

## Лабораторна робота № 7

### Тема: Віддалений доступ в Linux

**Мета роботи:** Ознайомитися на практиці із засобами віддаленого управління в операційній системі Linux Ubuntu. Набути досвіду і навички управління віддаленим доступом.

#### Завдання.

1. Використовуючи теоретичні відомості вивчити призначення і правила роботи з сервісом ssh.
2. Довстановити при необхідності необхідну програмне забезпечення (Ssh, sshd, putty).
3. Забезпечити доступ по протоколу ssh до свого комп'ютера. Надати права і пароль для віддаленого управління вашим комп'ютером сусіднього - проти годинникової стрілки - комп'ютера.
4. Встановити віддалене підключення до віддаленого (remote) комп'ютера. Як Remote виступає сусідній за годинниковою стрілкою комп'ютер. Тобто ви повинні керувати віддалено правим комп'ютером, і забезпечити можливість віддаленого управління вашим комп'ютером лівому від вас комп'ютера.

#### Короткі теоретичні відомості та порядок виконання завдань

##### 1. Протоколи віддаленого доступу: telnet і ssh

Операційна система UNIX спочатку розроблялася як Інтернет-сервер. Засоби для роботи з Мережею вбудовані безпосередньо в ядро цієї операційної системи, а все необхідне програмне забезпечення для організації сервера входить до складу дистрибутива. UNIX-система працює з усіма мережевими протоколами (особливо з TCP/IP) краще, ніж будь-яка інша операційна система для платформи Intel. Недарма кажуть, що UNIX створений для мережі, як птах для польоту. Всі перераховані вище якості стосуються також і ОС Linux. Існує безліч напрямків, де використовуються Linux-сервери: WWW-сервери, FTP-сервери, поштовики, шлюзи. Тому віддалене управління Linux-сервером має велике значення.

Для віддаленого доступу до Linux використовуються два протоколи telnet і SSH. Telnet - протокол лінії передачі даних Інтернет, який дає можливість комп'ютера функціонувати як термінал, що працює під управлінням віддаленого комп'ютера. Протокол telnet був спочатку розроблений для ARPAnet і є важливою частиною протоколу передачі даних TCP / IP.

Є три головних проблеми пов'язані з використанням telnet, роблячи його поганим вибором для сучасних систем з точки зору безпеки:

Використовувані за умовчанням демони telnet мають декілька вразливостей, виявлених за ці роки, і ймовірно ще кілька досі існують.

Telnet не шифрується ніякі дані, які надсилаються через встановлену зв'язок (Включаючи паролі), і таким чином стає можливим прослуховування зв'язку і використання пароля пізніше для зловмисних цілей.

Відсутність системи аутентифікації в telnet не дає ніякої гарантії, що зв'язок, встановлена між двома віддаленими хостами не буде перервана в середині.

Небажано використання протоколу telnet в системах, для яких важлива безпека, таких як громадський Інтернет. Сеанси telnet не підтримують шифрування даних. Це означає це будь-який, хто має доступ до будь-якого маршрутизатора, комутатора або шлюзу в мережі між двома віддаленими комп'ютерами, з'єднаними сеансом зв'язку по протоколу telnet, може перехопити проходять пакети і легко отримати логін і пароль для доступу в систему (або заволодіти будь-якою іншою інформацією, якою обмінюються ці комп'ютери) за допомогою будь-якої загальнодоступної утиліти подібно tcpdump і Ethereal.

SSH - (Secure Shell) - мережевий протокол, що дозволяє виробляти віддалене управління комп'ютером і передачу файлів. Схожий за функціональністю з протоколом telnet, проте використовує алгоритми шифрування інформації, що передається.

Недоліки telnet привели до дуже швидкого відмови від використання цього протоколу на користь більш безпечного і функціонального протоколу SSH. SSH надає всі ті функціональні можливості, які представлялися в telnet, з додаванням ефектного кодування з метою запобігання перехоплення таких даних, як логіни і паролі. Введена в протоколі SSH система аутентифікації з використанням публічного ключа гарантує, що віддалений комп'ютер дійсно є тим, за кого себе видає.

Оскільки використання для віддаленого управління протоколу telnet неправильно з точки зору безпеки, в лабораторній роботі розглянемо тільки віддалене управління по SSH.

Існує дві версії протоколу SSH: Опис технології протоколу SSH-1:

Спочатку клієнт посилає серверу запит на встановлення SSH з'єднання і створення нового сеансу. З'єднання буде прийнято сервером, якщо він приймає повідомлення подібного роду і готовий до відкриття нового сеансу зв'язку. Після цього клієнт і сервер обмінюються інформацією, які версії протоколів вони підтримують.

З'єднання буде продовжено, якщо буде знайдено відповідність між протоколами і отримано підтвердження про готовність обох сторін продовжити з'єднання по даному протоколу. Відразу після цього сервер посилає клієнтові постійний публічний і тимчасовий серверний ключі. Клієнт використовує ці ключі для шифрування сесійного ключа. Незважаючи на те, що тимчасовий ключ посилається прямим текстом, сесійний ключ і раніше безпечний. Після цього сесійний ключ шифрується тимчасовим ключем і публічним ключем сервера і, таким чином, тільки сервер може його розшифрувати. На цьому етапі і клієнт і сервер мають сесійним ключем і, отже, готові до безпечного сеансу передачі зашифрованих пакетів.

Аутентифікація сервера відбувається виходячи з його можливості розшифровки сесійного ключа, який зашифрований публічним ключем сервера. Аутентифікація клієнта може відбуватися різними способами, в тому числі DSA, RSA, OpenPGP

або по пароллю. Сесія триває до тих пір, поки і клієнт і сервер здатні аутентифікувати один одного. Встановлене з'єднання по протоколу SSH-1 дозволяє захистити дані, що передаються стійким алгоритмом шифрування, перевіркою цілісності даних і стисненням.

Опис технології протоколу SSH-2:

Обидва протоколу, по суті, виконують одні і ті ж функції, але протокол SSH-2 робить це більш елегантно, більш безпечно і більш гнучко. Основна відмінність між протоколами полягає в тому, що протокол SSH-2 розділяє всі функції протоколу SSH між трьома протоколами, в той час як протокол SSH-1 являє собою один єдиний і неподільний протокол. Модуляцією функцій протоколу SSH в трьох протоколах - протоколі транспортного рівня, протоколі аутентифікації і протоколі з'єднання, робить протокол SSH-2 найбільш гнучким і потужним механізмом створення безпечних тунелів. нижче дано короткий опис і призначення кожного з трьох протоколів, що складають протокол SSH-2:

Протокол транспортного рівня - надає можливість шифрування і стиснення даних, що передаються, а також реалізує систему контролю цілісності даних.

Протокол з'єднання - дозволяє клієнтам встановлювати багатопоточний з'єднання через оригінальний SSH тунель, таким чином, знижуючи навантаження, яку створюють клієнтські процеси.

Протокол аутентифікації - протокол аутентифікації відділений від протоколу транспортного рівня, тому що не завжди буває необхідним використання системи аутентифікації. У разі якщо потрібна аутентифікація, процес захищається оригінальним безпечним каналом, встановленим через протокол транспортного рівня.

Слід зазначити, що протокол SSH не вирішує всіх проблем мережевої безпеки. Він лише фокусує свою увагу на забезпеченні безпечної роботи таких додатків, як емулятори терміналу. Використання реалізацій протоколу SSH на серверах і в клієнтських додатках допомагає захистити дані лише в процесі передачі. Протокол SSH жодним чином не є заміною брандмауерів, систем виявлення вторгнень, мережевих сканерів, систем аутентифікації та інших інструментів, що дозволяють захистити інформаційні системи і мережі від атак.

Сервером SSH служить демон sshd, який запускається на UNIX-машині.

Як SSH-клієнта в даний час використовують OpenSSH, PuTTY, SecureCRT, SFTPPlus, TeraTerm і ін. В лабораторному практикумі будуть використані найбільш поширені OpenSSH - для підключення Linux-клієнта і PuTTY - для підключення Windows-клієнта.

OpenSSH (Open Secure Shell - відкритий безпечний shell) - набір програм, надають шифрування сеансів зв'язку в комп'ютерних мережах з використанням протоколу SSH. Він був створений під керівництвом Тео де Раадта як відкрита альтернатива комерційного ПЗ від SSH Communications Security.

PuTTY (від TTY - телетайп, англ. Putty - замазка) - вільно розповсюджуваний клієнт для протоколів SSH. Спочатку розроблявся для Windows, проте пізніше портований на Linux.

## 2. Налаштування сервера ssh на сервері

Щоб служба SSH на сервері почала працювати, необхідно запустити на ньому демон sshd. Бажано додати команду запуску в сценарій завантаження системи. Демон sshd працює по 22 порту. Можна запускати його з-під супердемона xinetd / inetd, але зазвичай sshd запускається самостійно - в режимі standalone.\_

Конфігураційний файл сервера sshd називається / etc / ssh / sshd\_config. Довідку по його синтаксису можна отримати по команді man 5 sshd\_config. У пакеті openssh-server знаходиться конфігураційний файл з типовими настройками.

Щоб захистити ваш комп'ютер від небажаних вторгнень ззовні, рекомендується вписати в конфігураційний файл директиву allowedaddress і перерахувати через пробіл IP-адреси тих машин, з яких дозволено вхід клієнтів. Наприклад, для робочої станції ws2 в лабораторії можна дозволити віддалене підключення тільки з комп'ютера викладача і найближчого комп'ютера зліва:

```
allowedaddress 192.168.1.100 192.168.1.101
```

Приклад файлу конфігурації / etc / ssh / sshd\_config:

```
# Спочатку намагаємося працювати по протоколу SSH 2, а потім,  
# якщо та сторона не підтримує другу версію, - по SSH 1 Protocol 2,1  
# Ключ для протоколу SSH версії 1 HostKey / etc / openssh / ssh_host_key  
# Ключі для протоколу SSH2 - RSA і DSA HostKey /etc/openssh/sshjhost_.rsajtey  
HostKey / etc / openssh / ssh_host_dsa_key  
# Час життя і розмір ключа ssh версії 1 KeyRegenerationInterval 3600  
# За замовчуванням використовується розмір 768 біт, встановимо 1024  
ServerKeyBits 1024  
# Час, через яке ключі сервера будуть створені заново.  
# Періодична зміна ключів підвищує безпеку системи.  
KeyRegenerationInterval 1h  
# Забороняємо реєстрацію користувача root по ssh.  
# Це не виключає можливості віддаленого  
# адміністрування: просто root доведеться зайти під  
# звичайним користувачем, а потім виконати команду su.  
# Зате зломисникові знадобиться вкрати  
# не один, а два паролі: і root, і звичайного користувача.  
PermitRootLogin no  
# Протоколювання (розкоментуйте, якщо потрібно  
# вести журнал за допомогою системи syslog) #SyslogFacility AUTH  
# Аутентифікація  
# Включає парольний аутентифікацію  
# і забороняє порожні паролі  
PasswordAuthentication yes PermitEmptyPasswords no  
#StrictModes yes  
# Використовуємо RSA-аутентифікацію  
RSAAuthentication yes PubkeyAuthentication yes
```

```
# Аутентифікація rhosts - зазвичай не використовується,
# тому забороняємо її:
# призначені для користувача файли- /. Rhosts і - /. Shosts не
використовуватимуться
RhostsAuthentication no IgnoreRhosts yes
# НЕ використовувати PAM аутентифікацію
PAMAuthenticationViaKbdInt no
# Додатковий час клієнту на те, щоб аутентифікуватися.
# Якщо за цей час клієнт не зміг ввести пароль,
# з'єднання буде припинено
LoginGraceTime 2m
# Шлях до банера
Banner / some / path
# Підсистема sftp-сервера
Subsystem sftp / usr / libexec / openssh / sftp-server
```

Запустимо сервер з CentOS Linux. Командою ps (process status) з ключами C (by command name) і l (long format) перевіримо, чи запущений демон sshd, а командою ifconfig перевіримо адресу сервера (рис. 1).

```
root@CentOS-7 ~# ps -C sshd -l
  PID TID          C PRI NI ADDR SZ WCHAN TTY          TIME CMD
  1234 1 0 2205    1 0  04      0 - 1234 -   ?        00:00:00 sshd
root@CentOS-7 ~# ifconfig
eth0:
  Link encap:Ethernet  HWaddr 88:0C:29:89:76:CC
  inet addr:192.168.100.3  Bcast:192.168.100.255  Mask:255.255.255.0
  inet6 addr: fe80::28c:29ff:fe89:76cc Scope:Link
  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
  RX packets:0 errors:0 dropped:0 overruns:0 frame:0
  TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1000
  RX bytes:0 (0.0 KiB)  TX bytes:1776 (1.5 KiB)
  Interrupt:167 Base address:0x2000

lo:
  Link encap:Local Loopback
  inet addr:127.0.0.1  Mask:255.0.0.0
  inet6 addr: ::1:1:1:1 Scope:Host
  UP LOOPBACK RUNNING  MTU:65536  Metric:1
  RX packets:0 errors:0 dropped:0 overruns:0 frame:0
  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:0
  RX bytes:568 (5.6 KiB)  TX bytes:568 (5.6 KiB)

root@CentOS-7 ~#
```

Рис. 1. Перевірка завантаження служби sshd і IP-адреси сервера. Бачимо, що процес sshd запущений. Батьківським для нього є процес ініціалізації з ідентифікатором 1. IP-адреса сервера дорівнює, як було задано при установці, 192.168.100.3.

Завантажуємо клієнтську машину з Alt Linux Lite. Запускаємо на ній термінал і перевіряємо зв'язок з сервером - набираємо команду ping 192.168.100.3. Як видно з рис. 2 встановити з'єднання з сервером відразу після завантаження не вдається.



Рис. 2. Перевірка зв'язку з Linux-сервером

Потрібно налаштування IP-протоколу. Вибираємо (див. Рис. 3) Налаштування - Центр управління системою - Мережа, вводимо IP-адреса 192.168.100.4 і натискаємо кнопку «Застосувати».

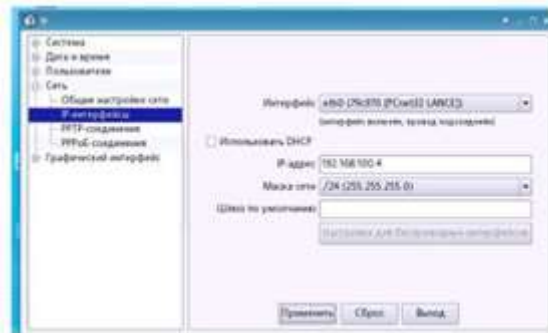


Рис. 3. Налаштування IP-інтерфейсу на Linux-клієнті

Знову перевіряємо можливість встановлення зв'язку з сервером. Як видно з рис. 4 тепер клієнт «бачить» сервер.

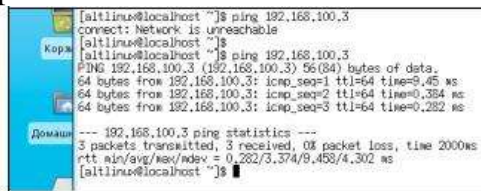


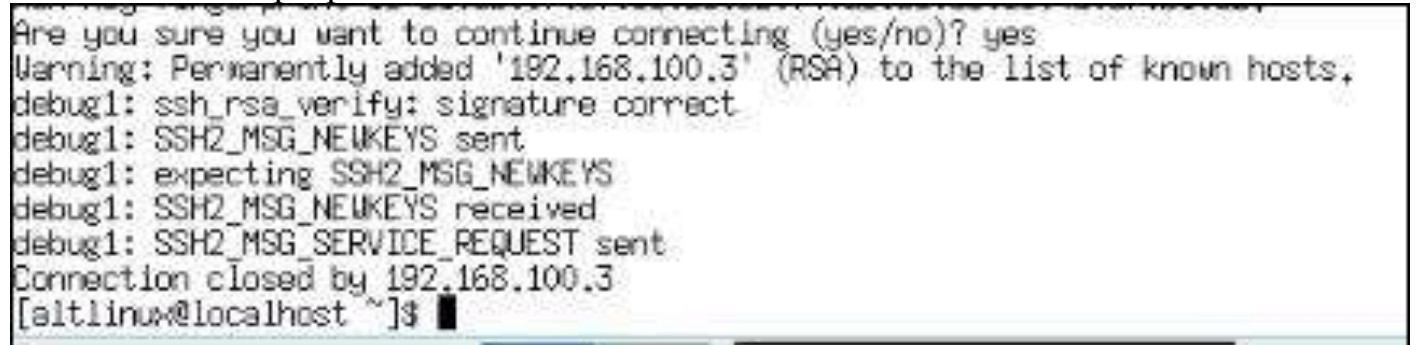
Рис. 4. Після включення клієнта в мережу 192.168.

100 з'єднання встановлено Оскільки настройка при завантаженні з Live-CD не зберігається, задавати IP-адреса клієнту треба при кожному завантаженні Alt Linux с Live-CD. Тепер пробуємо підключитися віддалено до сервера.

Клієнтська програма ssh може запускатися з численними ключами, формат її запуску в загальному випадку наступний .:

ssh [ключі] [ключі\_с\_аргументами] [логін\_імя @] хост.домен [команда]

Ми будемо використовувати ключ l, за допомогою якого можна вказати, від імені якого користувача буде здійснюватися реєстрації на віддаленій машині, і ключ v, який включає відображення всієї налагоджувальної інформації. Набираємо команду ssh з адресою сервера і цими ключами. Як видно з цього малюнка, клієнт і сервер обмінялися інформацією про те, які версії протоколів вони підтримують (OpenSSH\_4.7p1 на клієнтській стороні і OpenSSH\_4.3 на сервері), сервер надіслав відкритий RSA ключ і програма запитує від користувача підтвердження на включення сервера в список відомих хостів.



```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.100.3' (RSA) to the list of known hosts.
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: SSH2_MSG_SERVICE_REQUEST sent
Connection closed by 192.168.100.3
[altlinux@localhost ~]$
```

Рис. 5. Отримання відкритого ключа для шифрування даних від сервера.

Це важлива властивість протоколу SSH. Він розроблений щоб захистити користувача проти атак відомих як спуфінг або «зловмисник посередині». Одна з проблем, пов'язана зі старими протоколами типу Telnet, FTP, RSH і ін. Крім того, що вони передають всю інформацію відкритим текстом, полягає в тому, що дані протоколи уразливі до атаки подібного роду. Атакуючий, який має доступ до проміжної мережі, може перехопити ваші пакети, зберегти їх, а потім відіслати безпосередньому адресатові. Ще гірше, він може перезаписати ваші пакети, наприклад, замінивши ls -la mydir на rm -r mydir (видалення замість перегляду) або відправити вам протроянений файл замість оригіналу через ваш перехоплений FTP сеанс. Атакуючий може, нарешті, просто перенаправити ваше з'єднання до іншого комп'ютера, так що ви пошлете ваш пароль іншій машині.

Використовуючи цей прийом, атакуючий зможе дізнатися пароль, який захищає вашу обліковий запис, і зможе потім реєструватися під вашим ім'ям для своїх цілей.

SSH надає можливість підтвердження автентичності хоста, з яким ви з'єднуєтеся. Якщо ви правильно верифікувати хост, не існує способів читання або маніпуляції вашими пакетами проміжним пристроєм. Успішна верифікація хоста показує, що з'єднання зашифровано від початку до кінця - ваш SSH клієнт встановив безпечне з'єднання безпосередньо з SSH сервером, і ніякі проміжні машини не мають доступу до цього з'єднання.

Так як ми підключаємося до цієї машини вперше, і SSH не працює за принципом третього довіреної особи, такого як Certificate Authorities, вся робота, пов'язана з управлінням ключами, лежить на вас. Ваш клієнт показує відбиток відкритого



ключа (key fingerprint), просту для читання рядок чисел, яку ви можете використовувати для перевірки ключа вручну. Якщо ви відповідаєте "Так, відбиток правильний", ваш SSH клієнт продовжить аутентифікацію, давши вам можливість ввести ваш пароль і приступити до роботи.

Якщо під час з'єднання з сервером атакуючий перехопив ваше SSH з'єднання і пропустив через себе весь трафік, він може підсунути вам свій ключ, замість реального відкритого ключа хоста. Існує кілька способів перевірки автентичності ключа. Наприклад, власник системи може розмістити відбиток ключа на своїй безпечної веб сторінці. Інший варіант - ви можете зателефонувати системному адміністратору хоста і перевірити ключ по телефону (якщо виключена можливість перехоплення зловмисником телефонної розмови).

Підтверджуємо продовження з'єднання, набравши yes, і отримуємо від програми повідомлення про те, що сервер включений в список відомих хостів (рис. 6).

```
connection closed by 192.168.100.3
[altlinux@localhost ~]$ ssh 192.168.100.3 -i root -v
OpenSSH_4.7p1, OpenSSL 0.9.8d 28 Sep 2006
debug1: Reading configuration data /etc/openssh/ssh_config
debug1: Applying options for *
debug1: Connecting to 192.168.100.3 [192.168.100.3] port 22.
debug1: Connection established.
debug1: identity file /home/altlinux/.ssh/id_rsa type -1
debug1: identity file /home/altlinux/.ssh/id_dsa type -1
debug1: Remote protocol version 2.0, remote software version OpenSSH_4.3
debug1: match: OpenSSH_4.3 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_4.7
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client aes256-ctr hmac-md5 none
debug1: kex: client->server aes256-ctr hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<4096<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug1: Host '192.168.100.3' is known and matches the RSA host key.
debug1: Found key in /home/altlinux/.ssh/known_hosts:1
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey,gssapi-with-mic,password
debug1: Next authentication method: publickey
debug1: Trying private key: /home/altlinux/.ssh/id_rsa
debug1: Trying private key: /home/altlinux/.ssh/id_dsa
debug1: Next authentication method: password
root@192.168.100.3's password: █
```

```
debug1: Next authentication method: password
root@192.168.100.3's password:
debug1: Authentication succeeded (password).
debug1: channel 0: new [client-session]
debug1: Entering interactive session.
debug1: Sending environment.
debug1: Sending env LC_MONETARY = ru_RU.UTF-8
debug1: Sending env LC_NUMERIC = ru_RU.UTF-8
debug1: Sending env LC_MESSAGES = ru_RU.UTF-8
debug1: Sending env LC_COLLATE = ru_RU.UTF-8
debug1: Sending env LANG = ru_RU.UTF-8
debug1: Sending env LC_CTYPE = ru_RU.UTF-8
debug1: Sending env LC_TIME = ru_RU.UTF-8
Last login: Wed May  6 16:22:37 2009 from 192.168.100.7
[root@CentOS-2 ~]# █
```

Рис. 6. Перевірка RSA

Коли ви відповіли "так", наш SSH клієнт зберіг ключ сервера у файлі known\_hosts, який фактично є вашим особистим Certificate Authority - списком ключів всіх SSH серверів, з якими ви працюєте.



Тепер можна віддалено підключитися до сервера. Повторюємо команду ssh 192.168.100.3 -l root -v і отримуємо інформацію про встановлення з'єднання (рис. 7), де останньою стадією аутентифікації виступає введення пароля користувача root віддаленого комп'ютера-сервера.

```
[root@CentOS-2 ~]# reboot
Broadcast message from root (pts/0) (Wed May 6 16:09:31 2009):
The system is going down for reboot NOW!
[root@CentOS-2 ~]# Connection to 192.168.100.3 closed by remote host.
Connection to 192.168.100.3 closed.
[altlinux@localhost ~]# ssh 192.168.100.3 -l root
ssh: connect to host 192.168.100.3 port 22: No route to host
[altlinux@localhost ~]# ssh 192.168.100.3 -l root
root@192.168.100.3's password:
Last login: Wed May 6 16:03:24 2009 from 192.168.100.7
[root@CentOS-2 ~]# ps -l
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
4 S 0 2548 2546 0 83 0 - 1122 wait pts/0 00:00:00 bash
4 R 0 2571 2548 0 84 0 - 1043 - pts/0 00:00:00 ps
[root@CentOS-2 ~]#
```

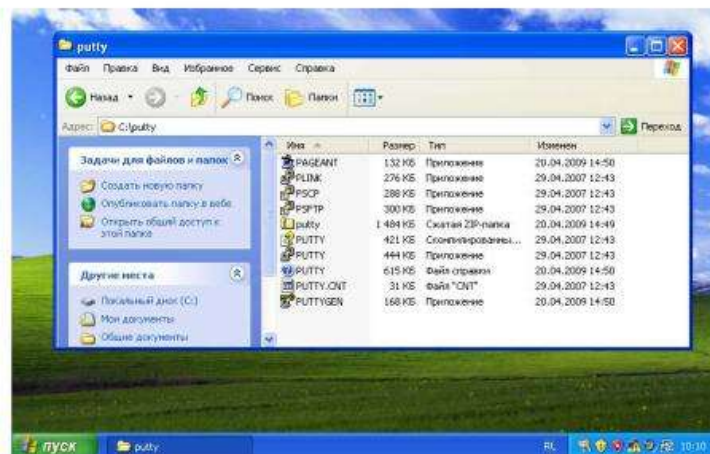


Рис. 7. Інформація про встановлення з'єднання

На цьому малюнку видно, що відразу після команди reboot встановити заново віддалене підключення не вдається - сервер ще не завантажився. Після встановлення з'єднання команда ps показує, що на сервері виконуються два

призначених для користувача процесу - оболонка `bash` і власне команда `ps`. Обидва процеси запущені з віддаленої консолі `pts / 0`.

Завершуємо віддалений сеанс командою `logout`.

Розроблена лабораторна установка дозволяє в деталях вивчати віддалене управління Linux-сервером. Налаштування SSH на сервері може проводитися з використанням конфігураційного файлу `sshd_config`. Довідку по його синтаксису користувач може отримати по команді `man sshd_config`. У пакеті `openssh-server` знаходиться конфігураційний файл з типовими настройками.

#### 4. Віддалений доступ з Windows-клієнта

Вбудовані клієнти SSH є не у всіх операційних систем, у UNIX - є, у Windows - немає. Під Windows можна поставити SSH клієнта - PuTTY, (офіційний сервер <http://www.chiark.greenend.org.uk/~sgtatham/putty/>). Склад використовуваного дистрибутив а

PuTTY показаний на рис. 8.

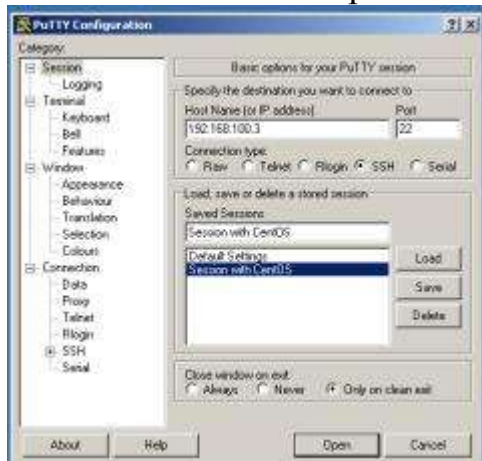


Рис. 8. Склад дистрибутива PuTTY.

Putty це кілька окремих програм, призначених для роботи з unixсервером по протоколам SSH1, SSH2, Telnet, Rlogin, Raw. Програма працює на Windows для Intel x86 і Alpha, а також на UNIX. Показаний на рис. 8 повний комплект програм, під загальною назвою PuTTY, складається з наступних утиліт:

- `putty.hlp` - файл довідки;
- `putty.exe` - клієнт для з'єднання з сервером по протоколах telnet, ssh, raw, rlogin;
- `puttygen.exe` - генератор rsa / dsa ключів;
- `pageant.exe` - агент аутентифікації, зберігає ключі в пам'яті, при його використанні не потрібно руками вводити ключову паролльний фразу;
- `plink.exe` - інтерфейс командного рядка для putty;
- `pscp.exe` - безпечне копіювання файлів;
- `psftp.exe` - безпечний ftp клієнт для копіювання, перегляду, перейменування файлів і т.д.

Встановлювати putty необов'язково, можна просто скопіювати файли в потрібну директорію.

Запускаємо `putty.exe` і створюємо профіль з настройками для нашого віддаленого сервера. У PuTTY можна створювати профілі для різних SSH серверів, так що Вам не доведеться забивати настройки для конкретного сервера, коли Ви захочете до

нього черговий раз під'єднатися. Зараз ми знаходимося в категорії Sessions. Вводимо адресу хоста 192.168.100.3 в рядок Host Name (or IP address), залишаємо пропонувані за умовчанням номер порту (22) і Protocol (SSH). Під написом Saved Sessions (збережені сесії) вводимо ім'я профілю, наприклад, Session with CentOS, яке допоможе згадати, до якого сервера відноситься цей профіль. Потім йдемо в категорію Connection -> Data і вказуємо в Auto-login username ім'я користувача, під яким будемо підключатися до SSH сервера - root. Тепер повертаємося в категорію Sessions, зберігаємо профіль, натиснувши Save. Наступного разу, коли Ви запустите PuTTY, просто виберіть відповідний профіль з Saved Sessions, клікніть Load і Open. Тепер ми можемо приєднатися до нашого SSH сервера, просто клікнувши Open. При першому з'єднанні цього клієнта, з'являється (див. Рис. 9) попередження про загрозу безпеці, аналогічне повідомленням, показаному на рис. 9. Це відбувається тому, що PuTTY ще не відомий відкритий ключ хоста сервера. PuTTY записує ключ хоста для кожного сервера, до якого ви приєднуєтесь, в реєстр Windows. Кожен раз, коли ви приєднаєтесь до сервера, вона перевіряє, що ключ хоста сервера надав є той же самий ключ, що був в останньому з'єднанні. Якщо це не так, ви отримаєте попередження і будете мати можливість обірвати ваше з'єднання, перш ніж ви наберете будь-яку приватну інформацію (таку як пароль). Оскільки ми підключаємося до сервера по протоколу SSH, то до цього з'єднання відноситься все сказане в попередньому вище щодо перевірки відкритого ключа сервера для підключення Linux-клієнта. У нашому випадку сумніву в достовірності сервера відсутні - натискаємо кнопку:

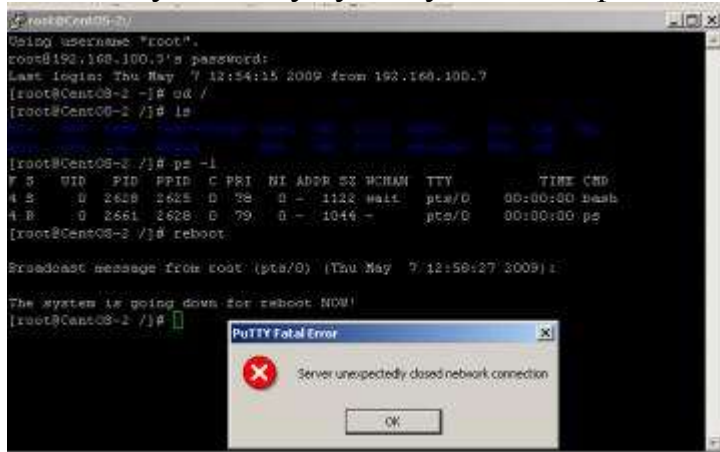


Рис. 9. Попередження PuTTY при першому підключенні до віддаленого хосту. Оскільки ми зберегли ім'я, під яким єднаємося, в налаштуваннях профілю, нам не потрібно забивати його знову, ми тільки вкажемо пароль користувача (рис. 9). Віддалене з'єднання встановлено. На рис. 9 показано виконання кількох команд на віддаленому Linux-сервері і потім - дистанційна перезавантаження сервера. Після перезавантаження сесія стає неактивною. Її слід закрити і знову з'єднатися з сервером, коли він завантажиться. Було розглянуто найбільш поширений спосіб з'єднання - з пральний ідентифікацією. Якщо хто-небудь дізнається ім'я та пароль, він теж зможе з'єднатися. Тому якщо у Вас простий пароль і / або Ви стали жертвою атаки з

прямим перебором, можуть бути проблеми. В якості альтернативи можна використовувати спосіб Аутентифікації самого користувача за використанням шифрування з відкритим ключем. Ви можете використовувати PuTTYgen для виготовлення пари закритого і відкритого ключів. Потім відкритий ключ треба буде передати серверу, а закритий ключ прикріпити до профілю PuTTY. Ці процедури докладно описані в керівництві putty.hlp.

Таким чином, розроблена лабораторна установка дозволяє в деталях вивчати віддалене управління Linux-сервером з Windows-клієнта.

Порядок виконання роботи:

1) Перевірити, чи запущений супер-сервер xinetd. Якщо не запущений - доустановити супер-

сервер з пакета xinetd-2.3.14-10.el5.i386.rpm. (/usr/sbin)

2) Перевірити, використовуючи команду find, наявність і місце розташування ssh-сервера sshd. Якщо sshd не встановлено - доустановити з пакетів openssh-4.3p2-16.el5.i386.rpm, openssh-askpass-4.3p2-16.el5.i386.rpm; openssh-server-4.3p2-16.el5.i386.rpm. (/usr/sbin)

*Linux:*

1. OpenSSH; якщо Ви користуєтеся Linux, наприклад Ubuntu, швидше за все все вже встановлено, до нас, так що можна сміливо писати в терміналі: ssh root@.

2. PuTTY є і для Linux, в тому числі в офіційних репозиторіях Debian і Ubuntu. MAC OS:

1. OpenSSH; це безкоштовно!

Плагіни для браузерів:

Також можна прокинути RDP-шний трафік через SSH-тунель. Для цього потрібно поправити конфігураційний файл xrdp:

```
$ vi /etc/xrdp/xrdp.ini
```

У секцію потрібно додати рядок: address 127.0.0.1

```
$ systemctl restart xrdp
```

Перевірити, що все правильно, можна так:

```
$ Nmap -p 3389 Starting Nmap 6.47 (http://nmap.org) at 2016-10-04 13:07 MSK Nmap scan report for unspecified.mtw.ru () Host is up (0.0087s latency). PORT STATE SERVICE 3389 / tcp closed ms-wbt-server
```

*Запустіть PuTTY.* У деревовидному меню зліва Connection SSH Tunnels. Далі додаємо новий Forwarded Port (Source port: 3389, Destination: localhost: 3389). Натискаємо Add.

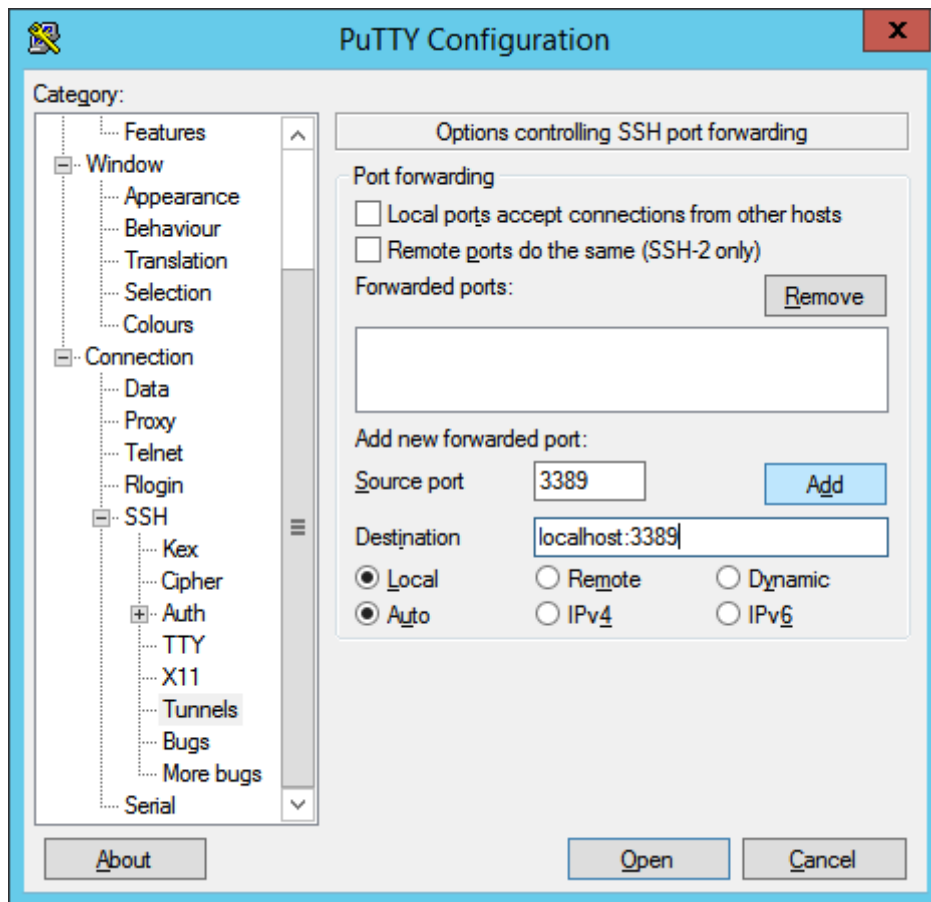


Рис. 10.

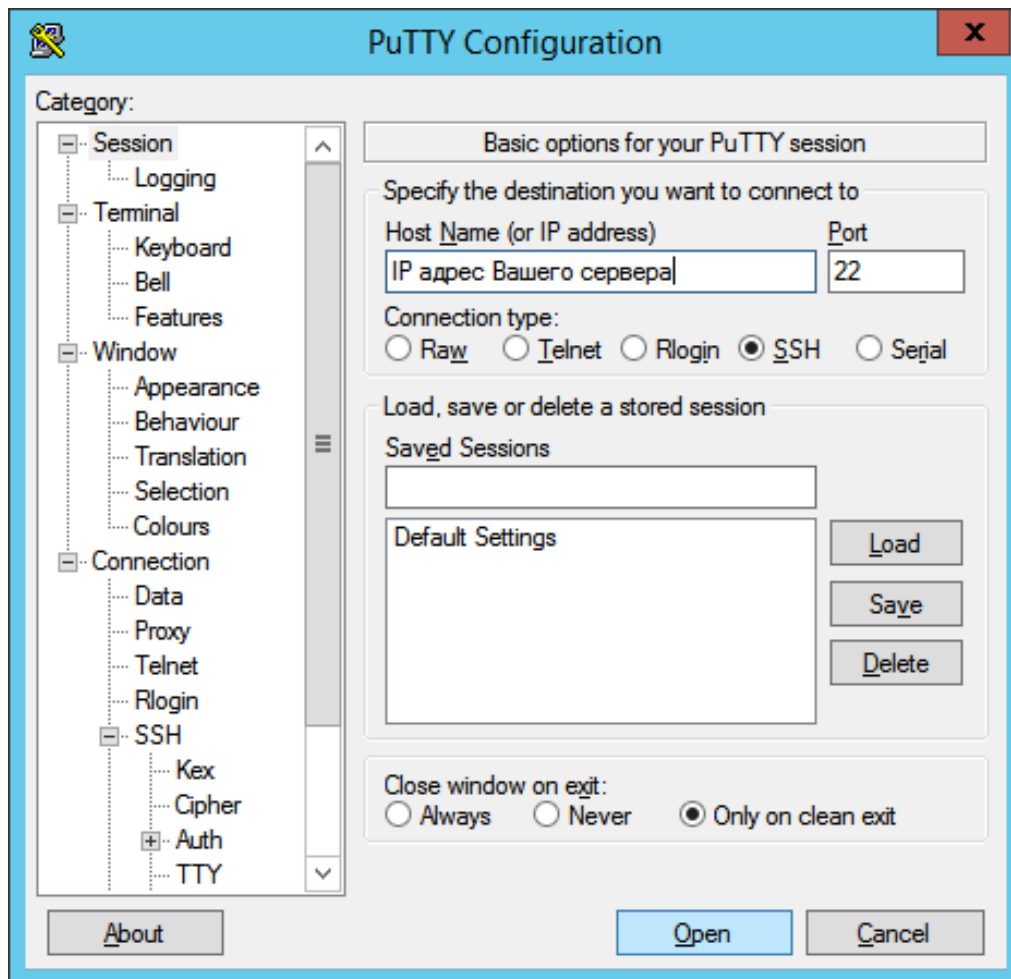


Рис. 11.

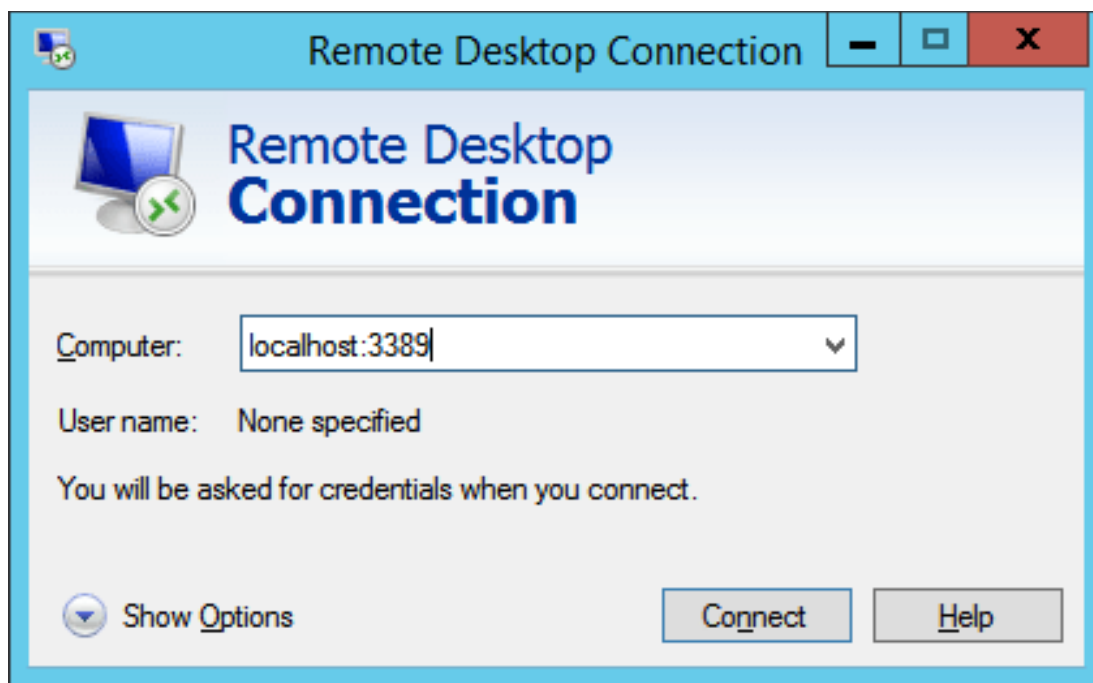


Рис. 12.

VNC клієнт:



Для прикладу поставимо цю DE:

```
$ Apt-key adv --recv-keys --keyserver keys.gnupg.net E1F958385BFE2B6E $ echo  
"deb http://packages.x2go.org/debian jessie main" \u003e /etc/apt/sources.list.d/x2go .list  
$ echo "deb-src http://packages.x2go.org/debian jessie main" \u003e \u003e /etc/apt/sources.list.d/x2go.list $ apt-get update $ apt-get install x2go-keyring && apt-get update $ apt-get install x2goserver x2goserver-xsession
```

Висновок наступної команди повинен показати, що x2go готовий до роботи:

```
$ Systemctl status x2goserver? x2goserver.service - LSB: Start and stop the X2Go daemon  
Loaded: loaded (/etc/init.d/x2goserver) Active: active (running) since Tue 2016-10-11 22:05:51 MSK; 30min ago ...
```

Потрібно знайти в файлі profile рядок «mesg n» і замінити її на «tty -s && mesg n».

```
$ Vi .profile
```

Наступна команда виведе шлях до виконуваного файлу startfluxbox, знадобиться під час налаштування клієнта:

```
$ Whereis startfluxbox
```

Установка сервера на Ubuntu:

```
$ Apt-get install xfce4 xfce4-terminal $ add-apt-repository ppa: x2go / stable $ apt-get update $ apt-get install x2goserver x2goserver-xsession  
$ Vi .profile
```

Для Windows - завантажуюмо, ставимо, запускаємо.

Запускаємо клієнт:

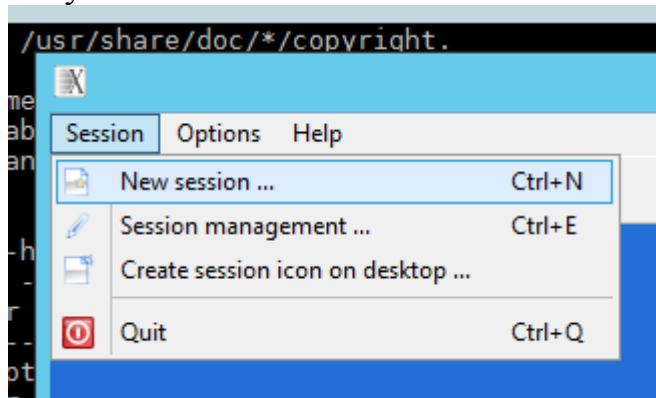


Рис. 13.

В налаштуваннях сесії вказуємо: в поле Host - IP вашого сервера, в поле Login - root, порт залишаємо як є, session type - той GUI який ставили.



Рис. 14.

Як ви можете бачити, є можливість аутентифікації по ключу. Загалом багато всякого. І звук можна через PulseAudio виводити.

Після натискання Ок ви побачите ось такі ось чарівні штучки, на які потрібно натиснути для отримання запиту на введення пароля і підключення до обраної сесії:

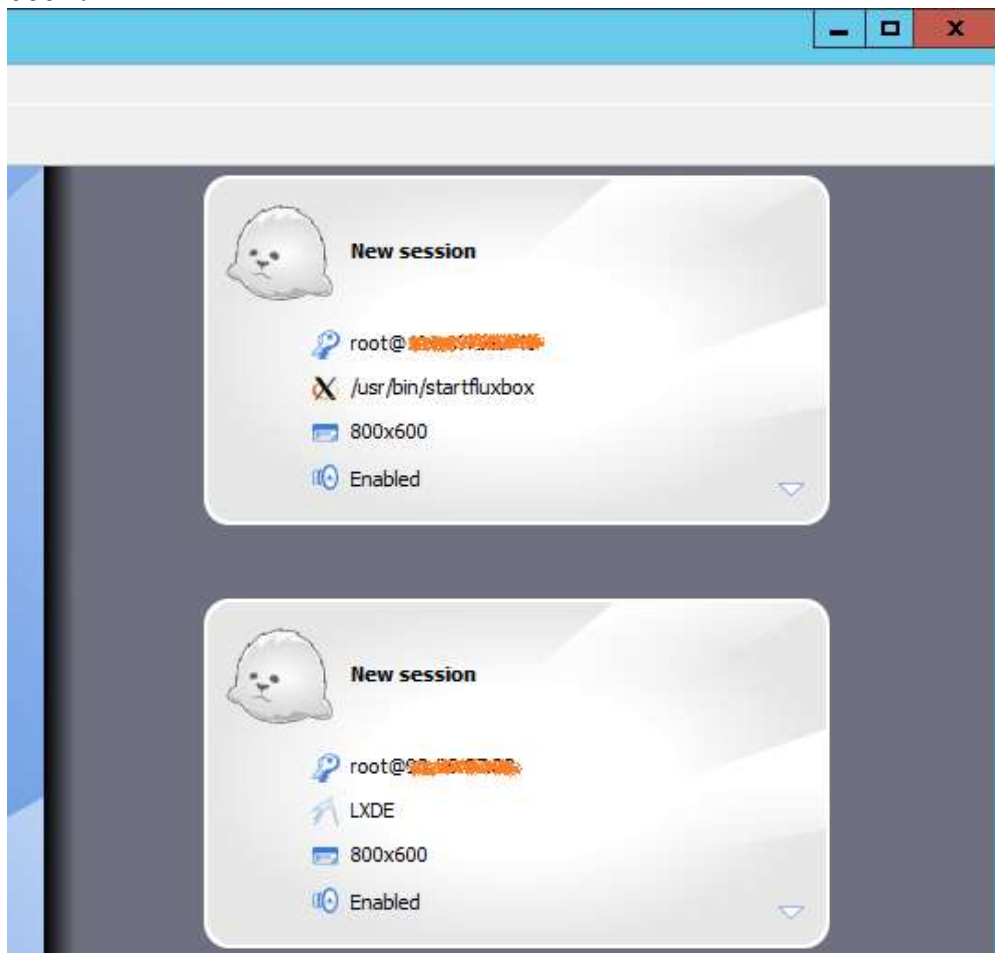


Рис. 15.

Шлях до FluxBox'a доводиться прописувати руками.

Важливою можливістю x2go є можливість запуску будь-якого графічного додатка взагалі без установки DE. Для цього в настройках сесії потрібно в секції session type потрібно вибрати пункт single application і вибрати виконується додаток або ввести шлях до програми яку слід запустити.

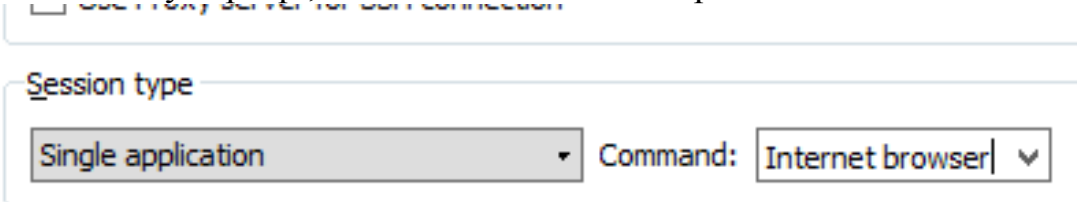
У цьому випадку установка ПО на сервер буде виглядати наступним чином. У випадку з Ubuntu:

```
$ Add-apt-repository ppa: x2go / stable $ apt-get update $ apt-get install x2goserver x2goserver-xsession
```

А тепер важливий момент, підключитися без цього фікса не вийде! Потрібно знайти в файлі .profile рядок «mesg n || true »і замінити її на« tty -s && mesg n ».

```
$ Vi .profile $ apt-get install firefox xterm
```

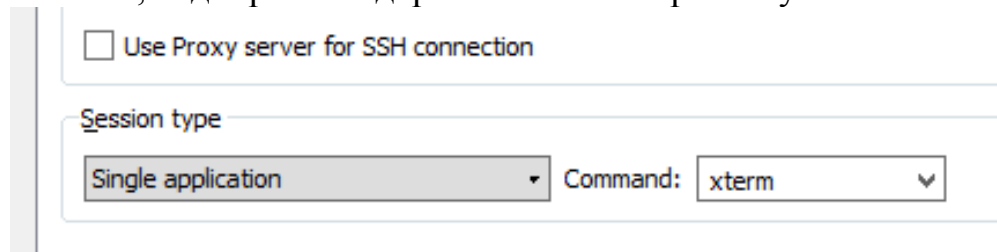
І налаштувавши сесію як показано нижче, можна буде запустити браузер на віддаленому сервері, а на вашій машині відкриється вікно його відображає:



The image shows a configuration window for an x2go session. It has a section titled "Session type" with a dropdown menu currently showing "Single application". To the right of this is a "Command:" label followed by another dropdown menu showing "Internet browser".

Рис. 16.

Або так; тоді просто відкриється вікно терміналу:



The image shows a configuration window for an x2go session. It has a checkbox labeled "Use Proxy server for SSH connection" which is unchecked. Below it is a section titled "Session type" with a dropdown menu currently showing "Single application". To the right of this is a "Command:" label followed by another dropdown menu showing "xterm".

Рис. 17.

Нижче ви можете бачити скріншот вікна статусу поточної сесії. Помаранчевими цифрами відзначені кнопки:

1. «Suspend session» - після натискання на цю кнопку з'єднання буде розірвано, але сесія залишиться і буде очікувати повторного підключення. Всі запущені вами на сервері додатки продовжать свою роботу;
2. «Terminate session» - після натискання підключення до сервера буде розірвано, а запущені вами на сервері програми будуть завершені.

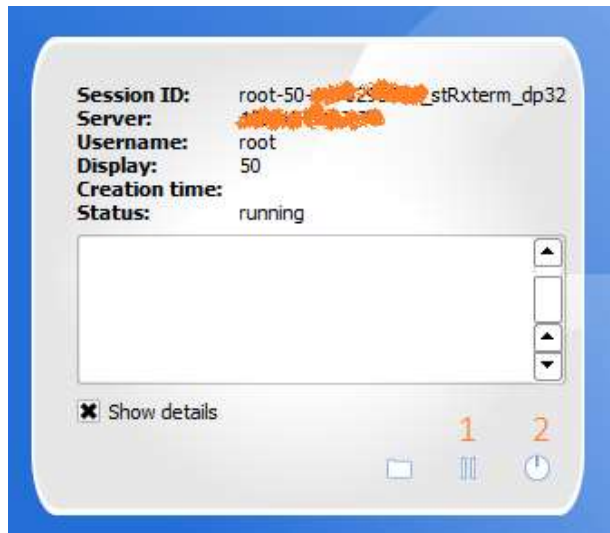


Рис. 18.

### *TeamViewer*

Останній спосіб віддаленого доступу до робочого столу.

Установка на Ubuntu:

```
$ Apt-get update $ apt-get install lubuntu-desktop $ reboot $ dpkg --add-architecture  
i386 $ apt-get update $ wget  
http://download.teamviewer.com/download/teamviewer_i386.deb $ dpkg -i  
teamviewer_i386 .deb $ apt-get -f install $ teamviewer --passwd
```

Також необхідно прийняти ліцензійну угоду TeamViewer'a, це можна зробити за допомогою «Аварійного режиму», або додати наступні рядки в кінець файлу /opt/teamviewer/config/global.conf:

```
$ Echo "EulaAccepted \u003d 1" \u003e /opt/teamviewer/config/global.conf $  
echo "EulaAcceptedRevision \u003d 6" \u003e  
/opt/teamviewer/config/global.conf $ teamviewer --daemon restart
```

Наступна команда покаже стан демона TeamViewer'a і необхідний для підключення дев'ятизначний TeamViewer ID:

```
$ Teamviewer --info
```

Після запуску клієнта завантаженого тут, потрібно ввести TeamViewer ID в поле Partner UD і натиснути на кнопку «Connect to partner». Далі TeamViewer запросить пароль.

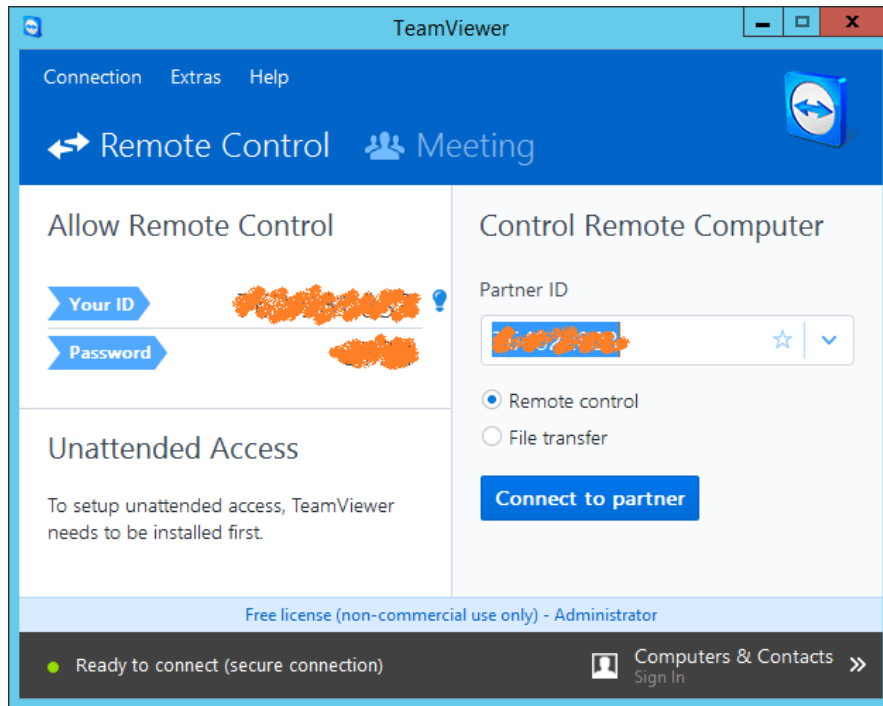


Рис. 19.

### **Потрібно:**

- 1) Виконати поставлене завдання в ОС Windows Server 8.0 на віртуальній машині або сервері;
- 2) Оформити виконане завдання у вигляді звіту з відповідними скриптами, якщо їх використовували;
- 3) Зробити відповідні висновки;
- 4) Звіти подавати в електронних файлах у форматі .pdf розміром до 5мБ та не більше 2 штук.