Bezpieczeństwo systemów informatycznych

ĆWICZENIE SSH

1. Secure Shell i protokół SSH

1.1 Protokół SSH

Protokół SSH umożliwia bezpieczny dostęp do zdalnego konta systemu operacyjnego z zastosowaniem kryptograficznego uwierzytelniania i szyfrowania transmisji. Ponadto możliwe jest też szyfrowane tunelowanie ruchu pomiędzy arbitralnymi portami TCP, co umożliwia tworzenie tuneli wirtualnych na poziomie aplikacyjnym (propagacja połączeń). Transmitowane dane mogą podlegać automatycznej kompresji.

1.2 Program ssh

Program ssh zastępuje w działaniu programy z rodziny *remote operations* (r∗), jak rlogin:

```
local> ssh -l username remotehost
```

lub rsh:

```
local> ssh -l username remotehost cat /etc/HOSTNAME
```

Umożliwia jednakże osiągnięcie podwyższonego stopnia bezpieczeństwa przy dostępie do zdalnego konta. Cała transmisja jest bowiem szyfrowana (łącznie z procedurą logowania, a więc i ewentualnym przesłaniem hasła), a do weryfikacji tożsamości użytkownika również stosowane są mechanizmy kryptograficzne.

Nazwa użytkownika zdalnego konta może być podana w postaci `username@remotehost`, np.:

```
local> ssh username@remotehost cat /etc/HOSTNAME
```

Sesja zdalnego dostępu jest ustanawiana po kryptograficznym uwierzytelnieniu systemu zdalnego metodą *challenge-response* (z wykorzystaniem np. algorytmu RSA lub ECDSA) oraz po ustaleniu symetrycznego klucza sesji (metodą Diffiego-Hellmana). Kryptograficzne uwierzytelnianie systemu zdalnego wykorzystuje zbiór kluczy publicznych zdalnych systemów znanych systemowi lokalnemu, przechowywanych w plikach /usr/local/etc/ssh_known_hosts oraz ~/.ssh/known_hosts. Tajność kluczy prywatnych wystarcza do obrony systemu przed atakami typu name spoofing, IP spoofing czy routing spoofing.

1.2.1 Metody uwierzytelniania użytkownika

Program ssh może dokonać uwierzytelnienia użytkownika metodą klasyczną, poprzez hasło, komunikacja jest jednak szyfrowana, więc hasło nie jest transmitowane tekstem jawnym. Jednak możliwe jest kryptograficzne uwierzytelnienie użytkownika metodą *challenge-response*.

Kryptograficzne uwierzytelnianie zdalnego użytkownika wykorzystuje zbiór kluczy publicznych zaufanych użytkowników, przechowywanych w pliku ~/.ssh/authorized_keys. Aby umożliwić uwierzytelnienie kryptograficzne na zdalnym serwerze należy wybrany klucz publiczny umieścić na docelowym koncie (zdalnego serwera) w pliku ~/.ssh/authorized_keys, np. poleceniem `ssh-copy-id`.

1.2.2 Stosowane algorytmy kryptograficzne

SSH dokonuje uwierzytelnienia kryptograficznego przy pomocy np. algorytmu RSA lub ECDSA. Do szyfrowania komunikacji stosowane symetryczne algorytmy kryptograficzne, m.in. AES lub Chacha, a do zapewnienia integralności komunikacji np. HMAC lub UMAC. Konkretne algorytmy można wskazywać odpowiednią opcją pliku konfiguracyjnego ~/.ssh/config (opcje `Ciphers`, `MACs`).

1.3 Polecenie scp

Polecenie zdalnego kopiowania plików scp zastępuje rcp z rodziny *remote operations*:

```
local> scp username@remotehost:/etc/HOSTNAME  remotename.txt
```

Polecenie scp obsługuje wszystkie wymienione wyżej mechanizmy uwierzytelniania zdalnego dostępu.

1.4 Zarządzanie kluczami kryptograficznymi

Do tworzenia pary kluczy kryptograficznych służy program ssh-keygen. Zapisuje on klucz prywatny użytkownika wygenerowany dla algorytmu RSA (ew. ECDSA) w pliku ~/.ssh/id_rsa (~/.ssh/id_ecdsa), a klucz publiczny w pliku ~/.ssh/id_rsa.pub (~/.ssh/id_ecdsa.pub). W celu dodatkowego zabezpieczenia klucz prywatny może być zapisany w pliku w postaci zaszyfrowanej algorytmem zaszyfrowanej. W takim przypadku, późniejszy dostęp do tego klucza wymagać będzie podania sekwencji `passphrase`.

```
local> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (~/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ~/.ssh/id_rsa.
Your public key has been saved in ~/.ssh/id_rsa.pub.
The key fingerprint is:
5c:13:fc:48:75:19:9b:27:ec:5c:4f:d3:b1:a2:6c:da user@host.domain
```

Polecenie `ssh-copy-id` skopiuje wówczas klucz publiczny na wskazane konto w zdalnym systemie.

1.4.1 Zarządzanie kluczami zdalnych systemów

Program ssh-keygen jest również wykorzystywany do generowania kluczy systemu, prywatnego oraz publicznego, odpowiednio w plikach: /usr/local/etc/ssh_hosts_key i /usr/local/etc/ssh_hosts_key.pub. Ponadto pozwala zarządzać bazą kluczy known-hosts:

– wyświetlenie klucza systemu o nazwie `remotehost`:

```
local> ssh-keygen -F remotehost
```

– usunięcie klucza systemu o nazwie `remotehost`:

```
local> ssh-keygen -R remotehost
```

1.5 Tunele wirtualne warstwy aplikacji (TCP port forwarding)

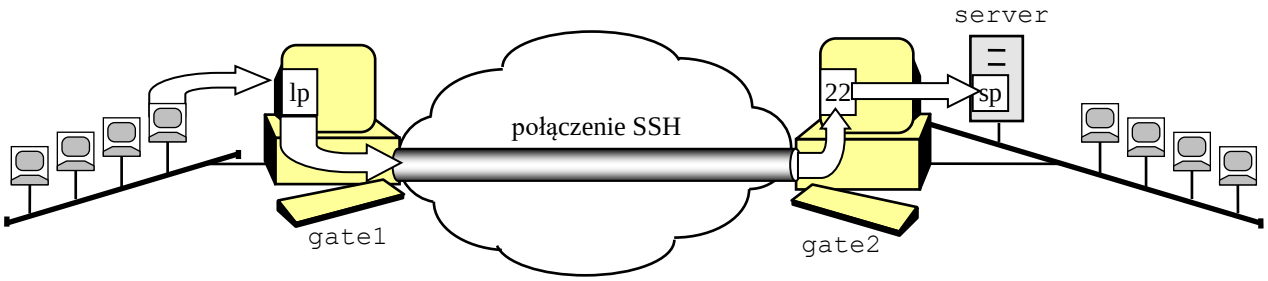
Istnieje możliwość zbudowania tunelu wirtualnego pomiędzy dwoma komputerami (np. bramami typu BastionHost), przechodzącego przez sieć niezabezpieczoną (np. publiczną), Tunel ten może być następnie wykorzystywany do bezpiecznej komunikacji w warstwie aplikacyjnej pomiędzy innymi komputerami (np. klientem znajdującym się przed pierwszą bramą i serwerem za drugą bramą).

```
gate1> ssh -L [bind_address:]localport:server:serviceport gate2
```

Takie polecenie utworzy tunel pomiędzy bramami `gate1` i `gate2`, który umożliwi komunikację poprzez port `localport` z usługą `serviceport` na komputerze `server` (`-L` = Local port forwarding). Komunikacja jest zaszyfrowana na odcinku `gate1-gate2`.

`bind_address` – opcjonalny adres lokalnego interfejsu do nasłuchiwania (bramy `gate1`)
`localport` – port TCP lokalnego systemu (bramy `gate1`)
`server` – nazwa (domenowa) lub adres (IP) komputera, którego port będzie udostępniany poprzez tunel wirtualny (za bramą `gate2`)
`serviceport` – udostępniany zdalnie port TCP na serwerze `server`

Propagowanie połączeń w tunelu przedstawia schematycznie poniższy rysunek.



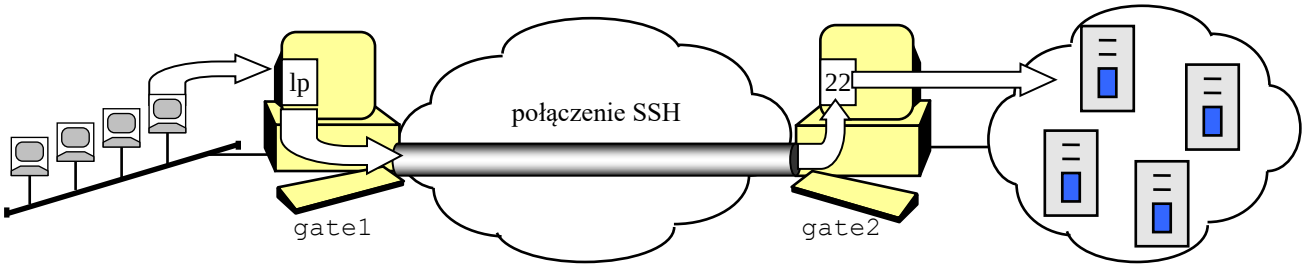
Identyczny efekt do poprzedniego ma polecenie zestawiające tunel od strony przeciwnej:

```
gate2> ssh -R [bind_address:]remoteport:server:serviceport gate1
```

(`-R` = Remote port forwarding)

`bind_address` – opcjonalny adres zdalnego interfejsu do nasłuchiwania (bramy `gate1`)
`remoteport` – port TCP zdalnego systemu (bramy `gate1`)
`server` – nazwa (domenowa) lub adres (IP) komputera, którego port będzie udostępniany poprzez tunel wirtualny (teraz przed bramą `gate2`)
`serviceport` – port TCP na serwerze `server`

Istnieje także tunel umożliwiający wykorzystanie SSH jako pełne proxy aplikacyjne (SOCKS):



Efekt taki uzyskamy poleceniem:

```
gate1> ssh -D [bind_address:]localport gate2
```

(`-D` = application-level port forwarding)

`localport` – port TCP lokalnego systemu (bramy `gate1`)

1.6 Tunele wirtualne warstwy sieciowej (VPN)

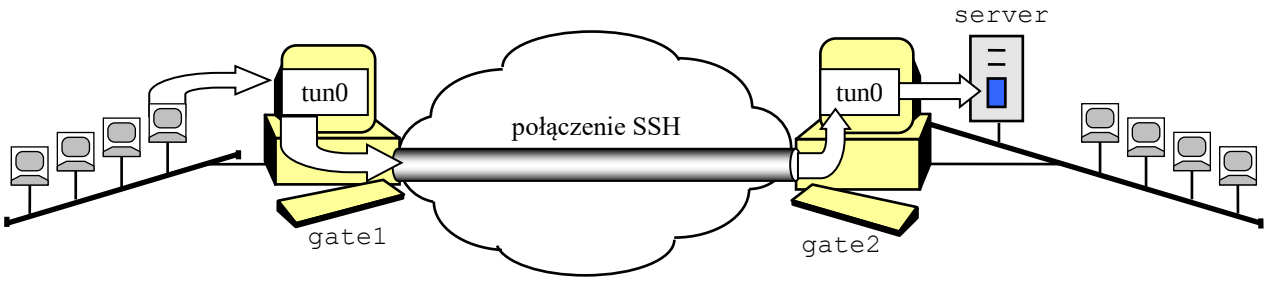
SSH oferuje też zbudowania tunelu wirtualnego pomiędzy dwoma komputerami (np. bramami typu BastionHost), przechodzącego przez sieć niezabezpieczoną (np. publiczną). Tunel ten może być następnie wykorzystywany do bezpiecznej komunikacji w warstwie sieciowej (IP) pomiędzy innymi komputerami (np. klientem znajdującym się przed pierwszą bramą i serwerem za drugą bramą).

```
gate1> ssh -w local_tun[:remote_tun] gate2
```

Takie polecenie utworzy tunel VPN pomiędzy bramami `gate1` i `gate2`, który umożliwi tunelowanie komunikacji między wirtualnymi interfejsami `local_tun` i `remote_tun`. Dodatkowo należy ustalić adresy IPv4 lub IPv6 na tych interfejsach oraz odpowiedni routing pakietów. Komunikacja jest zaszyfrowana na odcinku `gate1-gate2`. Do prawidłowego działania połączenie SSH musi zostać ustanowione pomiędzy administratorami obu systemów.

`local_tun` – numer wirtualnego interfejsu po stronie lokalnej (lub `any` = najbliższy wolny numer wirtualnego interfejsu)
`remte_tun` – numer wirtualnego interfejsu po stronie zdalnej

Propagowanie połączeń w tunelu przedstawia schematycznie poniższy rysunek.



1.7 Plik konfiguracyjny

Istnieje możliwość skonstruowania profili połączeń. Profile należy zapisać w pliku ~/.ssh/config.
Profil plik konfiguracyjny ma następującą strukturę:

```
Host nazwa_profilu
  Hostname adres_systemu
  User nazwa_użytkownika
  Port numer_portu
  LocalForward localport server:serviceport
  RemoteForward remoteport server:serviceport
  DynamicForward localport
Host nazwa_kolejnego_profilu
…
```

Uruchomienie tak zdefiniowanego profilu jest następujące: `ssh nazwa_profilu`. `Hostname` jest zbędny jeżeli nazwa profilu odpowiada nazwie zdalnego systemu

1.7.1 Proxy

Bardzo ciekawą opcją jest możliwość definiowania komend proxy w pliku konfiguracyjnym:

```
Host host3
  ProxyCommand ssh -A -q -e none -W host3:22 host2
```

Wydanie polecenia `ssh host3` spowoduje zestawienie tunelu do host2 i przez ten tunel zostanie zestawione połączenie SSH z host3.

1.7.2 Wildcards w pliku konfiguracyjnym

W pliku konfiguracyjnym istnieje możliwość wykorzystania symboli wieloznacznych. Na przykład można zdefiniować profil dotyczący wszystkich systemów należących do konkretnej domeny:

```
Host *.cs.put.poznan.pl
  User jbond
  Port 55555
```

Wykonanie polecenia `ssh unixlab.cs.put.poznan.pl` spowoduje zastosowanie powyższego profilu.

Możliwe jest połączenie znaków wieloznacznych i komendy proxy w następujący sposób:

```
Host *.cs.put.poznan.pl
  ProxyCommand ssh -q -e none -A -W %h:22 gate3
```

Wykonanie polecenia `ssh unixlab.cs.put.poznan.pl` spowoduje zestawienie tunelu do gate3 i dopiero w tym tunelu zestawienie połączenia do unixlab.cs.put.poznan.pl.

🕮 Literatura dodatkowa:

OpenSSH https://www.ssh.com/ssh/openssh/

PuTTY https://www.ssh.com/ssh/putty/

ssh-keygen https://www.ssh.com/ssh/keygen/

🗒 Problemy do analizy:

- Jakim algorytmem szyfrowany jest plik przechowujący klucz prywatny użytkownika?
- Czy hasło chroniące ten plik (`passphrase`) trzeba podawać przy każdym dostępie do niego?