Instrukcja do zajęć laboratoryjnych

Język ANSI C (w systemie LINUX)

wersja: 1.00

Nr ćwiczenia:	Dodatkowe - SSH	
Temat:	**Praca z protokołem SSH**	
Cel ćwiczenia:	Celem ćwiczenia jest zapoznanie studenta z działaniem podstawowego dla bezpieczeństwa pracy protokołem SSH (ang. Secure SHell).	
Wymagane przygotowanie teoretyczne:	Podstawy pracy w środowisku LINUX	
Sposób zaliczenia:	Sprawozdanie w formie pisemnej.	[ ]
	Pozytywna ocena ćwiczenia przez prowadzącego pod koniec zajęć.	[x]

1. Konwencje przyjęte w instrukcji

`Czcionka o stałej szerokości`

Nazwy programów, poleceń, katalogów, wyniki działania wydawanych poleceń.

`Czcionka o stałej szerokości pogrubiona`

Podaje tekst, który należy dosłownie przepisać. W przypadku plików źródłowych wyróżnia istotniejsze fragmenty.

`Czcionka o stałej szerokości kursywą`

Tekst komentarza w przykładowych sesjach przy terminalu.

`Czcionka o stałej szerokości kursywą pogrubiona`

Wyróżnia istotniejsze fragmenty wyników działania wydawanych poleceń.

2. Uwagi wstępne

W obecnych czasach, gdy sieci komputerowe (a za tym i Internet) bardzo rozrosły się i stały się dostępne dla każdego, zasadniczego znaczenia nabrały sprawy związane z bezpiecznym przesyłaniem danych między poszczególnymi komputerami. Mamy tutaj na myśli głównie zagadnienia związane z takim przesyłaniem danych, aby osoby nieautoryzowane nie miały do nich

(łatwego) dostępu. Przesyłane dane będą więc bezpieczne, gdy będzie trudno je podsłuchać. A to z kolei osiągniemy, gdy cała transmisja będzie zakodowana i, co bardzo ważne, będzie zakodowana z użyciem algorytmów i metod, które są dostatecznie „silne" kryptograficznie. Powszechnie stosowaną w takich sytuacjach metodą jest przesyłanie wszystkich danych z wykorzystaniem protokołu SSH (ang. *Secure Shell*).

3. Protokół SSH – krótki opis działania

Protokół `ssh` (ang. *Secure Shell*) jest jednym z tzw. *protokołów zdalnej sesji*. Oznacza to, że program korzystający z tego protokołu (często również o nazwie zawierającej skrót "`ssh`") umożliwia komunikację ze zdalnym komputerem. Dzięki kodowaniu przesyłanych danych jest on uważany za program bardzo bezpieczny. Obecnie wypiera on używane wcześniej takie protokoły jak `telnet`, `rlogin`, `rcp`, `rsh`, które ze względu na to, że przesyłają dane w jawnej postaci (bez kodowania) mogą być źródłem luk w systemie bezpieczeństwa. Innym rozszerzeniem oprogramowania realizującego połączenia `ssh` jest protokół `scp` umożliwiający bezpieczne przesyłanie plików. Powoli wypiera on opracowany w początkach istnienia sieci Internet protokół `ftp`, którego podstawową wadą była możliwość podsłuchania zarówno hasła jak i zawartości transmisji.

Przy pomocy `ssh` można więc poprzez sieć Internet zalogować się na się na odległym serwerze i pracować na nim tak, jak przy pomocy fizycznie podłączonego doń terminala. Inaczej jednak niż w przypadku protokołu `telnet`, protokół `ssh` zapewnia szyfrowanie całej transmisji (łącznie z nazwą konta oraz hasłem, transmitowanym podczas sekwencji logowania się na serwerze). Protokół ten, udostępniony początkowo w wersji 1, szybko ewoluował i aktualnie częściej używana jest jego 2. wersja. Oczywiście, ze względu na liczne udoskonalenia włączone do wersji 2. protokołu, zalecane jest posługiwanie się tą właśnie wersją.

Zasada działania protokołu `ssh` opiera się na kryptograficznej technologii **RSA** i jest następująca: każdy z komputerów, na którym zainstalowane jest oprogramowanie `ssh` posiada parę kluczy: tzw. **klucz prywatny** dostępny tylko dla administratora komputera oraz **klucz publiczny** dostępny dla wszystkich użytkowników sieci. Klucze te są tak zbudowane, że informację zaszyfrowaną kluczem prywatnym można rozszyfrować **tylko** przy pomocy klucza publicznego i odwrotnie, informację zaszyfrowaną kluczem publicznym można rozszyfrować **tylko** przy pomocy klucza prywatnego. Klucze są więc ze sobą powiązane, ale żadnego z nich nie można odtworzyć na podstawie znajomości drugiego (mówiąc dokładniej: teoretycznie można, ale jest to bardzo czasochłonna operacja – współczesne komputery wymagają bardzo długich czasów obliczeń, które powodują, że w praktyce złamanie takich kluczy jest niemożliwe).

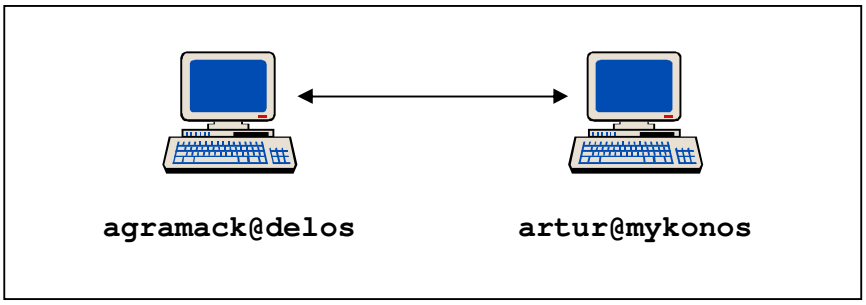
Komunikacja rozpoczyna się od **autoryzacji komputerów**. Każdy z komputerów, na których jest zainstalowany pakiet `ssh`, posiada parę kluczy - **klucz prywatny**, możliwy do odczytania jedynie przez administratora danego komputera oraz przez demona `sshd`, obsługującego usługę `ssh` i **klucz publiczny** dostępny dla wszystkich użytkowników sieci. Połączenie jest realizowane po stronie klienta przez program `ssh` (*SSH Client*), a po stronie serwera przez demona `sshd` (*SSH Server*). Kiedy klient łącząc się z demonem (serwerem) jako pierwsze dane otrzymuje klucz publiczny serwera. Klient porównuje ten klucz z kluczem zachowanym w wewnętrznej bazie danych, z poprzednich połączeń (o ile wcześniej łączył się z tym serwerem, jeżeli nie, to tylko zapamiętuje go w bazie danych). **W przypadku stwierdzenia niezgodności kluczy połączenie jest zamykane.** W przypadku zgodności kluczy klient generuje losową liczbę 256 bitową. Liczbę tę szyfruje przy pomocy swojego klucza prywatnego oraz klucza publicznego serwera i tak zakodowaną przesyła do serwera. Serwer przy pomocy swojego klucza prywatnego i klucza publicznego klienta

rozkodowuje przesyłkę, czyli wygenerowaną przez klienta losowo liczbę. Liczba ta stanowi klucz używany do kodowania podczas dalszej komunikacji.

Po dokonaniu **autoryzacji komputerów** następuje **autoryzacja użytkownika**. Najpierw sprawdza się istnienie klucza publicznego klienta w globalnej bazie danych kluczy publicznych innych serwerów (plik `/etc/ssh_known_hosts`) lub w indywidualnej bazie danego użytkownika (plik `~/.ssh/known_hosts`). W przypadku znalezienia tego klucza, demon zezwala na dokonanie autoryzacji na podstawie plików `/etc/hosts.equiv`, `/etc/shosts.equiv`, `~/.rhosts` lub `~/.shosts`. Pliki `/etc/shosts.equiv` i `~/.shosts` stanowią odpowiedniki plików `/etc/hosts.equiv` oraz `~/.rhosts` ale wyłącznie dla usługi `ssh`, a więc stanowią znacznie lepszą metodę autoryzacji. W przypadku **niepowodzenia autoryzacji**, sprawdzana jest obecność pliku `~/.ssh/authorized_keys` zawierającego klucze publiczne danego użytkownika wygenerowane na innych stacjach. Plik ten może sobie stworzyć każdy użytkownik indywidualnie przy pomocy polecenia `ssh-keygen` na stacji klienta i poprzez przesłanie go na serwer (na przykład przy pomocy zwykłego `ftp` – klucz publiczny nie musi być chroniony, więc kodowanie transmisji nie jest konieczne). Następnie serwer próbuje dokonać autoryzacji użytkownika w sposób analogiczny do przeprowadzonej wcześniej autoryzacji komputerów, tzn. wymiany z klientem zakodowanej informacji przy pomocy pary kluczy: klucz publiczny użytkownika, klucz prywatny serwera i klucz prywatny użytkownika, klucz publiczny serwera. W przypadku niepowodzenia tej metody, demon pyta się użytkownika o jego hasło. Ponieważ transmisja jest kodowana, nie ma obawy o podsłuchanie tego hasła przez niepowołaną osobę.

4. Protokół SSH – kilka ćwiczeń

Każdy student ma do dyspozycji 2 konta na 2 różnych maszynach z zainstalowanym systemem LINUX. Parametry tych kont zostaną podane na początku zajęć przez prowadzącego. W poniższej instrukcji zakładamy, że mamy do dyspozycji dwa komputery o nazwach **`delos.iie.uz.zgora.pl`** oraz **`mykonos.iie.uz.zgora.pl`**. Ponadto na komputerze **`delos`** jest stworzony użytkownik **`agramack`** natomiast na komputerze **`mykonos`** użytkownik **`artur`**. Obrazuje to poniższy rysunek.



Ćwiczenie 1: Komputer `mykonos` po raz pierwszy łączy się z komputerem `delos`

Na komputerze `delos` chcemy wykonać **zdalnie** polecenie `date`. Ponieważ łączymy się po raz pierwszy z komputerem `delos`, więc w katalogu domowym użytkownika `artur` na komputerze `mykonos` tworzony jest katalog `.ssh` (z przodu nazwy jest kropka, gdyż jest to katalog ukryty) a w nim tworzony jest plik `known_hosts`, zawierający klucz publiczny komputera `delos`.

Używamy opcji `-l`, aby jawnie wskazać konto na maszynie zdalnej. Zanim na zdalnej maszynie wykonane zostanie polecenie zostaniemy poproszeni o podanie hasła użytkownika `agramack`.

```
artur@mykonos:~$ ssh -l agramack delos.iie.uz.zgora.pl date
The authenticity of host 'delos.iie.uz.zgora.pl (192.168.21.75)' can't be
established.
RSA key fingerprint is cc:27:e5:4e:78:12:c5:73:89:96:6d:f1:4d:33:10:b3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'delos.iie.uz.zgora.pl,192.168.21.75' (RSA) to
the list of known hosts.
agramack@delos.iie.uz.zgora.pl's password:
Fri Dec 12 10:05:14 CET 2003
```

Ćwiczenie 2: Komputer `mykonos` **po raz kolejny łączy się z komputer** `delos`

Tym razem w katalogu `~/.ssh` już jest plik `known_hosts`, więc nie pojawia się pytanie o jego utworzenie.

```
artur@mykonos:~$ ssh -l agramack delos.iie.uz.zgora.pl date
agramack@delos.iie.uz.zgora.pl's password:
Fri Nov 28 15:22:56 CET 2003
```

Gdy nie podamy komendy do wykonania po prostu połączymy się z serwerem `delos` i będziemy mogli pracować na nim zdalnie.

```
artur@mykonos:~$ ssh -l agramack delos.iie.uz.zgora.pl
agramack@delos.iie.uz.zgora.pl's password:
Linux delos 2.4.22 #1 SMP pon lis 3 15:44:22 CET 2003 i586 unknown

*********************************************
*            Witamy na serwerze             *
*           delos.iie.uz.zgora.pl           *
*             Debian GNU/Linux              *
*                                           *
*         Uniwersytet Zielonogorski         *
*     Instytut Informatyki i Elektroniki    *
*                                           *
*             Administrator:                *
*        a.gramacki@iie.uz.zgora.pl         *
*********************************************

You have new mail.
Last login: Sat Nov 29 11:40:46 2003 from mykonos.iie.uz.zgora.pl
agramack@delos:~$ ls
GNUstep          bashrc.stub  pvm3      pvm_hosts    test_suite
agramack@delos:~$ logout
Connection to delos.iie.uz.zgora.pl closed.
```

Oglądamy zawartość utworzonego pliku `known_hosts`. Zawiera on klucz publiczny serwera `delos` (gdy w przyszłości będziemy łączyli się z innymi komputerami do tego pliku będą dopisywane ich klucze publiczne).

Uwaga: podane w dalszej części instrukcji klucze należy traktować jako przykładowe. W czasie wykonywania ćwiczenia będą one najprawdopodobniej inne.

```
artur@mykonos:~/.ssh$ cat known_hosts
delos.iie.uz.zgora.pl,192.168.21.75 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAurHKD+PrYv3rn2py40t3DRcf9V1RNU2Ds5Hn1oLq/xIMah
q39KkvGdS4dVPT4i9ho21WcLBCrlQzUCxgkc4OLXs0+vHhUAYElRtSp3WK42dSjvlBQqovTrol
egh/42+RSdbuxzEz2Ry80ZzK6vuP3+q//THS95UKe6NOoyOQXIU=
```

Robimy eksperyment: ręcznie zmieniamy (w jakimkolwiek edytorze) zawartość pliku `known_hosts`. W oryginalnym kluczu jako trzeci znak od końca występuje duża litera `X`. My

zmieniamy ją na małe x. System stwierdza niezgodność klucza na serwerze delos z tym, zapisanym w pliku known_hosts. Generowane jest stosowne ostrzeżenie. System odmawia wykonania polecenia.

Powodem pojawienia się takiego komunikatu może być np. to, że administrator komputera delos zmienił (wygenerował od nowa) klucz publiczny. Nie można jednak wykluczyć, że zmiana klucza jest efektem np. ataku hakera na komputer delos lub mykonos. Program ssh ostrzega nas tylko o niezgodności kluczy, nie wyrokując co jest tego przyczyną!

```
Było: ...OQXIU=     Jest: ...OQxIU=

artur@mykonos:~/.ssh$ ssh -l agramack delos.iie.uz.zgora.pl date
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle
attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
cc:27:e5:4e:78:12:c5:73:89:96:6d:f1:4d:33:10:b3.
Please contact your system administrator.
Add correct host key in /home/artur/.ssh/known_hosts to get rid of this
message.
Offending key in /home/artur/.ssh/known_hosts:1
RSA host key for delos.iie.uz.zgora.pl has changed and you have requested
strict checking.
Host key verification failed.
```

Ćwiczenie 3: Nie używamy opcji -l

Na komputerze delos próbujemy wykonać zdalnie polecenie date. Ponieważ na komputerze mykonos jesteśmy zalogowani jako użytkownik artur, system domyślnie zakłada, że na zdalnym komputerze będziemy wykonywać polecenie również na koncie artur. Ponieważ na komputerze delos nie ma konta artur pojawiają się poniższe komunikaty.

```
artur@mykonos:~$ ssh delos.iie.uz.zgora.pl date
artur@delos.iie.uz.zgora.pl's password:
Permission denied, please try again.
artur@delos.iie.uz.zgora.pl's password:
Permission denied, please try again.
artur@delos.iie.uz.zgora.pl's password:
Permission denied (publickey,password,keyboard-interactive).
artur@mykonos:~$
```

Ćwiczenie 4: Tworzymy własne klucze: prywatny i publiczny

Jako użytkownik artur na serwerze mykonos tworzymy swoją własną parę kluczy: publiczny i prywatny. Wybieramy rodzaj klucza rsa (inny dostępny to dsa. Patrz dokumentacja man ssh).

```
artur@mykonos:~/.ssh$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/artur/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/artur/.ssh/id_rsa.
```

```
Your public key has been saved in /home/artur/.ssh/id_rsa.pub.
The key fingerprint is:
62:82:5d:60:60:d3:ac:7c:4d:b5:1c:63:e0:b8:05:d5 artur@mykonos
artur@mykonos:~/.ssh$ ls -l
total 12
-rw-------    1 artur    students      887 Nov 28 15:01 id_rsa
-rw-r--r--    1 artur    students      223 Nov 28 15:01 id_rsa.pub
-rw-r--r--    1 artur    students      244 Nov 28 14:30 known_hosts
```

Oglądamy zawartość wygenerowanego klucza publicznego.

```
artur@mykonos:~/.ssh$ cat id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEA4ptL+KYuekqS590/wMSV68Ufwl8jjFjGjorECMTww6aUcq
agfBwksvQmnzmDuXhA2JYFWBC4ISMnR0f7jICJmCQF708T5eiB77DHrgp9yaNOXABOWCkJ7779
m3nrFKoNJTIZaB5spP3FAoKc2S0ztdyFPqtVpkQ9/3ugRBj6Abk= artur@mykonos
```

Ćwiczenie 5: Komputerowi `delos` przekazujemy nasz klucz publiczny

Wygenerowany na komputerze `mykonos` klucz publiczny **przekazujemy** na komputer `delos` (na przykład przy pomocy zwykłego `ftp` – klucz publiczny nie musi być chroniony, więc kodowanie transmisji nie jest konieczne). Klucz ten **musi** być zapisany w katalogu `~/.ssh` w pliku o nazwie `authorized_keys`.

```
agramack@delos:~/.ssh$ cat authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEA4ptL+KYuekqS590/wMSV68Ufwl8jjFjGjorECMTww6aUcq
agfBwksvQmnzmDuXhA2JYFWBC4ISMnR0f7jICJmCQF708T5eiB77DHrgp9yaNOXABOWCkJ7779
m3nrFKoNJTIZaB5spP3FAoKc2S0ztdyFPqtVpkQ9/3ugRBj6Abk= artur@mykonos
```

Teraz łącząc się z `delos` nie musimy już podawać hasła użytkownika `agramack`.

```
artur@mykonos:~/.ssh$ ssh -l agramack delos.iie.uz.zgora.pl date
Sat Nov 29 11:38:49 CET 2003
```

Aby proces autoryzacji mógł przebiegać prawidłowo na komputerze `mykonos` **musi** istnieć i być poprawny klucz prywatny. Plik z tym kluczem **musi** mieć nazwę `id_rsa`. Gdy jeden z tych warunków nie jest spełniony protokół `ssh` odmawia automatycznej autoryzacji – jesteśmy proszeni o podanie hasła.

```
artur@mykonos:~/.ssh$ mv id_rsa id_rsa_OLD
artur@mykonos:~/.ssh$ ssh -l agramack delos.iie.uz.zgora.pl
agramack@delos.iie.uz.zgora.pl's password:
```

Klucz prywatny będzie uważany za błędny również wówczas, gdy jego prawa dostępu będą większe niż `-rw-------`. W takiej sytuacji program `ssh` generuje ostrzeżenie.

```
artur@mykonos:~/.ssh$ ls -l id_rsa
-rw-------    1 artur    students      887 Nov 28 15:13 id_rsa
artur@mykonos:~/.ssh$ chmod 640 id_rsa
artur@mykonos:~/.ssh$ ls -l id_rsa
-rw-r-----    1 artur    students      887 Nov 28 15:13 id_rsa
artur@mykonos:~/.ssh$ ssh -l agramack delos.iie.uz.zgora.pl date
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0640 for '/home/artur/.ssh/id_rsa' are too open.
It is recommended that your private key files are NOT accessible by
others.
This private key will be ignored.
bad permissions: ignore key: /home/artur/.ssh/id_rsa
```

```
Enter passphrase for key '/home/artur/.ssh/id_rsa':
agramack@delos.iie.uz.zgora.pl's password:
```

Ćwiczenie 6: Polecenia `rsh`, `rcp`, `rlogin`

W wielu instalacjach LINUX-a z powodów bezpieczeństwa dawniej używane polecenia `rsh`, `rcp` oraz `rlogin` są wyłączone i stworzone są symboliczne dowiązania do polecenia `/usr/bin/ssh`, które wspiera transmisję kodowaną protokołem *ssh*. Można się o tym przekonać wydając polecenia:

```
artur@mykonos:/etc/alternatives# cd /etc/alternatives/
artur@mykonos:/etc/alternatives# ls -l rlogin rsh rcp
lrwxrwxrwx    1 root     root   12 Sep 11 12:26 rcp -> /usr/bin/ssh
lrwxrwxrwx    1 root     root   12 Sep 11 12:26 rlogin -> /usr/bin/ssh
lrwxrwxrwx    1 root     root   12 Sep 11 12:26 rsh -> /usr/bin/ssh
artur@mykonos:/etc/alternatives#
```