Bastion Hosts

CYBR371: System and Network Security, (2024/T1)

Arman Khouzani, Mohammad Nekooei
Slides modified from "Masood Mansoori"

13 May 2024

Victoria University of Wellington – School of Engineering and Computer Science

Bastion Host

Device that sits on the network perimeter. Has been specially protected through OS patches, authentication, encryption, etc.

Steps in creating and hardening a bastion host:

- Select a machine with sufficient memory & process speed.
- Choose and install OS and any patches or updates.
- Determine where the bastion host will fit in the network configuration.
- Install services you want to provide.
- Remove services and accounts that aren't needed.
- Back up the system and all data on it.
- Conduct a security audit.
- Connect the system to the network.

Selecting the Bastion Host Machine

Location on the Network:

- Typically located outside the internal network.
  - Combined with packet-filtering devices.
- Multiple bastion hosts are set up in the DMZ.

Linux Security

Linux Architecture

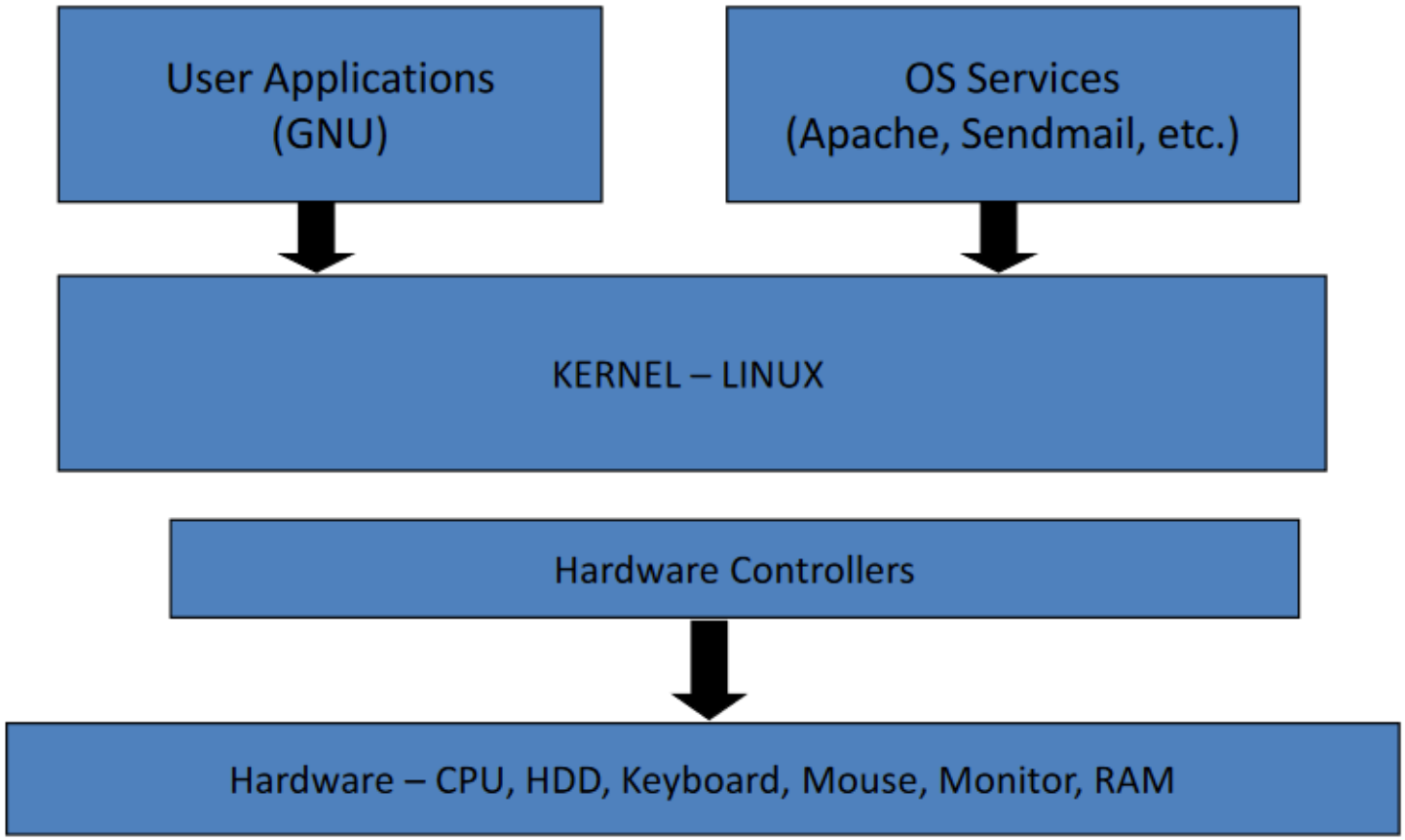
Linux Kernel: the actual code that interfaces between user applications and hardware resources.

Hardware controllers: used by the kernel to interact with hardware.

Operating System Services: software other than the kernel that are considered part of the OS. E.g., X Windows system, command shell.

User Applications: software other than kernel and services: text editors, browsers, etc.

Linux Architecture



Linux Kernel

It is separately distributed from user applications and other software

Uses modules, which can be dynamically loaded

- **# lsmod**
- **# modprobe <Module name>**

For instance, support for FAT32 need not be fixed, but can be added dynamically.

Kernel can be completely recompiled and unnecessary components can be removed - unlike Windows.

Kernel Security

One of the most important ways to keep Linux secure is to ensure a patched kernel.

Check your kernel version:

- `# uname -a`

Physical Security

- Server closet: Secured room to store servers.
- Limit access to computer itself.
- Remove CD-ROM devices from workstations.
- Ensure BIOS prevents booting from USB ports.
- Ensure BIOS password is set.

Boot Security

- Boot configuration is decided by LILO (Linux Loader) or GRUB (Grand Unified Boot Loader).
- Set boot loader password in LILO or GRUB configuration file **`/etc/grub/grub.conf`**.
- Check that only one OS is configured to load.
- If required ensure there is an entry for **`password=`** in **`grub.conf`**.

Linux Auditing

Linux Auditing

In most Linux distributions all global activities including startup messages are save in:

- **/var/log/syslog**
- **/var/log/messages**

Kernel events, errors, and warning logs, helpful for troubleshooting are saved at:

- **/var/log/kern.log**

Hardware including driver related issues:

- **/var/log/dmesg**

Linux Auditing

All boot related messages including boot failures, unexpected or unplanned shutdown or reboot:

- **`/var/log/boot.log`**

Information about scheduled tasks (cron jobs) are saved at:

- **`/var/log/cron`**

Linux Auditing

All security-related events such as logins, root user actions are saved at:

- **`/var/log/secure`**
- **`/var/log/auth.log`**

Logs all the failed attempts for login to the system:

- **`/var/log/faillog`**

Linux Auditing

Linux auditing is done using **`syslog`** daemon. Syslog is a standard for creating and transmitting logs.

Configuration file is **`/etc/syslog.conf`**. Format is:

```
Facility.Priority           Action
```

- **Facility** – the application/program that is generating the logs
- **Priority** – `emerg`, `alert`, `crit`, `err`, `warning`, `notice`, `info`, `debug`, `none`
- **Action** – send it to a file, send it to console, send it via email, send it to another system (loghost).
- Segregation of responsibilities – send logs to another system, where the security administrator has control.

Linux Auditing: syslog format and example

Message

```
May 8 18:11:45 sshserver sshd[223456] : Failed password
    for root from 130.195.3.23 port 22 ssh2
```

Adding More Info

```
<%pri%>%protocol-version% %timestamp:::date-rfc3339% %
    HOSTNAME% %app-name% %procid% %msgid% %msg%n
```

Will generate

```
<36>1 2022-05-05T18:11:45.003Z sshserver sshd - -
    pam_unix(sshd:auth): authentication failure; logname=
     uid=0 euid=0 tty=ssh ruser= rhost= 130.195.3.23
```

Linux Auditing: Important tools/commands to work with logs

Recent logins:

- **`last`**

Last login time for all users (dormant users):

- **`lastlog`**

Last failed logins (requires to create `/var/log/btmp` file):

- **`lastb`**

Tools for Log Analysis:

- **`Swatch`**: real-time monitoring of logs
- **`Logsentry`**
- **`Logwatch`**

Linux Network Security

Network Security: The Basics

1. Regularly scans ports on network computers
    - Know what ports and services are open/running
2. Minimize number of running network services
3. Use TCP Warppers, superdaemons xinetd/inetd
4. Check and verify trusted hosts
5. Do not run network services as root
6. Disable shell remote login for network daemons
7. Ensure using secured protocols
8. Monitor and test your firewall rules
9. Monitor and test your IDPS regularly

1 - Open Ports and Services

Find out the running network services and the open ports

netstat is used to display very detailed information about individual network connections, overall and protocol-specific networking statistics, and much more

Network Security: netstat example

Find out which process is using a particular port:

- **# netstat -an | grep ':80'**

Use the **-p** option to see which processes are responsible for which open ports

- **# netstat -anp**

List all tcp ports using netstat –at

```
# netstat -at
Active Internet connections (servers and established)
Proto       Recv-Q      Send-Q      Local Address       Foreign Address     State
Tcp         0           0           localhost:30037     *:*                 LISTEN
Tcp         0           0           localhost:ipp       *:*                 LISTEN
Tcp         0           0           *:smtp              *:*                 LISTEN
Tcp6        0           0           localhost:ipp       [::]:*              LISTEN
```

2 – Minimise Running Services

The `/etc/services` file contains a list of network services and the ports to which they map.

Services are configured by individual files in `/etc/xinetd.conf/` and `/etc/inetd.conf`

- Close/Disable services not needed

3 – TCP Wrapper and Super Daemons

TCP wrapper: program that can start a network daemon

When a client attempts to connect to a network service on a remote system, the following configuration files are used to determine whether client access is allowed or denied.

- **`/etc/hosts.allow`**
- **`/etc/hosts.deny`**

E.g., To allow clients on the 192.168.2.0 subnet to access FTP (daemon is **`vsftpd`**):

```
vi /etc/hosts.allow
vsftpd : 192.168.2.*
```

3 – TCP Wrapper and Super Daemons

The xinetd/inetd daemon are TCP-wrapped super services which control access to a subset of popular network services, including FTP, IMAP, and Telnet.

3 – TCP Wrapper and Super Daemons

Advantages of TCP Wrappers

- They listen to multiple ports and invokes only requested services and reduces the load that services place on a system
- Services such as FTP, Telnet, SMTP can be activated on demand rather than having to run continuously
- Transparency
- Centralised management of multiple protocols
- Additional features (e.g. ACLs, additional filters)

`Xinetd` vs. `Inetd`

- Logic change from earlier `inetd.conf` file.

Build-in controls similar to other TCPWrappers and more:

- **Access_control:** which hosts are allowed to connect and at what times
- **Logging:** which data gets logged
- **Resource utilisation:** limits on maximum connections supported, CPU usage, etc.
- **Redirection**

3 – TCP Wrapper and Super Daemons

Services are started by `/etc/rc.d` scripts and `xinetd`

- **`chkconfig –list`**
- **`chkconfig levels {numbers} {service} on|off`**

Close/Disable services not needed, in the `xinetd.conf` and `inetd.conf` files.

- **`/etc/xinetd`**
- **`/etc/inetd`**

4 - Check and verify trusted hosts

Besides `/etc/hosts.deny` and `/etc/hosts.allow` files, Entries in `/etc/hosts.equiv` and `/etc/hosts.lpd` are critical

- They allow users from those hosts to connect remotely without supplying a password!

Also, users can create .rhosts and .netrc files in their home directories, which function similarly. Find these as well

- **`# sudo find / -name rhosts`**

`/etc/hosts` and **DNS**?

5,6 - Run network services as root (NOT!)

Ensure network service daemons for essential services not run as root user when possible

Ensure that shell listed in **`/etc/passwd`** for daemons is set to **`/sbin/nologin`**

- Intruders will not be able to get BASH shell

Network Security – Remote Access Protocols

Telnet and **FTP** vs. **SSH**, **`rcp`** vs. **`scp`**

- Telnet and FTP are plain-text protocols
- Any inside user can sniff the traffic, even on switched networks with relative ease
    - Should be replaced by SSH
- SSH support a large number of encryption algorithms to provide services equivalent to Telnet and FTP
- Configuration is in **`/etc/sshd/sshd_config`**
- SSH clients are available for free, **`putty`** for Windows, now built-in in Windows 10, 11

(Linux) User and Group Security

User Security

- No user must login directly as **root**.
- Administrators must login with their own accounts, and then use **su** to become **root**.
  - This ensures accountability.
- Viable alternative is the **sudo** utility, which allows:
  - Listing of privileged accounts.
  - Actions that can be taken by these accounts.
  - Time out of logged in user, so they have to re-authenticate in order to use **sudo**.
  - Minimise **root** user's time logged in.

Users and Groups

Linux understands `Users` and `Groups`.

- A user can belong to several groups.
- A file can belong to only one user and one group at a time.
- A particular user, the superuser "**`root`**" has extra privileges (**`uid = "0"`** in `/etc/passwd`).
- **Only root can change the ownership of a file.**

Users and Groups Security

- User accounts are created in `/etc/passwd`.
- Hashed passwords, password and account lockout policies are in `/etc/shadow`.
- Password and account lockout policies can be set during account creation, or with the **`chage`** command.
- Check user information using **`finger`** package.

Users and Groups Security: `shadow` file format

1. **Username**
2. **Password's salted hash: `$id$salt$hashed`**, where **`$id`**:
   - **`$1`**: MD5
   - **`$2a`**: Blowfish
   - **`$2y`**: Blowfish
   - **`$5`**: SHA-256
   - **`$6`**: SHA-512
3. **Last-changed:** Days since Jan 1, 1970 that password was last changed.
4. **Minimum:** The minimum days required between password changes.
5. **Maximum:** The maximum days the password is valid (after that user is forced to change his/her password).
6. **Warn:** The number of days before password is to expire that user is warned that his/her password must be changed.
7. **Inactive:** number of days after password expiry that account is disabled
8. **Expire:** days since Jan 1, 1970 that account is disabled i.e. an absolute date specifying when the login may no longer be used.

Users and Groups Security: Password and Account Lockout

- Other stronger policies require use of **PAM**: **Pluggable Authentication Modules**.
- PAM allows the following to be set:
  - Minimum password length.
  - No dictionary words.
  - No part of username in the password.
  - Number of alphanumeric and punctuation characters to be present.
- PAM is configured in the **`/etc/pam.d`** folder.
- DEMO: change of password for user `auditor`.

Users and Groups Security

Group information is in **/etc/group**

/etc/passwd and **/etc/group** divide data fields using "**:**"

```
# cat /etc/passwd joeuser:x:1000:1000:Joe User„,:/home/joeuser:/bin/bash
```

```
# cat /etc/group joeuser:x:1000:
```

Check the groups memberships of a user:

```
# groups <username>
```

Format:

1. Groupname
2. Password
3. GroupID (GID)
4. Other members of the group separated by "**,**"

Application Security

A program may be run by a user, when the system starts or by another process.

Before the program can execute the kernel inspects several things:

- Is the file containing the program accessible to the user or group of the process that wants to run it?
- Does the file containing the program permit execution by that user or group (or anybody)?
- In most cases, while executing, a program inherits the privileges of the user/process who started it.

Processes and System Services

Check the processes running on a system:

- **ps –aux**
- **ps –aux | grep –i <name>**

Kill a specific process:

- **sudo kill <processID>**
- **sudo killall –I <processname>**

Stop a service temporarily:

- **sudo service ssh/sshd status/stop/start/restart**

Running a Process

- Every process “runs as” some user.
  - extremely important that this user is not **`root`** since any bug can compromise the entire system.
- may need root privileges, e.g. bind port.
  - have root parent perform privileged function.
  - but main service from unprivileged child.
- User/group used should be dedicated.
  - easier to identify source of log messages.

Running in chroot Jail

- **`chroot`** confines a process to a subset of **`/`**
  - maps a virtual **`/`** to some other directory
  - useful if have a daemon that should only access a portion of the file system, e.g. FTP
  - directories outside the chroot jail aren't visible or reachable at all
- Contains effects of compromised daemon.
- Complex to configure and troubleshoot.
  - must mirror portions of system in chroot jail

Access Rights

- We have covered these!
- Files are owned by a user and a group (ownership).
- Files have permissions for the user, the group, and other.
- ”other” permission is often referred to as ”world”.
- The permissions are Read, Write and Execute (r, w, x).
- The user who owns a file is always allowed to change its permissions.

Access Rights

- **`Suid/Sgid`**: Check very carefully. Especially when the file is owned by **`root`**.
- **`.rhosts`** file (open **`r-services`**)
- **`# sudo find / -perm +4000`**

File Integrity

File Integrity can be verified:

- Size and timestamp: can be modified to fool the auditor.
- MD5 hashes: secured method, but tedious.
- File Integrity Software:
  - Must be used immediately after the installation.
  - Create a database of MD5 hashes of all critical files.
  - Monitor changes to these files and send alerts.
  - ▶ **`Tripwire`**: commercial, scalable, central console.
  - ▶ **`AIDE`**: open-source, reasonably enterprise-level.

Conclusion

- Linux is not secure in default configuration.
- Security can be added to a very high level, but must be balanced with functionality.
- The correct Linux distribution must be chosen, and minimum installation done.
- Patches must be diligently applied.
- Syslog logs must be exported and analysed periodically.
- Network Services must be kept to a minimum.
- User and groups must be periodically audited.
- File/folder access control lists must be set.
- File Integrity software may be used in high-security installations.
- Application-specific security measures are also a must.

Next: Zero Trust Architecture and Micro-Segmentation