/ssh_host_key`
`# HostKeys for protocol version 2`
`HostKey /etc/ssh/ssh_host_rsa_key`
`HostKey /etc/ssh/ssh_host_dsa_key`
	Ustawienia ścieżek, w których znajdują się klucze.

`KeyRegenerationInterval 1h`	Czas, po którym maszyna renegocjuje klucze i inne parametry połączenia. Domyślnie 1 godzina.
`PermitRootLogin no`	Parametr określający, czy można logować zdalnie jako root.

`IgnoreRhosts yes`
`IgnoreRootRhosts yes`
`RhostsAuthentication no`
`RhostsRSAAuthentication no`
	Ignorowanie plików .rhosts, które wskazują "zaufane" maszyny, skąd mógłby się zalogować użytkownik bez podawania hasła oraz zezwalanie na autentykację za pomocą mechanizmu rhosts.

`PrintMotd yes`	Czy wypisywać komunikat powitania (motd - Komunikat ustawiamy najczęściej w /etc/motd).

`X11Forwarding no`
`X11DisplayOffset 10`
`X11UseLocalhost yes`
`XAuthLocation /usr/bin/X11/xauth`
	Czy przekazywać dane połączenia X11 (graficznego) i za pomocą którego programu dokonywać autentykacji użytkownika w środowisku graficznym X-Windows.

`# {Allow,Deny}Users | Groups` `DenyGroups student`	Zezwolenie dla grup i użytkowników na logowanie się za pomocą ssh.
`RSAAuthentication yes`	Wybór metody autentykacji. Zalecane ustawić yes tylko dla RSA.

`PasswordAuthentication yes`
`PermitEmptyPasswords no`
`IdleTimeout 30m`
	Autentykacja za pomocą haseł - włączanie, zezwolenie na puste hasła oraz czas rozłączania podczas oczekiwania na podanie hasła.

`# AllowHosts *.our.com friend.other.com`
`DenyHosts home.pl *.algonet.se`
`krakow.tpnet.pl opole.tpnet.pl`
	Z których maszyn można się łączyć za pomocą ssh.

`CheckMail no`	Czy po zalogowaniu ma się odbywać sprawdzanie poczty.
`AllowTcpForwarding no`	Przy takim ustawieniu nie jest możliwe tunelowanie ftp, ale maszyna jest bezpieczniejsza.
`AccountExpireWarningDays 30`	Powiadamianie o kończeniu się ważności konta (dni)
`PasswordExpireWarningDays 14`	Powiadamianie o kończeniu się ważności hasła (dni)
`ForcedPasswdChange yes`	Wymuszanie zmiany już nieważnego hasła

Po skonfigurowaniu serwera przyszedł czas na jego uruchomienie. Jeżeli chcemy, aby serwer uruchamiał się zawsze podczas systemu musisz tylko znaleźć plik wykonywalny sshd (zazwyczaj będzie w **/usr/local/sbin**), uruchomić go i jeśli działa, zrobić odpowiedni wpis uruchamiający w którymś z plików startowych w katalogu **/etc/rc.d/**, tak, by demon sshd uruchamiał się podczas każdego startu systemu. Najprostszym sposobem jest dopisanie na końcu pliku **/etc/rc.d/rc.local** (lub innego pliku konfiguracyjnego maszyny, który jest wykonywany podczas jej startu) ścieżki do demona sshd, czyli najprawdopodobniej **/usr/local/sbin/sshd**. Posiadacze systemu Mandrake/Mandriva mogę z konta root wykonać polecenie:
service sshd start

i cieszyć się gotowym serwerem SSH.

Konfiguracja klienta SSH:
Ogólnosystemowa konfiguracja klienta ssh znajduje się w pliku **/etc/ssh/ssh_config**, zaś opcje konfiguracyjne sprawdzane są w następującej kolejności:
opcje podane w linii komend
plik konfiguracyjny użytkownika ($HOME.ssh/config)
plik ogólnosystemowy

Przykład pliku konfiguracyjnego: ssh_config.

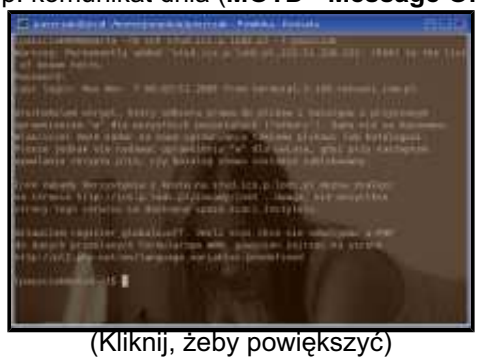
`Hosts *`	Otwiera sekcję dotyczącą połączeń do danego hosta - ***** oznacza wszystkie hosty.
`ForwardAgent yes`	Określa, czy Agent ma przekazywać klucze, czy nie.
`ForwardX11 yes`	Zezwala użytkownikom na przekazywanie połączeń X11
`RhostsAuthentication no` `RhostsRSAAuthentication no`	Zezwalanie na autentykację za pomocą mechanizmu rhosts.
`PasswordAuthentication yes`	Autentykacja za pomocą haseł.
`RSAAuthentication yes` `TISAuthentication no`	Wybór metody autentykacji. Zalecane ustawić tylko RSA.
`PasswordPromptHost yes` `PasswordPromptLogin yes`	Czy program ma pytać o hasła.
`FallBackToRsh no` `UseRsh no`	Możliwość użycia rsh w przypadku niepowodzenia połączenia za pomocą ssh. Zalecane usawić na nie.
`BatchMode no`	Możliwość użycia ssh w trybie wsadowym. Wyłączyć gdy nie jest koniecznie potrzebne. Może się przydać tylko w kilku przypadkach.
`EscapeChar ~`	Jaki znak powoduje wyjście z połączenia (jak w telnecie **`ctrl+]`**)
`Cipher 3DES`	Algorytm stosowany do szyfrowania przy połączeniu ze zdalną maszyną.

`Compression yes` `CompressionLevel 9`	Kompresja - domyślnie jest włączona, poziom - 6. Dziewięć jest najwyższym, 0 wyłącza.
`IdentityFile ~/.ssh/identity`	Położenie i nazwa pliku identyfikacji

Łączenie z serwerem SSH:
Do połączenia z serwerami SSH w systemach uniksowych tak naprawdę wystarcza tylko konsola. Wydajemy polecenie:
ssh mojserwer.pl

i czekamy na połączenie. Klient ssh będzie starał się zalogować nas na serwerze o tym samym nicku jakim aktualnie posługujemy się w lokalnym systemie. Jeśli zalogować się jako inny użytkownik piszemy:
ssh mojserwer.pl -l nazwa_użytkownika

Po zalogowaniu możemy zobaczyć np. komunikat dnia (**MOTD - Message Of The Day**):



(Kliknij, żeby powiększyć)

Zwolennicy graficznych porgramów mogą skorzystać z programu Putty. Wygląda on następująco:

(Kliknij, żeby powiększyć)

Bibliografia:
www.google.com
www.wikipedia.pl
http://matrix.umcs.lublin.pl/pusu/2001/referaty/ssh/podstr/konfig.htm

Jeśli masz jakieś pytania dotyczące Linuksa lub konfiguracji SSH zapytań na naszym forum