Bezpieczeństwo systemów informatycznych

ĆWICZENIE SSH/PGP

1. Secure Shell i protokół SSH

1.1 Protokół SSH

Protokół SSH umożliwia bezpieczny dostęp do zdalnego konta. Protokół pozwala na zastosowanie bezpiecznego uwierzytelniania użytkownika i szyfrowanie transmisji. Korzysta z portu TCP o numerze 22. Ponadto możliwe jest też szyfrowane tunelowanie ruchu pomiędzy arbitralnymi portami TCP, co umożliwia tworzenie tuneli wirtualnych. Transmitowane dane mogą podlegać automatycznej kompresji.

1.2 Program ssh

Program ssh zastępuje w działaniu program rlogin i rsh.

```
local> ssh –l username remotehost ls
```

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

1.2.1 Metody uwierzytelniania użytkownika.

Program ssh podejmuje próby uwierzytelnienia użytkownika w następującej kolejności:

1. uwierzytelnianie przez mechanizm zaufania do systemów zdalnych połączony z kryptograficznym uwierzytelnieniem obu systemów metodą klucza asymetrycznego wraz z ustaleniem symetrycznego klucza sesji (metodą Dieffiego-Hellmana);
2. uwierzytelnienie użytkownika kryptograficzną metodą klucza asymetrycznego, typu *challenge-response* (algorytm RSA lub DSA)
3. uwierzytelnianie klasyczne poprzez hasło systemowe użytkownika zdalnego (komunikacja jest jednak szyfrowana, więc hasło nie jest transmitowane tekstem jawnym).

Mechanizm zaufania może być identyczny, jak dla poleceń r∗ (bazujący na plikach /etc/hosts.equiv oraz ~/.rhosts). Jednak, z powodu wysokiego niebezpieczeństwa jego stosowania, protokół SSH stosuje swój własny system zaufania, wykorzystujący pliki, odpowiednio, /usr/local/etc/shosts.equiv oraz ~/.shosts. Mimo tego, stosowanie mechanizmu zaufania nie jest rekomendowane i niektóre implementacje protokołu SSH nie obsługują tego mechanizmu (w szczególności w standardzie SSH2 nie przewiduje się już tego mechanizmu).

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. Mechanizm szyfrowania asymetrycznego i tajność kluczy prywatnych wystarcza do obrony systemu przed atakami typu DNS spoofing, IP spoofing czy routing spoofing.

Kryptograficzne uwierzytelnianie zdalnego użytkownika wykorzystuje zbiór kluczy publicznych zaufanych użytkowników, przechowywanych w pliku ~/.ssh/authorized_keys. Weryfikacja tożsamości metodą *challenge-response*, przy zachowaniu tajności kluczy prywatnych użytkowników, umożliwia bezpieczne uwierzytelnianie.

Ssh dokonuje uwierzytelnienia przy pomocy algorytmu RSA.

1.2.2 Stosowane algorytmy szyfracji

W aktualnej wersji SSH v.2 obsługiwane są następujące symetryczne algorytmy szyfracji komunikacji:

- 3DES (wybierany domyślnie),
- Arcfour – bardzo szybki algorytm, pochodny RC4,

– Blowfish i CAST128 – algorytmy 128-bitowe

a także algorytmy podpisu elektronicznego (do zapewnienia integralności komunikacji):

– HMAC-MD5 oraz HMAC-SHA1.

1.3 Zarządzanie kluczami kryptograficznymi

Do tworzenia pary kluczy kryptograficznych służy program ssh-keygen. Zapisuje on klucz prywatny użytkownika wygenerowany dla algorytmu DSA (lub RSA) w pliku ~/.ssh/id_dsa (odpow. ~/.ssh/id_rsa), a klucz publiczny w pliku ~/.ssh/id_dsa.pub (~/.ssh/id_rsa.pub). Dla dodatkowego zabezpieczenia, klucz prywatny może być zapisany w pliku w postaci zaszyfrowanej, domyślnie algorytmem 3DES. Posługiwanie się wówczas tym kluczem wymaga każdorazowo podania sekwencji traktowanej jako rodzaj hasła.

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

Program ten jest również wykorzystywany do generacji kluczy systemu, prywatnego oraz publicznego, odpowiednio w plikach: /usr/local/etc/ssh_hosts_key i /usr/local/etc/ssh_hosts_key.pub.

1.4 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 localport:server:serviceport gate2
```

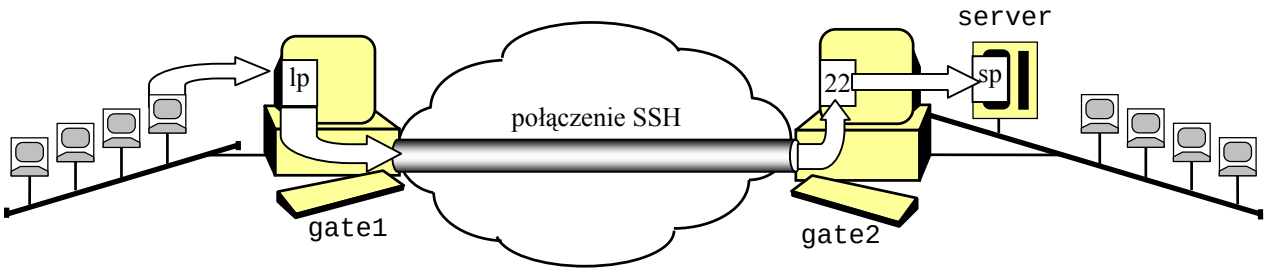
Takie polecenie utworzy tunel pomiędzy bramami `gate1` i `gate2`, który umożliwi komunikację poprzez port `localport` z usługą `serviceport` na komputerze `server` (`-L` = Local port forwarding). Komunikacja jest zaszyfrowana na odcinku `gate1-gate2`.

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

`server` – nazwa (domenowa) lub adres (IP) komputera, którego port udostępniamy poprzez tunel wirtualny (za bramą `gate1`)

`serviceport` – udostępniany zdalnie port TCP na serwerze `server`

Propagowanie połączeń w tunelu przedstawia schematycznie poniższy rysunek.



Identyczny efekt do poprzedniego ma kolejne polecenie:

```
gate2> ssh –R remoteport:server:serviceport gate1
```

(`-R` = Remote port forwarding).

`remoteport` – port TCP zdalnego systemu (bramy `gate1`)

`server` – nazwa (domenowa) lub adres (IP) komputera, którego port udostępniamy poprzez tunel wirtualny (przed bramą `gate2`)

`serviceport` – udostępniany zdalnie port TCP na serwerze `server`

Zadania:

1. Za pomocą narzędzi z pakietu OpenSSH nawiąż połączenie ze wskazanym serwerem (np. gumis).
2. Wykonaj zdalnie w systemie tego serwera polecenie cat /etc/HOSTNAME.
3. Dokonaj skopiowania lokalnego pliku do wybranego podkatalogu na koncie w systemie zdalnym.
4. Wygeneruj parę kluczy kryptograficznych szyfrowania asymetrycznego dla swojego lokalnego konta (bez zabezpieczania hasłem pliku z kluczem prywatnym). Wykorzystaj klucze do uwierzytelniania użytkownika w systemie zdalnym.
5. Zabezpiecz hasłem plik ze swoim kluczem prywatnym.
6. Uruchom lokalne propagowanie połączeń (port forwarding) tylko dla lokalnych połączeń (*loopback*) z portu 8080 na port 80 serwera www (www.cs.put.poznan.pl) w tunelu wirtualnym pomiędzy swoim lokalnym systemem a serwerem unixlab.
7. Uruchom lokalne propagowanie połączeń lokalnych i zdalnych (z sieci lokalnej) w konfiguracji jak powyżej.

2. Bezpieczna poczta elektroniczna PGP

2.1 System GnuPG (Gnu Privacy Guard)

Standardy PGP (RFC 1991) i OpenPGP (RFC 2440) umożliwiają podpisywanie oraz szyfrowanie plików (w tym korespondencji pocztowej) metodami asymetrycznymi i symetrycznymi. W implementacji GnuPG stosowane są szyfry RSA, DSA lub ElGamal dla szyfrowania asymetrycznego oraz 3DES, CAST5, Blowfish, Twofish, i AES (128b, 192b i 256b) dla szyfrowania symetrycznego. Podpis elektroniczny obsługują algorytmy MD5, SHA-1 i SHA-256 oraz RIPEMD-160. Ponadto w implementacji tej możliwe jest dokonywanie kompresji szyfrowanej treści algorytmami ZIP lub ZLIB.

2.2 Zarządzanie kluczami kryptograficznymi

Wszystkie operacje w systemie GnuPG wykonuje uniwersalne narzędzie gpg. Program ten umożliwia użytkownikowi m.in. wygenerowanie pary kluczy asymetrycznych o wybranej długości oraz utrzymuje zbiory („pęki") kluczy publicznych innych użytkowników.

2.2.1 Wygenerowanie klucza

```
local> gpg --gen-key
```

Pęki kluczy przechowywane są domyślnie w katalogu ~/.gnupg w plikach: pubring.gpg oraz secring.gpg. Bieżącą zawartość pęku posiadanych kluczy publicznych można wyświetlić poleceniem :

```
local> gpg –-list-keys
```

lub dla kluczy prywatnych:

```
local> gpg –-list-secret-keys
```

Przykładowy wynik działania pierwszego z tych dwóch poleceń może wyglądać następująco:

```
/home/jbond/.gnupg/pubring.gpg
------------------------------
pub  1024D/3BF84E43 2004-06-25 James Bond <spsk007@cs.put.poznan.pl>
sub  1024g/8B423A22 2004-06-25
```

Wygenerowany klucz publiczny można udostępnić, np. eksportując do pliku tekstowego w formacie ASCII:

```
local> gpg –-export –a –o ~/.gpgkey
```

2.2.2 Pozyskanie czyjegoś klucza publicznego

Pozyskanie klucza prywatnego może nastąpić:

– z pliku przesłanego lub pobranego (np. poprzez WWW) od właściciela klucza:

```
local> gpg –-import plik_z_kluczem
```

– lub z serwera kluczy:

```
local> gpg --keyserver certserver.pgp.com --recv-key 0xBB7576AC
```

2.2.3 Podpisywanie i szyfrowanie wiadomości

Program gpg pozwala podpisać dowolny plik, w szczególności list, swoim kluczem prywatnym:

```
local> gpg –-sign plik
```

W powyższym przykładzie wynik pojawi się w postaci skompresowanej w pliku o nazwie plik.pgp. Aby uzyskać podpis w postaci czytelnej (bez kompresji) należy wykonać polecenie:

```
local> gpg –-clearsign plik
```

Wówczas powstanie plik tekstowy w formacie ACII o nazwie plik.asc. Nazwę pliku wynikowego możemy zmieniać opcją `-o nazwa_pliku`.

Program gpg pozwala też zaszyfrować wiadomość kluczem publicznym konkretnego odbiorcy:

```
local> gpg –-recipient odbiorca –at –o list.txt plik
```

a także połączyć szyfrowanie z podpisem elektronicznym:

```
local> gpg -se –r odbiorca –at –o list.txt plik
```

2.2.4 Deszyfracja i weryfikacja wiadomości

Do deszyfrowania i weryfikowania podpisu służą opcje, odpowiednio, `-d` (`--decrypt`) oraz `--verify`:

```
local> gpg –d list
local> gpg –-verify list
```

2.2.5 Szyfrowanie plików

Do symetrycznego szyfrowania plików służy opcja `-c` programu gpg:

```
local> gpg –c -o szyfrogram plik.txt
```

lub:

```
local> gpg –symmetric --cipher-algo AES256 -o szyfrogram plik.txt
```

🕮 Literatura

strona domowa i dokumentacja projektu GnuPG: http://www.gnupg.org

💻 Zadania:

1. Za pomocą narzędzi pakietu GnuPGP wygeneruj pęki kluczy kryptograficznych szyfrowania asymetrycznego dla swojego konta pocztowego, a następnie udostępnij klucz publiczny poprzez usługę finger.
2. Wykorzystaj pakiet GnuPG do ochrony korespondencji z wybranym użytkownikiem.
3. Dokonaj zaszyfrowania wybranego pliku metodą szyfrowania symetrycznego narzędziami pakietu GnuPGP.