Praca zdalna na serwerze Linux

Mając już skonfigurowane interfejsy sieciowe można rozpocząć pracę zdalną z serwerem opartym o system Linux. W zasadzie, poza instalacją oraz podstawową konfiguracją (którą odbyliśmy w ramach poprzedniego materiału) jest to jedyna forma pracy z serwerowym systemem Linux.

W przeciwieństwie do systemu Windows Server główną formą pracy jest konsola. Poza niedogodnością, jaką stanowi wpisywanie poleceń (chociaż przy dłuższej pracy okazuje się to zaletą), forma ta ma same zalety. Po pierwsze nie obciąża zasobów sprzętowych systemu – jedyne co system musi robić to kontrolować znaki (przeważnie wielkości 1 bajta). Po drugie przesyłanie tekstu poprzez sieć jest znacznie mniej obciążające dla łącz internetowych – nawet na słabym połączenie (GPRS) praca wydaje się być komfortowa (chociaż to jeszcze zależy od jakości samego połączenia). Poza tym serwer może w ogóle nie posiadać karty graficznej – konsola jako taka z niej nie korzysta, przez co możemy zaoszczędzić na rachunkach za prąd (jednak w razie poważnej awarii karta przyda się by wyświetlić konsolę na monitorze).

Pierwotnie pracę zdalną zapewniała usługa telnet. Polegała ona dosłownie na przesyłaniu komend jako jawnych znaków tekstowych. Do pracy przez sieć usługa ta okazała się przez to zupełnie nie do wykorzystania – każdy mógł podsłuchać przesyłane przez nią hasła, poświadczenia itp.

Dlatego szybko opracowano nową metodę przesyłania znakowego – SSH (Secure SHell). Shell (powłoka) w systemach Unix/Linux to w zasadzie podstawowe narzędzie komunikacji użytkowników z jądrem systemu. Ponieważ w sieci można znaleźć wiele stron i artykułów na temat powłoki Unix/Linux nie będzie ona tutaj dodatkowo opisywana (w źródłach jest odnośnik do jednej ze stron).

Istotą jest natomiast działanie samego SSH. Praca zdalna możliwa jest tylko w przypadku posiadania zainstalowanego na serwera SSH – demona sshd (w Windows Server zwykło się to nazywać usługą). Demon ten nasłuchuje domyślnie na porcie 22, działa w oparciu o architekturę serwer-klient po protokole TCP. Nowe wersje (wersja 2) całą transmisję szyfruje przy użyciu algorytmu AES, chociaż można wybrać także Blowfish lub DES. Sama autoryzacja użytkowników może odbywać się standardowo poprzez hasło, klucze DSA, RSA lub protokół Kerberos.

Instalacja OpenSSH

W systemie wykorzystamy otwartą implementację SSH (istnieje także zamknięta, płatna wersja na stronie ssh.com). Najpierw sprawdzamy czy w systemie nie ma już zainstalowanego ssh (przeważnie usługa instalowana jest jako podstawowa):

ps ax | grep ssh

Jeżeli wynikiem będzie takie wyjście:

```
root@server2:/home/test# ps ax | grep ssh
 3387 tty1     S+     0:00 grep ssh
```

To system nie posiada zainstalowanego serwera SSH (nie ma uruchomionego procesu).

Instalujemy go poprzez:

apt-get install ssh

Gdy po instalacji jeszcze raz odpytamy o usługę otrzymamy komunikat:

```
root@server2:/home/test# ps ax | grep ssh
 3764 ?        Ss     0:00 /usr/sbin/sshd
 3793 tty1     S+     0:00 grep ssh
root@server2:/home/test# _
```

I to wszystko – teraz możemy się łączyć do naszego serwera. Z innego systemu Unix/Linux połączenie jest bardzo proste – wystarczy wydać polecenie:

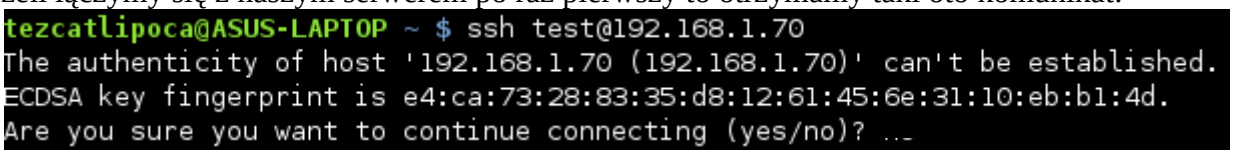
ssh <nazwa_uzytkownika>@<adres_serwera>

gdzie <nazwa_uzytkownika> to po prostu nazwa konta w systemie, z którego chcemy korzystać (np. test), a <adres_serwera> to adres sieciowy serwera (może to być również nazwa domenowa, chociaż najczęściej używa się adresu IP).

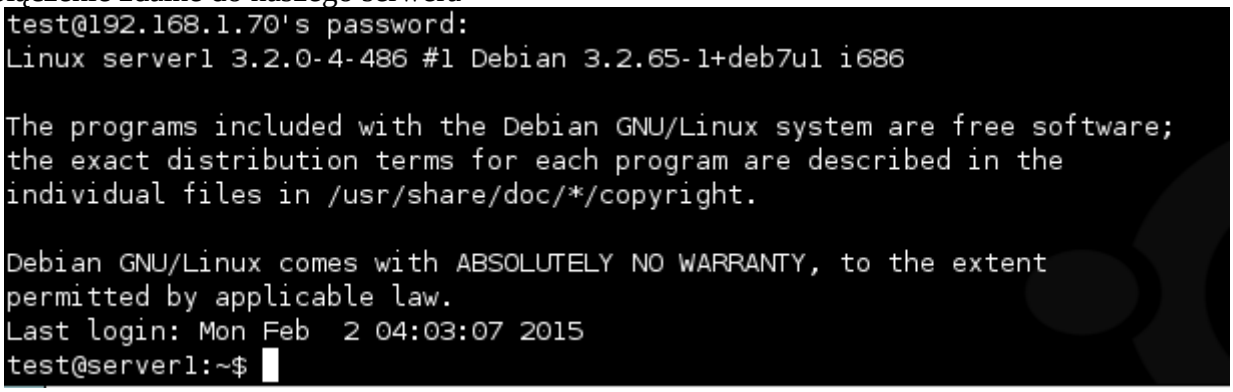
W naszym przypadku połączenia dokonalibyśmy poprzez komendę (zakładając, że posiada on adres 192.168.1.70):

ssh test@192.168.1.70

Jeżeli łączymy się z naszym serwerem po raz pierwszy to otrzymamy taki oto komunikat:



W celu kontynuacji połączenia musimy oczywiście odpowiedzieć twierdząco (yes). Zostaniemy poproszeni o podanie hasła. Jeżeli hasło zgodzi się z tym przypisanym do konta – otrzymujemy połączenie zdalne do naszego serwera



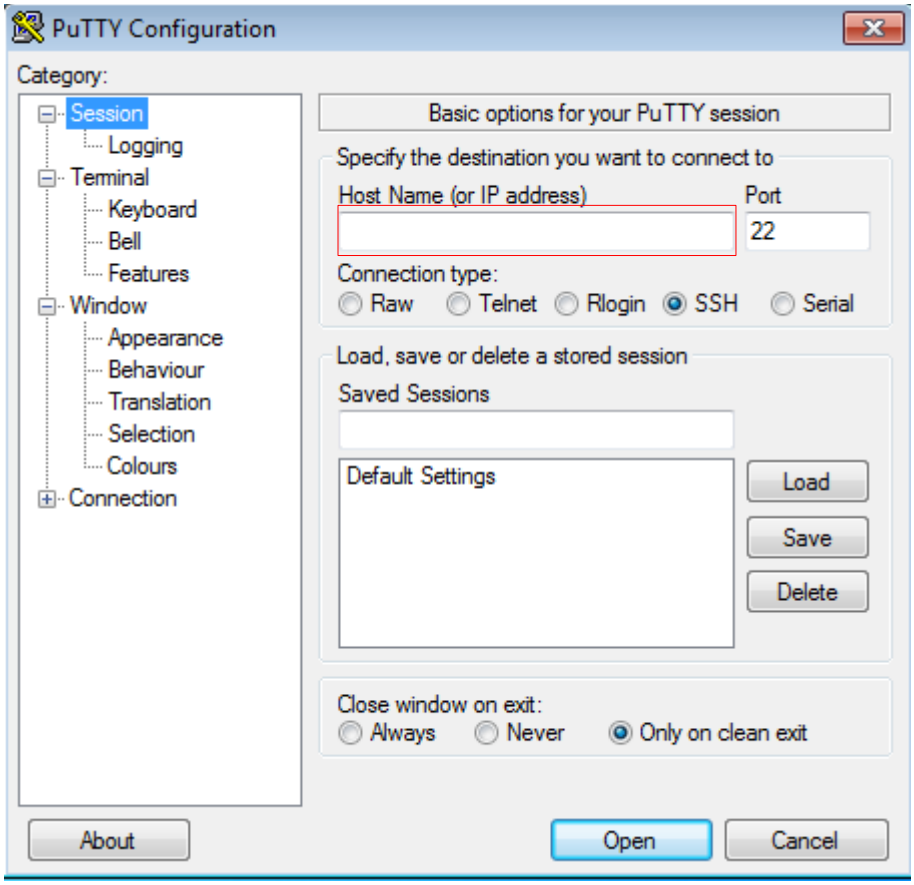
Jak widać, zmienia się znak zachęty na taki, jaki mielibyśmy będąc zalogowanymi na serwerze. Od teraz możemy pracować tak jakbyśmy pracowali bezpośrednio będąc przy nim.

INFORMACJA: Linux, w przeciwieństwie do systemu Windows Server, pozwala na utrzymywanie wielu sesji w ramach jednego konta. Oznacza to, że możemy zalogować się do systemu poprzez jedną konsolę, rozpocząć jakąś operację, po czym otworzyć następną konsolę, zalogować się do systemu i... pracować na obu konsolach. Przydatność tego podejścia widoczne jest szczególnie w przypadku, gdy musimy śledzić wykonywanie się jednego procesu lecz w międzyczasie chcielibyśmy wykonywać inne zadanie.

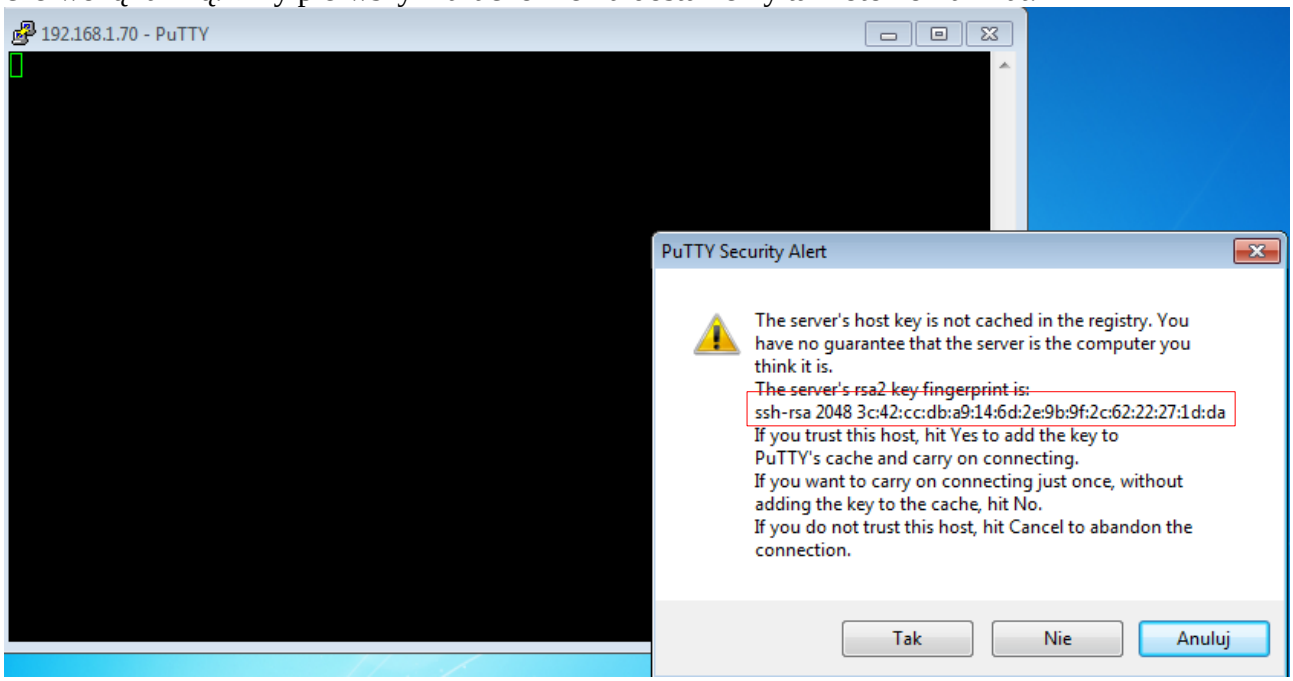
Jeżeli chcemy pracować z systemu Windows to musimy zaopatrzyć się w dodatkowy program (malutką aplikację) o nazwie PuTTY (albo inną, np. Msys czy Cygwin). PuTTY ma jednak tę zaletę, że jest „nieinwazyjny” dla systemu – nie trzeba nic instalować ani nic wstępnie konfigurować!

Program pobieramy ze strony http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html (oficjalna strona programisty tegoż narzędzia).

Po uruchomieniu program wygląda tak:



Jeżeli chcemy się po prostu jednorazowo połączyć wpisujemy nazwę hosta w pole oznaczone czerwoną ramką. Przy pierwszym uruchomieniu dostaniemy taki oto komunikat:

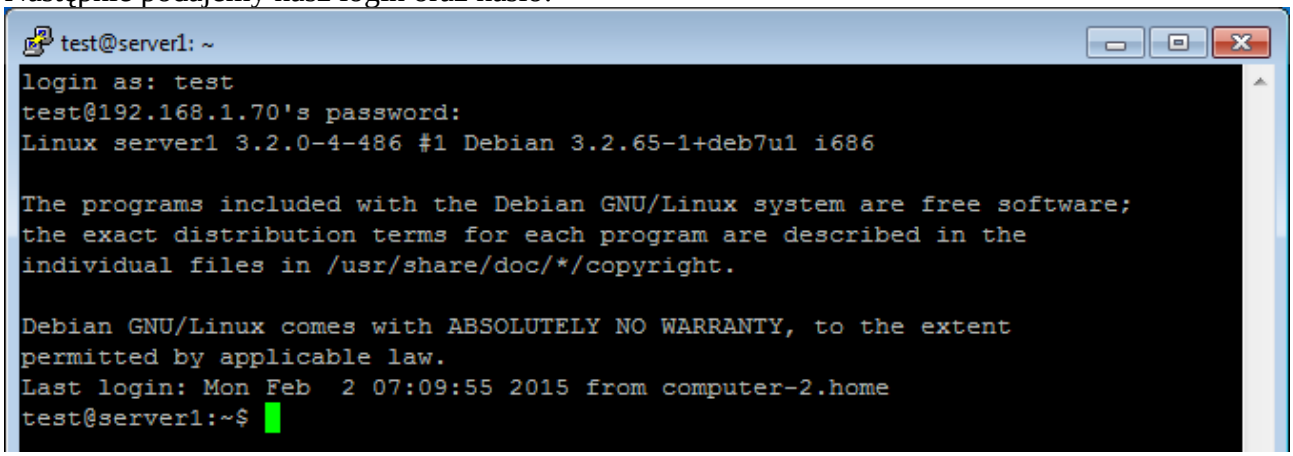


By się połączyć musimy oczywiście na to pytanie odpowiedzieć twierdząco.

INFORMACJA: Pytanie to ZAWSZE będzie nam towarzyszyć w przypadku, gdy łączymy się do danego komputera/serwera po raz pierwszy. Ostrzega nas ono, że przekazane od serwera ID nie zgadza się naszą lokalną bazą danych ID serwera (ID w tym przypadku to odcisk palca – fingerprint) – to to zaznaczone w czerwonej ramce. Praktyka ta stanowi zabezpieczenie przed zalogowaniem się na serwer, który mógł zostać podstawiony/spreparowany na atak (i kradzież naszych danych). Pytanie to będzie miało również miejsce w chwili gdy postawimy nowy serwer/

zainstalujemy na nowo system operacyjny (zmieni się jego odcisk!).

Następnie podajemy nasz login oraz hasło:

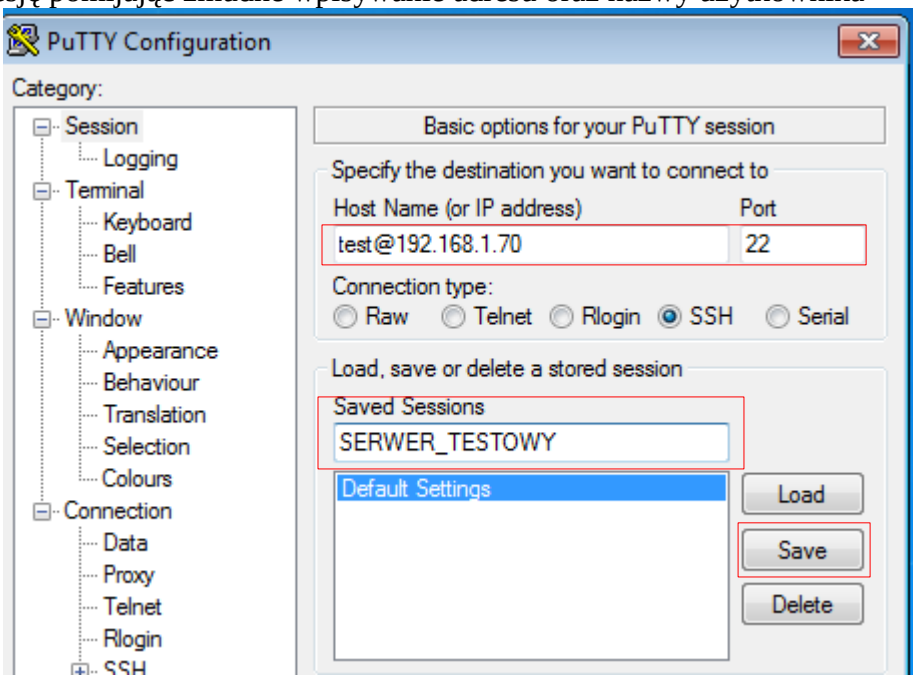


Od tego momentu możemy pracować na naszym systemie tak jak byśmy pracowali na systemie Linux.

INFORMACJA: Proszę pamiętać, że w pole adresu możemy także wpisać nazwę użytkownika (test@192.168.1.70) przez co podczas logowania będziemy musieli podać tylko hasło.

KONFIGURACJA PuTTY

Oczywiście korzystanie w ten sposób z tego małego, lecz rozbudowanego narzędzia przez dłuższy czas byłoby strasznie niewygodne. Dlatego jeżeli wiemy, że będziemy dłużej korzystać z danego serwera to możemy zapisać sobie jego sesję do pamięci systemowej – następnym razem wystarczy wybrać taką sesję pomijając żmudne wpisywanie adresu oraz nazwy użytkownika



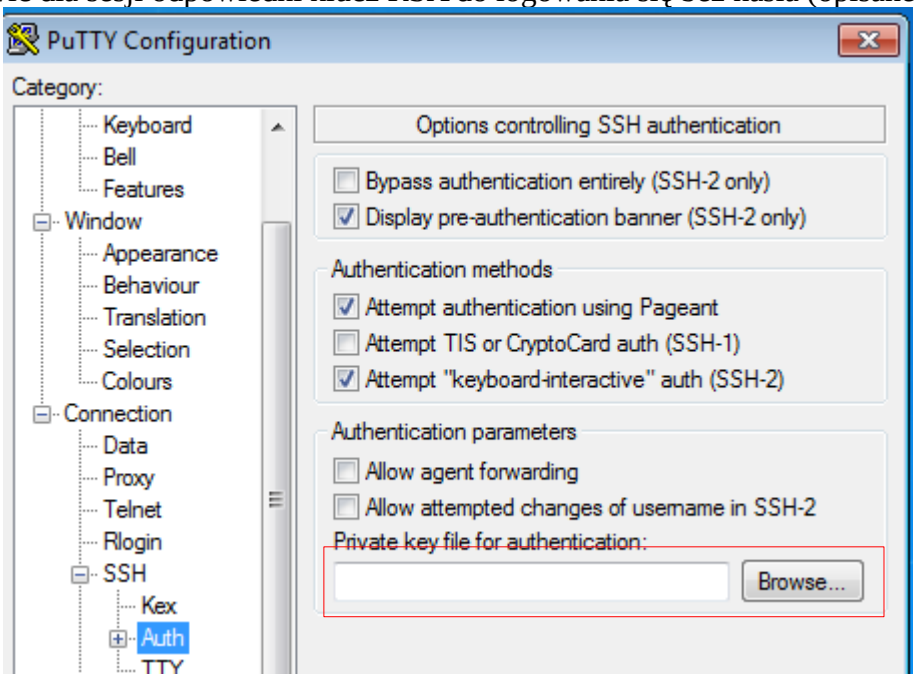
Najpierw wpisujemy parametry połączenia, później nazwę, pod jaką chcemy dane połączenie zachować, a na koniec klikamy przycisk Save. Od teraz połączenie będzie widniało na liście zapisanych sesji i będziemy mogli je bardzo szybko wybrać.

Proszę zauważyć, że lista „Category” zawiera sporą ilość opcji do wyboru. Klikając np. Logging możemy parametry rejestrowania przebiegu naszej sesji. PuTTY pozwala na rejestrowanie

wszystkich danych, tylko wyświetlanych bądź w stanie surowym (takim, jak widzi to SSH). Możemy wybrać także plik, do którego będziemy owe dane zbierać.

Pod opcjami Terminal oraz Windows możemy personalizować zachowania się programu (np. zmienić ilość linii wyświetlanych w oknie, zachowania dźwięków systemowych itp.)

Nas jednak najbardziej będzie interesować jeszcze jedna Connection->SSH->Auth. To tutaj możemy ustawić dla sesji odpowiedni klucz RSA do logowania się bez hasła (opisane później).



Czerwony prostokąt wskazuje miejsce, gdzie należy podać ścieżkę do takowego klucza.

KONFIGURACJA OpenSSH

Serwer SSH NIE POWINIEN DZIAŁAĆ NA USTAWIENIACH DOMYŚNYCH! Co prawda został stworzony tak by jak najdłużej opierać się przed atakami jednak pozostawienie go z domyślnymi parametrami może doprowadzić do przełamania usługi i wdarcie się intruza.

Konfiguracja usługi znajduje się w pliku
/etc/ssh/sshd_config

```
  GNU nano 2.2.6          Plik: /etc/ssh/sshd_config

# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768


^G Pomoc      ^O Zapisz     ^R Wczyt.plik^Y Poprz.str.^K Wytnij     ^C Bież.poz.
^X Wyjdź      ^J Wyjustuj   ^W Wyszukaj  ^V Nast.str. ^U Wklej     ^T Pisownia
```

Powyżej widać początek domyślnej konfiguracji usługi. Większość opcji posiada swój komentarz (rozpoczynający się od znaku #). Każdej linii nie będziemy tutaj omawiać, a jedynie te najważniejsze (szczegółowy opis konfiguracji man sshd_config)

a) należy zmienić port nasłuchiwania – port powinien być dla nas znany i łatwy do zapamiętania lecz różny od domyślnego. Możemy wybierać z puli 1 do 65535 jednak trzeba pamiętać, że powinniśmy używać tych z puli 49152 do 65535; są one tzw. portami dynamicznymi/prywatnymi do dowolnego użytku. Można również wykorzystywać porty powyżej 1024 (poniżej są przeważnie zarezerwowane dla domyślnych usług – chociaż to też dobry sposób na zmylenie potencjalnego włamywacza).
b) PermitRootLogin – to ustawienie powinno być ustawione na 'no'; w przeciwnym wypadku włamywacz w pierwszej kolejności będzie próbował przełamać hasło dla konta 'root'
c) PasswordAuthentication – opcja powinna być ustawiona na 'no'. Ponadto trzeba ją włączyć (w pliku występuje jako komentarz).

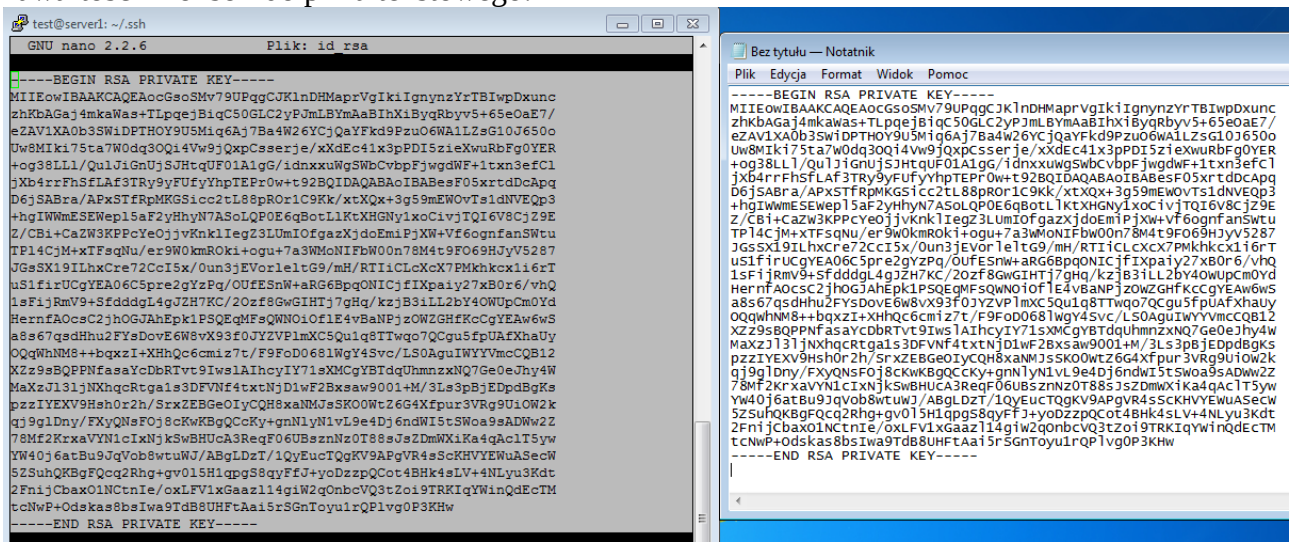
Po powyższych zmianach nie będzie można zalogować się już za pomocą hasła! Musimy w związku z tym wygenerować klucze, dzięki którym będziemy mogli logować się do systemu. Służy do tego narzędzie ssh-keygen. Przebieg generowania klucza przebiega mniej więcej tak:

```
test@server1:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/test/.ssh/id_rsa):
Created directory '/home/test/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/test/.ssh/id_rsa.
Your public key has been saved in /home/test/.ssh/id_rsa.pub.
The key fingerprint is:
8d:2e:73:f4:7f:31:b7:56:4f:39:89:87:96:65:f2:87 test@server1
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|                 |
|         o   . o |
|        S .   O.o|
|       o .   =E*=|
|      o o . . .=*|
|       +   .  .o.|
|             ...  |
+-----------------+
test@server1:~$
```

Najpierw jesteśmy proszeni o podanie lokalizacji wygenerowania klucza prywatnego (możemy kliknąć [ENTER] – zostanie użyta ścieżka z nawiasu; czerwona ramka). Następnie jesteśmy proszeni o podanie hasła autoryzującego nasz klucz RSA. Jeżeli chcemy logować się bez podawania hasła klikamy [ENTER] (niebieska ramka).
To wszystko – w katalogu naszego użytkownika został utworzony katalog .ssh, w którym będą znajdywać się dwa klucze – publiczny (id_rsa.pub) oraz prywatny (id_rsa). Zawartość pliku id_rsa musimy mieć zawsze przy sobie – tylko w ten sposób będziemy się w stanie łączyć z serwerem! Serwer musi posiadać zawartość pliku publicznego – tylko dzięki temu będzie w stanie uwierzytelnić naszego użytkownika. Jednak domyślna konfiguracja zakłada, że klucz nie znajduje się w pliku id_rsa.pub lecz w pliku authorized_keys. Aby wszystko było jak należy wykonujemy polecenie:

```
cd ~/.ssh/
echo id_rsa.pub > authorized_keys
```

Teraz trzeba zdobyć zawartość klucza prywatnego. Można otworzyć plik do edycji i skopiować jego zawartość z konsoli do pliku tekstowego:

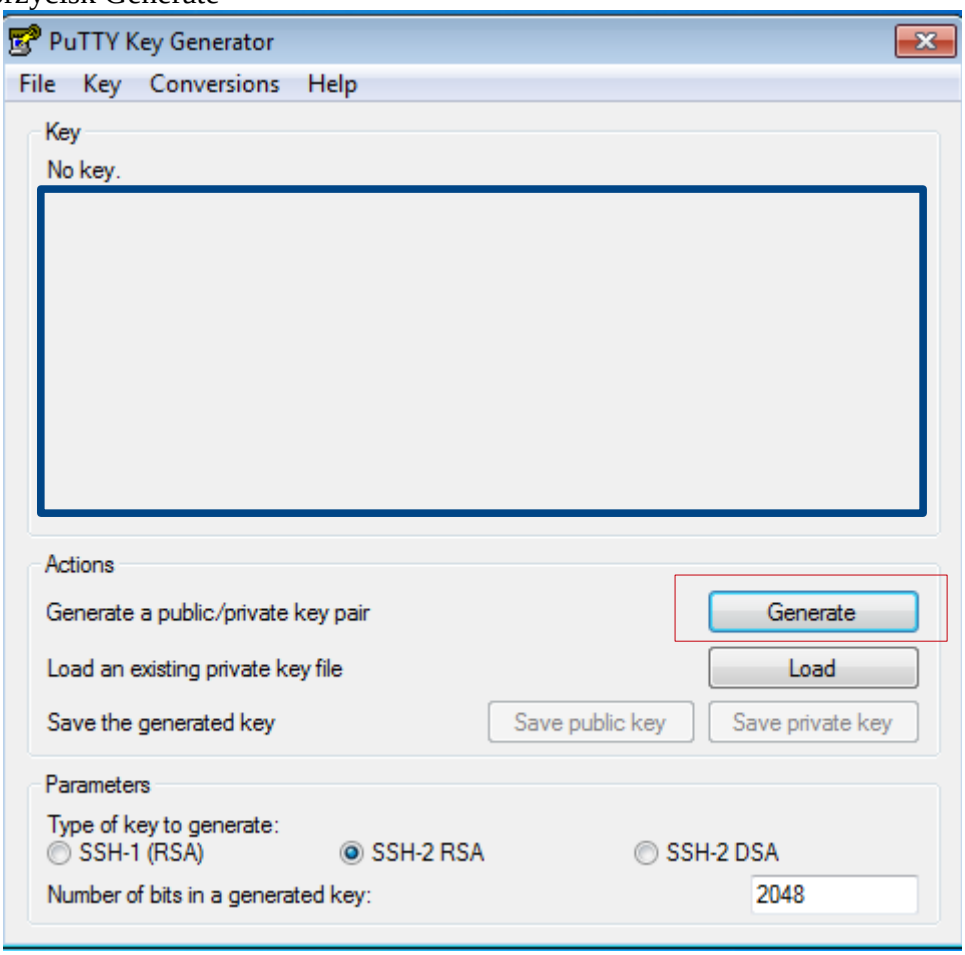


Plik można zapisać pod dowolną nazwą (np. test.ppk) w kodowaniu UTF-8.

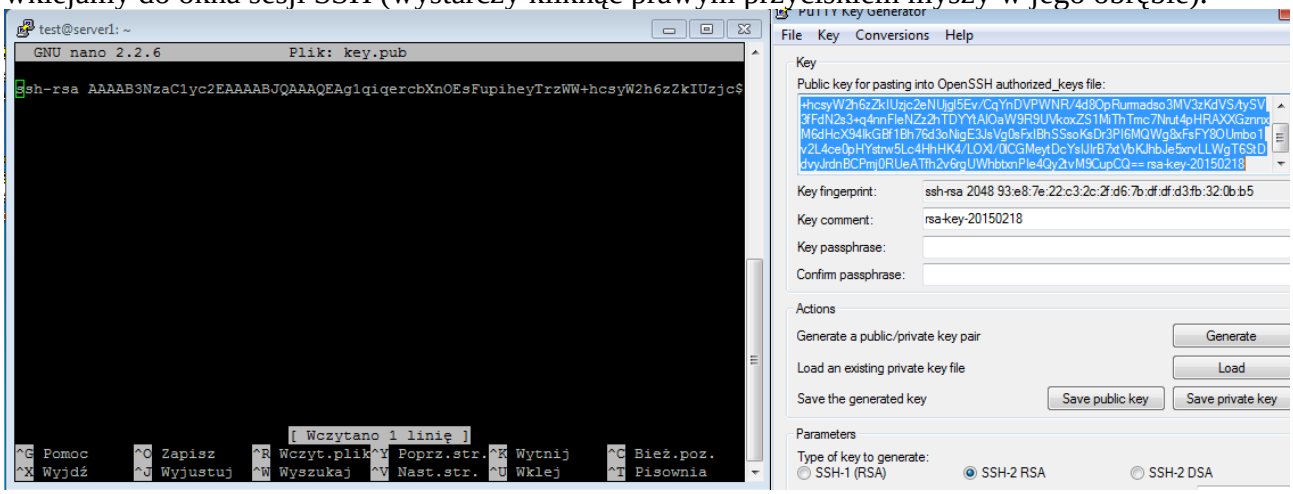
Teraz podając nasz klucz prywatny powinniśmy mieć możliwość logowania się bez podawania hasła.

Lecz... klucz nie działa! Dzieje się tak dlatego, że PuTTY oraz OpenSSH posiadają inne formaty generowania kluczy (oczywiście to PuTTY posiada zły format). Generowanie należy wykonać w inny sposób a mianowicie:

1) ściągnąć ze strony projektu PuTTY program puttygen
2) klikamy przycisk Generate



3) postępujemy zgodnie z instrukcjami na ekranie (poruszamy kursorem w pustym obszarze okna – niebieskie zaznaczenie)
4) zapisujemy klucz prywatny (Save private key) oraz, na wszelki wypadek, klucz publiczny (Save public key)
5) otwieramy połączenie SSH do naszego serwera
6) otwieramy edytor (np. nano)
7) kopiujemy zawartość z pola „Public key for pasting into OpenSSH authorized_keys file" i wklejamy do okna sesji SSH (wystarczy kliknąć prawym przyciskiem myszy w jego obrębie).



8) zapisujemy plik ([CTRL]+[X], [T], podajemy nazwę)
9) wykonujemy polecenie

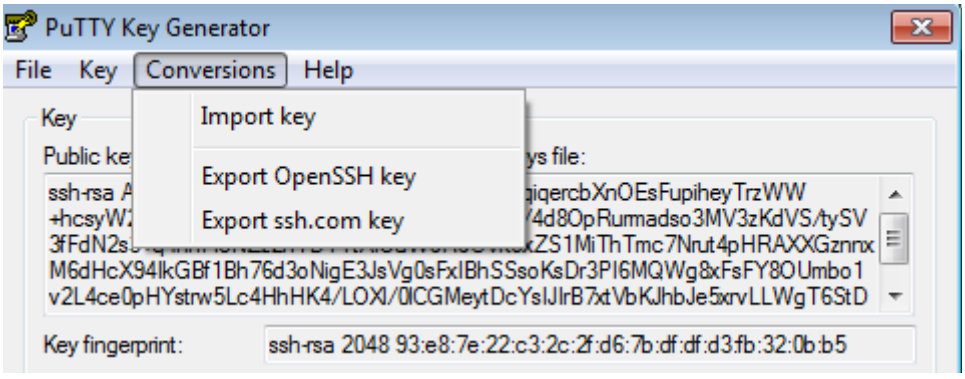
cat key.pub > ~/.ssh/authorized_keys

o ile nasz klucz zapisaliśmy w pliku key.pub (jak nie ZMIENIAMY NAZWĘ)
10) podmieniamy klucz w połączeniu PuTTY na ten wygenerowany przez Key Generator
11) logujemy się na serwer bez hasła! Trzeba tylko pilnować pliku z kluczem prywatnym (dodane rozszerzenie ppk) – kto go posiada jest w stanie łączyć się z naszym serwerem!

ŁĄCZENIE Z SERWERÓW UNIX/LINUX KLUCZEM PRYWATNYM Z PUTTY KEY GENERATOR

Może się zdarzyć, iż pracujemy naprzemiennie w różnych systemach operacyjnych. Klucz prywatny działający z PuTTY nie będzie działał z OpenSSH dostępnym na znakomitej większości systemu Unix/Linux. Aby temu zaradzić:

1) otwieramy program puttygen i klikamy przycisk Load
2) wskazujemy wcześniej wygenerowany klucz prywatny
3) kliamy Conversions->Export OpenSSh key i zapisujemy nasz klucz w formacie zrozumiałym przez OpenSSH

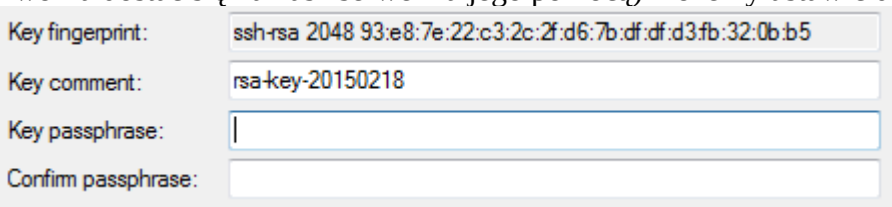


4) klucz ten, po zapisaniu, kopiujemy np. na USB lub na nasz zaufany serwer. Jeżeli będziemy z niego chcieli korzystać w przyszłości wystarczy kopiować go/podawać do niego dostęp przy łączeniu się do naszego serwera poprzez ssh na systemach Linux

WAŻNE!! Proszę pamiętać, by plik **authorized_keys** (na serwerze) posiadał uprawnienia 600. W tym celu, będąc zalogowani jako odpowiedni użytkownik (w tym wypadku użytkownik test) wydajemy odpowiednie polecenie:

chmod 600 ~/.ssh/authorized_keys

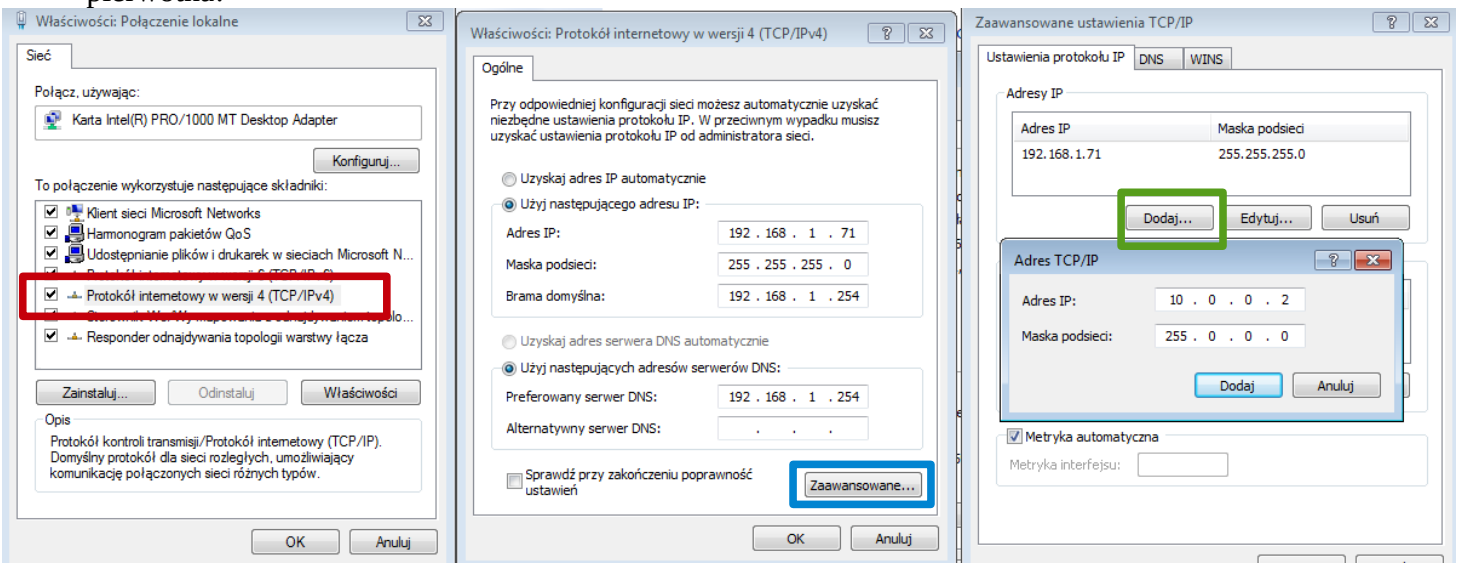
INFORMACJA: By plik z kluczem prywatnym był bardziej zabezpieczony (by nawet jego kradzież nie pozwoliła dostać się na nasz serwer za jego pomocą) możemy ustawić dla niego hasło.



Wpisujemy je w pole **Key passphrase** oraz **Confirm passphrase**. Oczywiście musimy na nowo zapisać klucz prywatny/publiczny. Dodatkowo przy każdy logowaniu będziemy zmuszeni podawać to hasło przy każdy logowaniu jednak jego bezpieczeństwo wzrośnie.

ZADANIA:
1) Skonfigurować połączenie SSH dla swojego serwera
2) Ustalić komunikację tylko za pomocą kluczy RSA
3) Sprawdzić w jaki jeszcze sposób można generować klucze na serwerze tak by działały one również na Windows z PuTTY (może jakiś jego odpowiednik?)
4) Czy da się przemienić klucz z systemu Linux na klucz PuTTY?
5) Proszę klucz z PuTTY wykorzystać do zalogowania się poprzez SSH na nasz serwer przez inny system Linux. W tym celu proszę sklonować maszynę naszego serwera oraz pozmieniać odpowiednio ustawienia sieciowe.

USTAWIENIA SIECIOWE:
Nadal pozostajemy przy swojej klasie 10.0.0.0/8. Na czas ćwiczeń dodajemy drugi adres w ustawieniach naszego głównego systemu operacyjnego ze wskazanej puli. Ponadto zmieniamy ustawienia sklonowanej maszyny tak by również posiadał adres z tej puli jednak INNY niż maszyna pierwotna!



ŹRÓDŁA:
http://leniwy.eu/news,8,Konfiguracja-powloki-shella-w-Linuksie-i-podstawowe-komendy.html