Assignment 2. SSH setup and use in applications

Laboratory

Most likely you'll need to do this lab entirely on SEASnet, treating one of the SEASnet GNU/Linux hosts as if it were your personal machine, and treating the others hosts as remote servers. Alternatively, if you have a bunch of ssh-capable machines of your own, you can do the lab there. Either way, specify in your log which machines you used.

Use OpenSSH to establish convenient logins, so you should use ssh-agent on your host to manage authentication. That is, you should be able to log out of a host (dropping all your connections to it), then log back in, type your passphrase once to `ssh-agent`, and then be able to use `ssh` to connect to any of your other hosts, without typing any passwords or passphrases.

You should also use port forwarding so that you can run a command on a remote host that displays on your host. For example, you should be able to log into a remote host, type the command `xeyes`, and get a graphical window on your display. You will need to use `ssh -X` and/or `ssh -Y` when you use port forwarding.

Finally, you should be able to do a multihop or "daisy-chain" connection conveniently. For example, you should be able to `ssh` into `lnxsrv07`, and then from `lnxsrv07` to `lnxsrv09`, and so on. You want these daisy chains to be convenient as possible. On SEASnet this is merely an exercise, since you can connect to any one of the `lnxsrv*` hosts as well as any of the others; but in the real world these sorts of connections are useful when for security reasons one cannot connect to an inner server directly. Investigate Transparent Multi-hop SSH and see which technique works best for you.

Keep a log of every step you personally took during the laboratory to configure your or your team members' hosts, and what the results of the step were. The idea behind recording your steps is that you should be able to reproduce your work later, if need be.

Homework

On the SEASnet GNU/Linux servers, use GNU Privacy Guard's shell commands to create a key pair. Use GPG version 2. Export the public key, in ASCII format, into a file `hw-pubkey.asc`. Use this key to create a detached signature for your submission so that the commands described below can successfully verify it. The detached signature file should be named `hw-pubkey.sig` and it will be a signature for `hw-pubkey.asc`.

If you are creating a key pair on the SEASnet GNU/Linux servers, you may exhaust its entropy pool as described in Launchpad bug 706011. The symptom will be a diagnostic saying "It is a good idea to perform some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy." Since you can't use the keyboard or mouse on the SEASnet servers, you'll have to use the disks, for example, by using the `find` command to copy every readable file to `/dev/null`; this is something that you can do in another session that is logged into the same machine. Please remember to interrupt the `find` once the key pair is generated, so that you don't tie up the server unnecessarily.

Briefly answer the following questions.

1. Suppose the other teams really had been observing all the bytes going across the network in your lab exercise. Is your resulting network still secure? If so, explain why, and explain whether your answer would change if (1) you assumed the other teams had also tapped your keyboards after you completed all client-server setup and had thereby obtained your team's keystrokes during later operation, or (2) you are booting off USB and you assume the other teams temporarily had physical control of the USB. If not, explain any weaknesses of your team's setups, focusing on possible attacks by such outside observers.
2. Explain why the `gpg2 --verify` command in the following instructions doesn't really verify that you personally created the file in question. How would you go about fixing this problem?

Submit

Submit the following files:

1. The file `hw-pubkey.asc` as described above.
2. The file `hw-pubkey.sig` as described above.
3. The file `log.txt` which is a copy of your lab log.
4. The file `hw.txt`, which is a copy of your homework log. It will contain the commands used to generate hw-pubkey.asc and hw-pubkey.sig. Also include the answers to the two homework questions.

`hw.txt` and `log.txt` should both be ASCII text files with no more than 200 columns per line.

The following shell commands should work:

```
mkdir -m go-rwx .gnupg
gpg2 --homedir .gnupg --import hw-pubkey.asc
awk '200 < length' log.txt hw.txt # This line should output nothing.
```

The `gpg2 --verify` command should say "Good signature". The last `awk` command should output nothing.

© 2005, 2007, 2008, 2010, 2012, 2016–2018, 2020 Paul Eggert. See copying rules.
$Id: assign2.html,v 1.45 2020/04/05 01:19:49 eggert Exp $