

Open Source Security Tools for Information Technology Professionals

School of Professional Studies (SPS)

The City University of New York (CUNY)

Aron Trauring
Adjunct Professor
CEO, Zoteca

Class 4 October 17th, 2005

Class Agenda

6:30-6:45 Article of the Week

6:45-7:45 Review / Hardening Linux Lab

7:45-8:00 Break

8:00-9:30 SSH with SSH Lab

Instructor

Aron Trauring – Adjunct Professor, CUNY SPS / CEO, Zoteca

Email: atrauring@zoteca.com

Personal Website: <http://aronst.org/>

Zoteca Corporate Website: <http://www.zoteca.com/>

FOSS Resources: <http://www.fourm.info/>

Review — Hardening

Trusted Computer Base

- Need a base computer to run security tools
- Heart of the security operation must itself be as secure as possible
- Entire list of elements that provide security: OS, programs, hardware, physical protection, procedures
- Critical that the OS, which underlies everything, be totally secure

Minimalism and Hardening

- Only what is needed should be made available.
- Only permit access to what you really need.
- Unless explicitly permitted, access is strictly forbidden.
- Hardening means applying minimalist principles to your system.

General Hardening Tasks

1. Disable or remove all unneeded services
2. Run latest version of services that are required
3. Disable or remove all unused user accounts
4. Keep up-to-date with security patches
5. Check and maintain system logs
6. Restrict access to special services (e.g. cron) to root
7. Review and fix file permissions on critical system files
8. Review and fix general system file permissions (e.g. no world write)
9. Remove development tools if you can
10. Limit the number of daemon processes permitted
11. Run local firewalls and not just perimeter firewalls
12. Run a system hardening script
13. Use sudo

Securing Network Services

1. Only use the secure version of a service (SSH, SFTP)
2. Disable insecure protocols (r*, FTP, Telnet)
3. Prevent service processes from having access to data they don't need through virtualization
4. Specify logging and access control for all services giving them only the minimal rights necessary
5. Services should have special users and not run as root (if possible)
6. Drop packets at the router firewall so they never reach server (prevent DoS)
7. Harden network infrastructure
8. Address wireless issues

Hardening Linux Lab

Update Options

- apt-get update / apt-get upgrade — CLI
- System → Administration → Update Manager (just patches)
- System → Administration → Synaptic Package Manager (updates and additions)
- Repositories
- Places where packages are stored on the Internet

Synaptic Update/Upgrade

- Reload
- Mark All Upgrades
- Apply
- Choose “smart upgrade”

Ubuntu Components

- Main — free software supported by Ubuntu distribution
- Restricted — non-free software essential to Ubuntu (e.g. binary hardware drivers)
- Universe — free software from rest of Debian; no guarantees of security updates
- Multiverse — non-free software with no guarantees at all

Ubuntu two versions — stable and development

- Hoary Hedgehog — previous stable
- Breezy Badger — current stable (Oct release)
- Dapper Drake — development release (4.2006)
- 6 month or so increments

Changing Repositories, Versions, Components

- “sudo gedit /etc/apt/sources.list” and remove comments on universe and comment CD
- Choose Settings → Repositories
- Don’t forget to reload (apt-get update)
- Can’t run Synaptic and/or Update and/or apt simultaneously

Create root password

- “sudo passwd root”

Webmin

- Add: webmin, webmin-core, webmin-cpan, webmin-status, webmin-xinetd
- Start up Firefox
- `https://localhost:10000/`
- Login as root (never save password)

Hardening Webmin

- Ubuntu automatically sets it up as
- http secure SSL
- Only localhost, and with n
- New certificate (not necessarily so for other distributions)
- Only one user (root)

Users

Xinetd

- More secure version of inetd
- Internet daemons
- Basic internet services
- Xinetd allows you to restrict access / number of connections / number of processes
- Nothing turned on by default
- Ubuntu doesn't even install by default

System Logs

Cron

- Disable to other than root

Bootup and Shutdown

- What processes get started at boot

Running Processes

- More detail than ps or top
- Since on web need to refresh manually

Filesystem Permissions

- “ls -al /”
- “ls -al /etc”

Bastille Linux

- Jay Beale
- Install
- `bastille -X` version
- `bastille c` — curses version
- `bastille non-interactive [config-file]` — once configure

SSH — Secure Shell

Clear Text Problem

- Data in packets available to anyone to read — clear text, plain text
- How to share while keeping data confidential?
- Remote Administration — How to allow access without making it easy to break in?

Types of Encryption

- Symmetric/Shared Secret — same key used to encrypt and decrypt
- Problem: if key is intercepted you are wide-open
- Asymmetric/Public Key — different key used to encrypt and decrypt

Public Key Encryption

- Whitfield Diffie, Martin Hellman, Ralph Merkle — 1976
- Split key into two parts — private and public
- Sender encrypts message with recipients public key
- Recipient decrypts message with private key
- Advantage: two entities can communicate secretly without sharing a secret key
- Created by using one way functions — easy to compute in one direction but not the reverse
- Prime number factors — easy to multiply two prime numbers but no algorithm to factor

Practical Implementation of PKE

- Allows for both key generation and digital signature
- RSA — Ron Rivest, Adi Shamir and Len Adleman at MIT, 1977 (U.S. patent expired in 2000)
- RSA <http://en.wikipedia.org/wiki/RSA>
- DSA — Digital Signature Algorithm. US Federal standard, 1991.
- DSA http://en.wikipedia.org/wiki/Digital_Signature_Algorithm

SSH

- Based on PKE — uses either RSA or DSA (RSA default)
- Establishes a secure connection
- All traffic over connection is encrypted
- Client-server model
- OpenSSH — FOSS version available on all Unices and Windows (PuTTY)

Stages of Establishing the SSH connection

1. Client needs to establish that it is talking to the machine you asked it to, and not another machine that's spoofing it (or sitting in the middle accepting data and passing it on transparently)
2. Server on the remote machine may want to establish that you are connecting from the machine you appear to be, and not another machine that's spoofing it
3. Client and server exchange keys for encrypting all future traffic between them
4. Client needs to convince the server that you are who you claim to be, and are authorized to do things on the server

Host Authentication

- Hosts that support SSH arrange to have a host identification key consisting of a public and private part.
- Server sends its public key to the client
- Client encrypts a random session key with it and sends the result back to the server.
- Client clearly knows the session key as it generated it, and only the server can use its private key to decrypt the message the client sent to it.
- This both secures the session, and assures the client that it must be talking to the correct server machine

Server Check

- Client has a list of public keys for server machines (taken from `/etc/ssh_host_key.pub` on server)
- Per-machine in `/etc/ssh/ssh_known_hosts`
- Per-user `~/.ssh/known_hosts`
- If a public key is not locally held, get warning: "Host key not found from the list of known hosts. Are you sure you want to continue connecting (yes/no)?"
- If answer yes, the host public key will be added to that user's personal list of host public keys
- Minor danger of DNS spoofing

Client Challenge

- Administrator has included the public key for the client machine in the per-machine list on the server machine.S
- Server creates a "challenge" encrypted with the client's host public key
- Only if the client is the machine it claims to be will it have access to the correct private key which enables it to decrypt the challenge and send it back to the server

Encryption — Session Key

- Goal: to prevent any data (both authentication information, and subsequent session data) transferred (password, keystrokes and program output) from being snoopable
- Ssh client and server use exchange keys for one of a number of different encryption methods used to encrypt all subsequent data transfers in the session
- Attacker would need to know the key in order to decrypt any part of the session, and the keys are negotiated in an encrypted form that requires knowledge of one or other hosts' secret host keys.
- Various encryption methods are available (TBD)

User Authentication

- rhosts — checks if in .rhosts file and only does host authentication: not secure and not often used
- rhosts with client challenge — slightly better
- password — user enters password. Password sent encrypted. Usually don't allow root login.
- Public key encryption

PKE User Authentication

- User creates key pair using ssh-keygen
- Puts copy of public key (~/.ssh/id_ds.pub or id_rsa.pub) in server machine
- Per-machine in /etc/ssh/authorized_keys
- Per-user in ~/.ssh/authorized_keys
- Server creates a challenge encrypted with user's public key.
- If the client knows the corresponding private key, it can decrypt the challenge and send it back to the server, which
- Server now knows that the client must have the secret and should therefore be allowed access.

SSH Uses

- Remote login
- Port forwarding

SSH Lab

SSH Remote Login

- Install SSH server
- Login to remote (neighbor)
- Accept key
- Compare ~/.ssh/known_hosts to neighbor's /etc/ssh_host_rsa_key.pub
- Note addition of hostname and IP address
- If hostname, IP address or key is changed you get a warning

Authenticate via Public Keys

- `ssh-keygen -t rsa`
- Enter a passphrase
- Copy over to remote machine
- `scp ~/.ssh/id_rsa,pub username@IP address/hostname:~/.ssh/authorized_keys`
- `ssh username@IP address/hostname`
- Open text editor and copy and paste key
- Make sure text editor did not leave an automatic backup
- Remember: key is one, unbroken line
- Go into Webmin and change authentication parameters

SSH Agent

- Allows you to start a session and load keys only once
- Add key to two remotes
- `ssh-agent`
- `ssh-add`
- Now ssh to each of the remote servers
- Tied to particular bash shell. Will disappear when you log out (`exit`).
- `ps -T`

SSH Keychain

- Like ssh-agent except keeps keys in memory even if you logout
- Install keychain
- Add to ~/.bash_profile:

```
keychain id_rsa
. ~/.keychain/$HOSTNAME-sh
```

- Null passphrase allows unattended reboots but allows easier exploitation of private key
- Add line to cron jobs for passwordless cron:

```
source ~/.keychain/$HOSTNAME-sh
```

Tunneling X Over SSH

- Enable in Webmin
- ssh -X
- X is a hog
- VNC better alternative

Port Forwarding with SSH

```
ssh -f -N -C -T -l [username] -L[localport]:localhost:[remoteport] [server
hostname/IP address]
```

- -L option forwards stuff from localport on the client box to remoteport on hostname
- -f option allows ssh to run in background after it prompts for the login password
- -N means don't execute any remote command (like a shell initiation script). This is used because all we want is port forwarding.
- -C means use compression.
- -T disables a pseudo-tty allocation (again because all we want to do is port forwarding, not open a remote shell).
- -l is login as username
- On Macintosh use 127.0.0.1 instead of localhost

Port Forward Webmin

```
ssh -f -N -C -T -l [username] -L10001:localhost:10000 [server hostname/IP
address]
```

Port Forward VNC

- Install vncserver and vncclient
- On server:

```
vncserver -geometry 800x600 -depth 16 :1
```

- This starts VNC on port 5901
- On client:

```
ssh -f -N -C -T -l [username] -L5902:localhost:5901 [server hostname/IP  
address]
```

```
vncviewer [-shared] localhost:2
```

SSH File Permissions

- For user accounts in ~/.ssh, use the following permissions:

```
~/.ssh mode 700
```

```
~/.ssh/id_dsa and other private keys mode 400
```

```
~/.ssh/id_dsa.pub and other public keys mode 644
```

```
~/.ssh/ssh_config mode 644
```

```
~/.ssh/known_hosts mode 644
```

```
~/.ssh/authorized_keys mode 644
```

- Files in /etc/ssh should have these permissions:

```
/etc/ssh mode 755
```

```
/etc/ssh/sshd_config mode 644
```

```
/etc/ssh/ssh_config mode 644
```

```
/etc/ssh/ssh_host_dsa_key and other private keys mode 400
```

```
/etc/ssh/ssh_host_dsa_key.pub and other public keys mode 644
```

```
/etc/ssh/moduli mode 644
```