Project #6: Using *ssh*, *scp* and *sftp* with Key-Based Authentication

- ssh, scp and sftp
- Going beyond Password Protection
- Creating Keys
- Creating Keys with a Passphrase
- Using Key-Based Authentication in Our Lab

ssh*, *scp* and *sftp

- *ssh*, *scp* and *sftp* are utilities that come with Unix
- Each of them is a secure version of other utilities that were used before the Internet became insecure

text

- *ssh* allows to login to another, remote machine from your local machine
- It is a secure version of *telnet*
- *scp* allows you to copy a file to or from

ssh*, *scp* and *sftp

your local machine from or to a remote machine

- It is a secure version of *rcp*
- *scp* will **not** allow you to copy /etc/shadow because that file contains encrypted password data
- *sftp* stands for **S**ecure **F**ile **T**ransfer **P**rotocol
- *sftp* is like *scp* because you can use it to

ssh, scp **and** *sftp*

get files. It is a secure version of *ftp*

- Although both *scp* and *sftp* can be used to transfer files between machines
- *sftp* has more features
- When you use it you get something resembling a login to the remote machine
- You can move around in the hierarchical filesystem list the contents of directories

ssh, *scp* and *sftp*

and use **relative paths**

- All three programs encrypt the network traffic between the two machines

Going beyond Password Protection

- You have been using *ssh* using password protection
- That means you can only login to a machine with *ssh* if you have an account on that machine
- And before you are actually granted access you must enter the correct password
- Using *ssh* with a password is better than nothing but there are many programs that can break passwords

Going beyond Password Protection

- For better protection, you can use key-based *ssh* logins
- This technology uses two numbers which are used for public key encryption
- One number is public and is put on the remote machine
- The other key is private and is stored in the .ssh directory inside your home directory of the local machine the one you are using to connect

Going beyond Password Protection

- The two keys are used together, when you log in
- Key-based *ssh* logins **do not** encrypt the packets between the two machines
- Instead they protect your ability to log in
- Once key-based logins have been set up only machines that have the private key in ~/.ssh can connect to the remote machine
- This is much more secure than a password

Creating Keys

- You will be creating keys for your personal Unix account not the sysadmin account
- To create the two keys, use the *ssh-keygen* utility
- You have the choice of two different encryption algorithms
  - RSA - Rivest-Shamir-Adleman algorithm
  - DSA - Digital Signature Algorithm

Creating Keys

- You select the algorithm with the -t option
- RSA is thought to be the stronger of the two
- Your other option is to choose how long the keys will be
- The longer the key the harder it is to break
- By default, a key of 2048 bits is created

Creating Keys

- If you use the -b option, you can create a longer key
- To create a 4096 bit key using RSA encryption, you would enter the following at the command line

```
    ssh-keygen -t rsa -b 4096
```

- When you do this, two files will be created: id_rsa and id_rsa.pub
- Both files be in your ~/.ssh directory

Creating Keys with a Passphrase

- Using key-based ssh logins is more secure than using password authentication
- But if someone manages to steal you private key they could log in to your account on remote machines
- To give you an additional layer of security you could use a passphrase when creating your keys

Creating Keys with a Passphrase

- This way, if someone somehow gets a copy of your private key it would be useless to them until they crack the passphrase
- This gives you time to generate a new set of keys and spread the public one to the servers you connect to

Using Key-Based Authentication in Our Lab

- Key-based authentication is used only for connecting to a remote machine using one of the ssh utilities

  ssh ***scp*** ***sftp***

- To enable key-based authentication you must add your public key to the file authorized_keys in the .ssh directory in your home directory of the **remote machine**
- But since we are all sharing our home directories we need to add the id_rsa.pub public

Using Key-Based Authentication in Our Lab

key to authorized_keys in our home directory of our virtual machines

- We do this using

```
cat id_dsa.pub >> authorized_keys
```

You must use the append redirection symbol **>>** so you don't lose the previous contents of the file