Warsztaty z Sieci komputerowych
Lista 7

1 Uwagi ogólne

Na początku pracy wydaj polecenie `netmode lab`. Na dzisiejszej pracowni karty `eth1` podpięte są do koncentratora, zaś karty `eth0` standardowo do przełącznika. W zadaniu 1 będziemy korzystać z karty `eth1`, zaś we wszystkich następnych z karty `eth0`.

2 Zadania

Zadanie 1. Przypisz karcie `eth1` adres IP równy `10.0.0.`*`nr_komputera`*. Pingając adres rozgłoszeniowy sieci `10.0.0.0/8` sprawdź, czy masz połączenie z pozostałymi komputerami w pracowni.

Dobierzcie się trójkami (będziemy posługiwać się określeniami „osoba A", „osoba B" i „osoba C"). Osoba A powinna zmienić hasło użytkownika `student` poleceniem

```
$> passwd
```

na trudne do odgadnięcia, a następnie przekazać je w sekrecie osobie B. Osoba B za pomocą polecenia

```
$> telnet adres_IP_komputera_A
```

powinna zalogować się na konto `student` na komputerze osoby A. W tym czasie osoba C powinna spróbować podsłuchać Wiresharkiem ruch pomiędzy komputerami osób A i B i odkryć zmienione hasło. Przydatna będzie opcja `Follow TCP stream`, wybierana z menu kontekstowego Wiresharka po kliknięciu jednego z pakietów należących do połączenia `telnet`.

Uwaga: standardowe hasło użytkownika `student` (tj. `cisco`) można przywrócić poleceniem

```
#> restore_passwd
```

Przeprowadźcie eksperyment ponownie zamiast `telnet` używając polecenia

```
$> ssh adres_IP_komputera_A
```

Czy podsłuchanie hasła jest nadal możliwe?

Zdekonfiguruj interfejs `eth1`, aktywuj interfejs `eth0` i skonfiguruj go za pomocą protokołu DHCP poleceniami

```
#> ifdown eth1
#> ifup eth0
```

Zadanie 2. W tym zadaniu skonfigurujemy `ssh` tak aby możliwe było łączenie się bez hasła. Najpierw poleceniem

```
$> ssh-keygen
```

wygeneruj klucz publiczny i prywatny. Zapisz je w domyślnych plikach (`.ssh/id_rsa.pub` oraz `.ssh/id_rsa`). Hasło zabezpieczające klucz pozostaw puste.[1] Obejrzyj właśnie wygenerowane pliki z kluczami.

Teraz wystarczy dopisać klucz publiczny do pliku `.ssh/authorized_keys` na serwerze SSH, z którym będziemy się łączyć. Tym serwerem będzie komputer sąsiada, niech A oznacza jego adres IP. Najpierw skopiuj klucz na komputer A poleceniem

```
$> scp .ssh/id_rsa.pub A:plik_docelowy
```

Następnie używając SSH zaloguj się na ten komputer

```
$> ssh A
```

jako hasło podając `cisco`. Na komputerze A dopisz skopiowany właśnie klucz publiczny do pliku `.ssh/authorized_keys`, a następnie wyloguj się:

```
$> cat plik_docelowy >> .ssh/authorized_keys
$> rm plik_docelowy
$> exit
```

Sprawdź, czy działania odniosły skutek, tj. czy możesz zalogować się teraz na komputer A bez podawania hasła. Polecenie

```
$> ssh -v A
```

wyświetli kolejne etapy nawiązywania połączenia.

Zadanie 3. W tym zadaniu wykonamy wysyłanie wiadomości przez SMTP, ale w połączeniu szyfrowanym. Skonfiguruj wybrany program pocztowy (dostępne powinny być KMail i Evolution) do korzystania z adresu pocztowego `ccna`i`@example.com`, gdzie i jest numerem Twojego komputera. W Evolution możesz skorzystać z uruchamianego na początku programu kreatora ustawień, gdzie jako serwer poczty przychodzącej ustaw `POP` o adresie `eagle-server.example.com`, zaś jako serwer poczty wychodzącej `SMTP` o takim samym adresie. W programie Evolution włącz szyfrowanie w ustawieniach programu (*Edit* | *Preferences*, karta *Mail Accounts* | *Sending Email*) wybierając *SSL Encryption*). Uruchom ponownie program Evolution, inaczej zmiany nie odniosą skutku. Przy próbie wysyłania poczty będziemy musieli zaakceptować certyfikat serwera.

Włącz Wiresharka nasłuchującego na interfejsie `eth0`. W programie Evolution kliknij przycisk *New*, napisz i wyślij testowy email do samego siebie. Spróbuj przeczytać zawartość komunikacji (opcja *Follow TCP Stream* Wiresharka).

Następnie nawiąż szyfrowane połączenie z serwerem SMTP poleceniem

[1] W praktyce pozostawianie klucza prywatnego niezabezpieczonego hasłem to zazwyczaj zły pomysł.

```
$> openssl s_client -quiet -connect eagle-server.example.com:465
```

i wyślij maila posługując się poleceniami protokołu SMTP (`MAIL FROM`, `RCPT TO` i `DATA`).

Zadanie 4. Innym sposobem na uzyskanie szyfrowanego połączenia z serwerem SMTP jest stworzenie tunelu SSH do serwera. Utwórz tunel SSH łączący port 2525 lokalnego komputera z portem 25 serwera `eagle-server.example.com` poleceniem

```
$> ssh -f -N -L 2525:localhost:25 ccnai@eagle-server.example.com
```

gdzie i jest numerem Twojego komputera. Sprawdź, jaka usługa odpowiada po drugiej stronie jeśli wpiszesz polecenie

```
$> telnet localhost 2525
```

W Wiresharku sprawdź, co jest przesyłane. Zmień konfigurację programu pocztowego wyłączając dla serwera SMTP szyfrowanie i zmieniając jego adres na `localhost` i port 2525. Sprawdź ustawienia wysyłając testowy email i podglądając pakiety w Wiresharku.

Zadanie 5. W tym zadaniu zapoznamy się z programem `gpg` będącym wolną implementacją standardu OpenPGP. Poleceniem

```
$> gpg --gen-key
```

utwórz parę kluczy PGP: publiczny i prywatny. Wybierz wartości domyślne poza: (i) swoimi danymi, (ii) adresem email (wpisz `ccnai@example.com`) oraz (iii) sensownym hasłem zabezpieczającym klucz prywatny.

Posiadane klucze (odpowiednio prywatne i publiczne) można wyświetlić poleceniami

```
$> gpg --list-secret-keys
$> gpg --list-keys
```

Na razie będą tam widoczne tylko Twoje klucze. Zapisz swój klucz publiczny w czytelnej postaci do pliku `klucz-pgp` poleceniem

```
$> gpg -a --export identyfikator_klucza > klucz-pgp
```

Wyślij powyższy plik sąsiadowi. Otrzymany od sąsiada klucz zapisz w pliku `klucz-pgp-sasiada`. Zaimportuj go do programu `gpg` poleceniem

```
$> gpg --import < klucz-pgp-sasiada
```

Ponownie wyświetl listę posiadanych kluczy:

```
$> gpg --list-secret-keys
$> gpg --list-keys
```

W pliku `wiadomosc` umieść tajną treść. Podpisanie wiadomości swoim kluczem prywatnym i zaszyfrowanie kluczem publicznym sąsiada następi po wydaniu polecenia

```
$> gpg -a -r odbiorca -se wiadomosc
```

gdzie *odbiorca* jest ciągiem umożliwiającym zidentyfikowanie klucza odbiorcy (np. imię odbiorcy lub identyfikator jego klucza). Program ostrzeże nas, że nie mamy zaufania do klucza, który właśnie wykorzystujemy. (Dostaliśmy go pocztą, ale czy jesteśmy pewni, że nadawcą był sąsiad a nie ktoś podszywający się pod sąsiada?) Aby to naprawić, przerwijmy szyfrowanie wciskając `Ctrl+C` i wydajmy polecenie

```
$> gpg --edit-key identyfikator_klucza_sasiada
```

Po znaku zachęty wpisz polecenie

```
> fpr
```

co wyświetli skrót dla posiadanego klucza publicznego sąsiada. Jeśli sąsiad ma taki sam skrót dla swojego klucza (sprawdźcie to, tj. niech sąsiad wejdzie w tryb edycji swojego klucza i też wywoła polecenie `fpr`!), to można bezpiecznie założyć, że posiadany klucz faktycznie do niego należy i podpisać go poleceniem

```
> sign
```

a następnie opuścić tryb edycji poleceniem

```
> quit
```

Następnie ponownie wydaj polecenie

```
$> gpg -a -r odbiorca -se wiadomosc
```

Szyfrogram zostanie zapisany do pliku `wiadomosc.asc`, który należy wysłać e-mailem sąsiadowi. Otrzymaną od sąsiada wiadomość zapisz do pliku `otrzymane`. Następnie odszyfruj ją swoim kluczem prywatnym i zweryfikuj prawdziwość podpisu sąsiada poleceniem

```
$> gpg -d otrzymane
```

Lista i materiały znajdują się pod adresem http://www.ii.uni.wroc.pl/~mbi/dyd/

Marcin Bieńkowski