LPIC-1 TRAINING COURSE

Topic 110: Security

Contents

Objectives

- Review system configuration to ensure host security in accrordance with local security policies
- Setup a basic level of host security
- Use public key techniques to secure data and communication

1. Administering Network Security

Network Super Server

- Listen for network connection on behalf of another program
- Hands off control of that connection to intended server
- Help reduce memory load and improve security
- Linux two main super servers: **inetd** and **xinetd**
  - **inetd** use ***TCP Wrappers*** to handle security
  - **xinetd** has builtin security features
- Servers that normally use super server: **telnet, FTP, TFTP, rlogin, finger, POP, IMAP**

Configuring *inetd*

- Main configuration file: **/etc/inetd.conf**
  - Other configuration files are in **/etc/inetd.d/**
- Syntax:

  `service socket protocol {wait|nowait} user server parms`
  - **`service`** name of the service as described in **/etc/services**
  - **`socket`** can be **stream**, **dgram** or **raw**
  - **`protocol`** can be **tcp** or **udp**
  - **`wait/nowait`** **wait** for **dgram** socket and **nowait** for other socket types
  - **`user`** username used to run the server
  - **`server`** server binary file, usually the TCP Wrappers (**/usr/sbin/tcpd**)
  - **`parms`** parameters that are passed to the server
- Example:

```
ftp stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.ftpd
```

- Disable a service: add # at the beginning of line
- Restarting **inetd** service: `/etc/init.d/inetd {restart|reload}`

Controlling Access via TCP Wrappers

- Configuration file: **/etc/hosts.allow** and **/etc/hosts.deny**
  - If a computer is listed in both files, **hosts.allow** takes precedence
- Syntax:

  `daemon-list: client-list`
  - **`daemon-list`** list of servers using the names in **/etc/services**
  - **`client-list`** list of computers (name or IP) to be granted of denied access
- Example:

  >/etc/host.deny

  `ALL: ALL`

  >/etc/host.allow

  `ALL: localhost`

  `popd, imapd: 192.168.1. .luna.edu`

  `sshd, ftpd: 192.168.7. EXCEPT 192.168.7.105`

Configuring *xinetd*

- Configuration file: **/etc/xinetd.conf**
  - Files in **/etc/xinetd.d/** are included
  - Each server run via **xinetd** installs a file in **/etc/xinetd.d**
- Syntax of **/etc/xinetd.conf**
- Example:

```
service ftp
{
    socket_type  = stream
    protocol     = tcp
    wait         = no
    user         = root
    server       = /usr/sbin/in.ftpd
    disable      = no
}
```

- Disable a service: change **disable** to **yes**
- Restarting **xinetd** service: **/etc/init.d/xinetd {restart|reload}**

Controlling Access via *xinetd*

- Using the following options in ***/etc/xinetd.conf***
  - Listen to only one network interface for the service:
    `bind = <IP Address>`
  - Accept connections only from IP addresses (similar to TCP Wrapper's **host.allow**)
    `only_from = <IP Addresses|Network>`
  - Deny connections only from IP addresses (similar to TCP Wrapper's **host.deny**)
    `no_access= <IP Addresses|Network>`
  - Set times during which users may access the server:
    `access_times = hour:min-hour:min`

Disabling Unused Servers

- Search for unused servers/services:
  - Look for open ports on a computer:
    `netstat -ap`
  - Listing all open internet/network files
    `lsof -i [4|6][TCP|UDP][@host][:service|port]`
  - Using remote network scanners (**nmap** or **nessus**)
    `nmap {-sT|-sU}` **`hostname`** **`|`** **`IPAddress`**
  - Examining configuration files: SysV startup scripts, inittab, **inetd/xinetd** configuration files
- Uninstall or reconfigure servers:
  - Disable the server in startup scripts or **inetd/xinetd**
  - Completely uninstall the server

Exercise 1

1. Verify that **xinetd** is installed and running
   - If not, install **xinetd** from installation .iso file and start **xinetd**
2. Install **telnet-server** from installation .iso file
3. View **xinetd** configuration file and directory. Check to see that telnet server configuration file is installed? Is this **telnet** server enabled or disabled
   - **/etc/xinetd.conf**
   - **/etc/xinetd.d/telnet**
4. Find the default port for **telnet** and in **/etc/services**
5. Can you open a telnet session to your localhost? Why?
6. Edit the telnet server configuration file you found in step 3 to enable it. Reload **xinetd** to re-read the configuration file
7. Retry openning a telnet connection to your localhost. Does it work?
8. Edit the telnet server configuration file to allow access only from 8:00AM to 9:00AM. Reload **xinetd**
   - `access_times = 8:00-9:00`
9. Re-openning a telnet connection to your localhost. Does it work?

Exercise 2

1. (As **root**) List all open TCP port in your system using **lsof**. Is there any **telnet** server running?
   - `lsof -i TCP`
2. Using **nmap** to scan all opened TCP port in your computer. Is the result the same as **lsof**?
   - `nmap -sT localhost`
3. Uninstall **telnet-server** and restart **xinetd** service
4. Rescan all opened TCP port in your computer. Is the telnet server's port closed?
5. Disable telnet server from **xinetd**
6. Verify your work with **lsof** and **nmap**

2. Administering Local Security

Securing Passwords

- Use strong passwords
- Change passwords frequently
- Use shadow passwords
- Keep passwords secret
- Use secure remote login protocols
- Be alert to shoulder surfing
- Use each password on just one system
- Be alert to social engineering

Limiting *root* Access

- Avoid logging in directly as **root**
  - no record of who typed the password
  - **root** password can be intercepted in various ways
- Run a single program with root previleges

```
su –c “command”      #require root’s password
sudo command         #requires user’s password
```

  - **sudo** configuration file: **/etc/sudoers**
    - Must edit this file via **visudo**
    - Syntax: **man 5 sudoers**
    - Example of **/etc/sudoers**:

```
Cmnd_Alias STORAGE = /sbin/fdisk, /bin/mount, /bin/umount
Cmnd_Alias PROCESSES = /bin/nice, /bin/kill, /bin/killall
%sys ALL = STORAGE, PROCESSES
%disk ALL = STORAGE
%wheel ALL = (ALL) ALL
```

Setting Login, Process and Memory Limits

- Done through Pluggable Authentication Modules (PAM) module called **pam_limits**
  - Editing ***/etc/security/limits.conf***
    - Syntax:
      `domain  type  item  value`
      - **`domain`** entity to which the limit applies (user, group, *)
      - **`type`** **hard** or **soft** limit, **-** for both hard and soft
      - **`item`** type of item is being limited (**core, data, fsize, nofiles, rss, stack, cpu, nproc, maxlogins, priority**)
      - **`value`** value that's to be applied to the limit
    - Example:
      `@limited  hard  cpu  2`
- Done (temporarily) through **ulimit** command
- ***/etc/nologin***: if present, only **root** may login

Locating SUID/SGID Files

- What if **rm** program's SUID bit was set?
- Search the entire computer for SUID and SGID programs:
  `find / -perm +6000 -type f`
- Use **chmod** to unset SUID/SGID from inappropriate programs

Exercise

1. Search your system for all files with SUID/SGID bit set. Is there any inapropriate file? Reset its SUID/SGID bit if needed.
   - **`find / -perm +6000 -type f`**
2. Create a new normal user named **linux**. Can this user use **sudo** to run **ifconfig**?
3. Edit **/etc/sudoers** to allow this **linux** user running any command as **root** via **sudo**. Verify your work.
   - **`linux ALL = (ALL) ALL`**
4. Configure your system to not allow this **linux** user logging in for more than 2 sessions. Verify your work.
   - **`echo “linux hard maxlogins 2”>> /etc/security/limit.conf`**

3. Configuring SSH

SSH Basics

- Telnet, FTP, VNC and X transfer data in unencrypted form
- SSH employ strong encryption techniques for all parts of network connection
  - Also provides file transfer feature
  - Able to tunnel other network protocols
- OpenSSH is the most popular SSH server
  - May be launched via a super server (**inetd/xinetd**) or startup script

Configuring Basic SSH Features

- OpenSSH configuration file: **/etc/ssh/sshd_config**
  - Large number of SSH options are commented out with default values
  - Some important options:
    - **`Protocol`** Safest configuration is to set Protocol 2
    - **`PermitRootLogin`** Enable/disable accepting direct login by root
    - **`X11Forwarding`** Enable/disable OpenSSH's X tunneling
    - **`AllowTcpForwarding`** Enable/disable OpenSSH's tunneling for TCP
  - For more information:
    `man sshd_config`
  - Reload **`sshd`** after reconfigured:
    `/etc/init.d/sshd reload`

SSH Keys

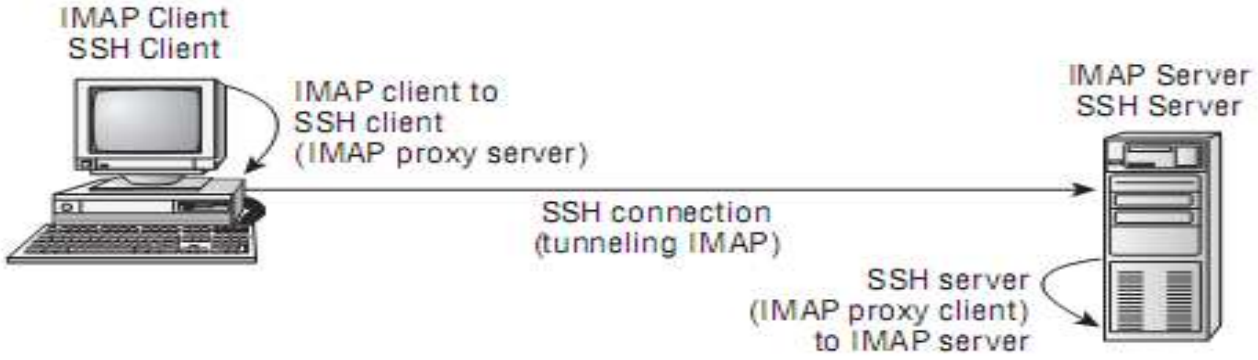
- Public key vs. private key
  - Data encrypted with a particular public key can only be decrypted with the matching private key
- **ssh-keygen** is used to generate public and private key
- OpenSSH server keys are stored in **/etc/ssh**
  - **ssh_host_rsa_key** vs. **ssh_host_rsa_key.pub**
  - **ssh_host_dsa_key** vs. **ssh_host_dsa_key.pub**
- When establishing a SSH connection, each side sends its public key to the other
  - SSH clients typically retain the public keys of servers they've contacted in **/etc/ssh/known_hosts** or **~/.ssh/known_hosts**

Controlling SSH Access

- Password authentication
- **TCP Wrappers** (if you run SSH from a **inetd** super-server)
- Firewall: SSH uses TCP port 22
- ***/etc/nologin***: If this file present, OpenSSH accept only **root** login

Using SSH

- Loging to remote system:
  `ssh user@remotesystem`
- Copying file via SSH:
  `scp filename user@remotesystem:/path`
- Using X11 forwarding:
  `ssh -X user@remotesystem`
- Using SSH Port Tunnelling for other TCP protocol:
  - Change **`AllowTcpForwarding`** to **`yes`** in **/etc/ssh/sshd_config**
  - Establish a special ssh connection to the server:
    `ssh –N –f –L lport:remotesvr:rport user@remotesystem`
    Example: **`ssh -N –f –L 142:mymailsvr:143 benf@mymailsvr`**
  - Configure the client program to connect to the local port



Configure Logins without Passwords

- Setup SSH client with keys and give the public key to the server computer
  - Should do only from a client that's very well protected
- Steps:
  1. Log into SSH client and generate SSH key:
     `ssh-keygen -q -t rsa -f ~/.ssh/id_rsa -C '' -N ''`
  2. Transfer the **~/.ssh/id_rsa.pub** from SSH client to SSH server (using **scp**, for example)
     `scp ~/.ssh/id_rsa.pub root@myserver:/~/.ssh/`
  3. Log into SSH server and add the contents of the **id_rsa.pub** file to the end of **~/.ssh/authorized_keys** file
     `cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys`

SSH Security Considerations

- Accept only protocol level 2 connections
- Refuse direct **root** login
- Disable X forwarding if unused
- Use TCP Wrappers or firewall to limt the machines that can contact an SSH server
- Keep SSH up to date
- SSH private key files should be well-protected

4. Using GPG

What is GPG

- GNU Privacy Guard
  - Open source re-implementation of the proprietary PGP (Pretty Good Privacy)
  - Based on public/private key infrastructure
  - Encrypting sensitive data such as email
    - Both sender and receiver must have GPG software
  - Enable digitally signing messages
    - Message can be read by recipients who lack the GPG software
    - Those who have these tool can verify that the contents haven’t been tapered with

Using GPG

- Generate keys (keys are stored in ***~/.gnupg/***):
  `gpg --gen-key`
- Export your public key to file for publicing
  `gpg --export name > publickey.pub`
- Import public key from other person:
  `gpg --import filename`
- View your available key:
  `gpg --list-keys`
- Encrypting data with a public key:
  `gpg --out encrypted-file --recipient uid --armor --encrypt original-file`
- Decrypting data:
  `gpg --out decrypted-file --decrypt encrypted-file`
- Signing message:
  `gpg --clearsign original-file`
- Verifying message:
  `gpg --verify recived-file`

Thank You !

BACKUP SLIDES

/etc/xinetd.conf syntax

```
service Service-name
{
   id = service-id
   type = INTERNAL/RPC/TCPMUX/TCPMUXPLUS/UNLISTED
   disable = yes/no
   socket_type = stream for TCP and dgram for UDP
   protocol = valid protocol from /etc/protocols
   wait = yes for singe thread (udp), no for multithread (tcp)
   user = the user the application runs as
   group = the group the application runs as
   server = the name of the program to be run for this service
   server_args = the arguments passed to the server
}
```