Linux Security

Overview

- This presentation will introduce you to some basic Linux concepts with an in-dept explanations from the security point-of-view
  - Services
  - Filesystem permissions
  - Login
  - Administrative permissions
  - Logging
  - SSH
  - Open ports

Services

- Services are handled by **systemd**
- **systemd** is an init system used in many Linux distributions to bootstrap the user space and manage system processes after booting.
  - It handles both system and user services
- **`systemctl`** is a command-line utility in Linux that is used to examine and control the systemd system and service manager.
- Here are the key aspects of systemctl:
  - Service Management
  - Log Viewing
- Historically, there were other service handlers
  - e.g., sysV, upstart, etc.

systemctl

- systemctl invoked with no params gives an overview of the whole operating system. It lists all the units (mainly services) of the system and tells their state

```
root@nginx:~# systemctl
  UNIT                                        LOAD   ACTIVE SUB       DESCRIPTION
  -.mount                                     loaded active mounted   Root Mount
  dev-.lxc-proc.mount                         loaded active mounted   /dev/.lxc/proc
  dev-.lxc-sys.mount                          loaded active mounted   /dev/.lxc/sys
  dev-full.mount                              loaded active mounted   /dev/full
  ifupdown-pre.service                        loaded active exited    Helper to synchronize boot up for ifupdown
  ifupdown-wait-online.service                loaded active exited    Wait for network to be configured by ifupdown
  networking.service                          loaded active exited    Raise network interfaces
  nftables.service                            loaded active exited    nftables
  nginx.service                               loaded active running   A high performance web server and a reverse proxy server
  postfix.service                             loaded active exited    Postfix Mail Transport Agent
  postfix@-.service                           loaded active running   Postfix Mail Transport Agent (instance -)
  rsyslog.service                             loaded active running   System Logging Service
  ssh@2-10.0.40.6:22-10.0.10.2:61209.service loaded active running   OpenBSD Secure Shell server per-connection daemon (10.0.10.2:61209)
```

systemctl

- Other functions can be accessed with `systemctl <command> <unit>`, where the unit is the name of the unit (like a service, socket, device, mount, d-bus, etc.) to be managed and the command can be one of the following:
  - `start`: starts a service.
  - `stop`: stops a running service.
  - `restart`: restarts a service, stopping it and then starting it again.
  - `status`: displays the current status of a service.
  - `enable`: enables a service to start on boot.
  - `disable`: disables a service from starting on boot.
  - `reload`: reloads the configuration of a service without interrupting its operation.
    - Not all the services supports this command.

Unit files

- When invoking `systemctl`, it lists all the units in the system. But how can systemd know all the units? How are they configured?
- Each unit has a unit file
- Unit files are located into the `/lib/systemd/system/` and the `/etc/systemd/system` directory
  - The first one represents the DEFAULTS units provided by the package maintainer
  - The second one represents a copy that can be edited to change the behaviour
  - DO NOT EVER MODIFY THE FILES IN THE `/lib/systemd/system/` directory
- Each file represents a unit and contains all the information of the unit itself
  - Environment variables/files
  - Startup/shutdown scripts
  - Unit dependency
  - and a lot more!

Services configurations

- Often located in a subfolder of the `/etc` folder
- The subfolder is often the name of the service
- If this is not the case,
  - they can be specified in the unit file
  - they can be read in the process command line (`ps aux | grep PID`)
  - you can always ask google ☺
- Often inside the main configuration file there is an “include” to other files
- Directories called `*.d` are often use to keep configuration files to be included

Change service configuration

- A change to the service configuration is never applied directly
- Usually, the configuration is applied at the startup of each service and does not change unless the administrator instructs the service to reload the configuration
- To properly change a service configuration:
  - Change the configuration file
  - Test the file for syntax (or logical) errors
    - Look for proper commands online
  - Reload or restart the service to apply the configuration

Filesystem permissions

- The main folders of the filesystem have some "default" permissions
- Some rules often applies
  - The `/etc` directory can be browsed by any users, but only the administrator can change its contents
    - The user needs to know the services configuration to understand how to interact with them and his limits
    - If a user was allowed to change a configuration, he could easily gain administrative access to the machine (services are executed by root)
      - Similar rules also applies for binaries
  - The users home directories are not accessible by other users
    - For privacy and security reasons

Login phases

The login process is composed of 4 main phases:

- Request for user credentials
  - e.g., username and password
- Check the database for the correctness of those credentials
- Change the current user and group to the newly logged user
  - setreuid() + setregid()
- execve of the shell
  - or of the desktop environment

The standard user database

- Users and their attributes in the /etc/passwd file
  - World readable
- Groups definitions in the /etc/group file
  - Contains also the list of all the users belonging to each group
  - World readable
- passwords stored in the /etc/shadow file
  - Readable only by root
- the `passwd` and `shadow` files are essential components of Linux and Unix-like operating systems, playing a crucial role in storing user account information

Passwd file

- Contains information about user accounts.
- World readable
- It's a text file with one line per user account, providing several pieces of information such as the username, user ID (UID), group ID (GID), etc.
- Example:
  - `username:x:1001:1001:User Name,,,:/home/username:/bin/bash`

Content of `/etc/passwd`

- Login name
- Encrypted password
  - No more used, but kept for compatibility reasons
- Numerical user ID (root is UID=0)
- Numerical group ID
- User name
  - also known as the GECOS field
- User home directory (es. /home/pizzonia)
- User command interpreter (es. /bin/bash)

Shadow file

- This file contains the password hashes for all users in the system.
- This file is accessible only to privileged users (like root).
- Each line in the shadow file corresponds to a user's password and contains several fields, including the username, hashed password, and password aging information
- The hashing of the password ensures that even if someone gains access to this file, they cannot easily decipher the actual passwords.
- During the hashing process, a salt is used to enhance security by introducing a random and unique value that is combined with the user's password before the hashing algorithm is applied.
- Example:
  - `username:$1$saltsalt$encryptedpassword:17958:0:99999:7:::`

Content of /etc/shadow

- Login name (foreign key /etc/passwd)
- Encrypted password
  - idsalt$hashedpassword
  - id: algorithm code
- Days since Jan 1, 1970 that password was last changed
- Days before password may be changed
- Days after which password must be changed
- Days before password is to expire that user is warned
- Days after password expires that account is disabled
- Days since Jan 1, 1970 that account is disabled
- A reserved field

Editing passwd and shadow files

- Directly editing the passwd and shadow file is highly discouraged
- The format is very rigid and even the smallest error could result in the system being unusable
  - Users can no more login
- There are lots of dedicated commands to manage users and their passwords
  - useradd and adduser
  - userdel and deluser
  - usermod
  - passwd

Authentication flexibility

- there are a lot of programs that need authentication
  - system programs e.g., atd, chfn, chsh, cron, cupsys, cvs, kdm, kdm-np, libcupsys2, login, passwd, ppp, samba, ssh, su, sudo, telnetd, xdm, xscreensaver
  - external software e.g., mysql, apache, ecc.
- there are many policies and ways to authenticate
  - local authentication e.g., /etc/passwd, /etc/shadow
  - central authentication e.g., radius, active directory, ldap
  - different policies e.g., something that I have, somewhere I am, something I know, something I am

Authentication flexibility

- It's impractical and often unfeasible for every piece of software to natively support every type of authentication mechanism that a system might use.
- As a result, software systems often rely on standard or widely-adopted authentication protocols, and may also provide interfaces for integrating additional, more specialized authentication services.

PAM

- PAM allows Linux systems to integrate a variety of authentication methods without modifying individual applications.
- Offers customizable authentication for different applications through simple configuration files.
- Compatible with all sort of authentication policies and types
  - e.g., password, biometrics, tokens, smart-card, etc.
- Enables or disables authentication methods dynamically without affecting system processes.
- Integral part of most modern Linux distributions, ensuring broad compatibility and support.

PAM: from the administrator point

- no need to have coding skills or edit every software
- possibly independent configuration for each application
- lots of authentication types possible
- multi-factor authentication

PAM: configuration example

```
/etc/pam.d/$ cat login
# PAM configuration for login
auth        requisite  pam_securetty.so
auth        required   pam_nologin.so
auth        required   pam_env.so
auth        required   pam_unix.so nullok
#auth       required   pam_permit.so
account     required   pam_unix.so
session     required   pam_unix.so
session     optional   pam_lastlog.so
```

Sudo and sudoers file

- Logging using root credentials (i.e., the credentials of the user called root) is generally discouraged
  - Usually, the root user has its login password NOT set, to avoid this behaviour
- To obtain administrator privileges the best practise is to use the `sudo` command
  - This allows to log all the accesses with administrator privileges and to limit the capabilities of single users
- The configuration of `sudo` is stored in the `/etc/sudoers` file
- In the sudoers file it is possible to specify lots of different configurations
  - e.g., some users could run sudo without typing the password
  - e.g., some users could be allowed to run just some commands as administrator without the ability to be root

Logging

- Logging is essential
- Linux logging can happen in two different ways
  - Direct file logging
  - Journald
- Historically the logging was handled by syslog
- In both cases, logs are written to files and those files in the time can grow very fast
  - The standard practice in Linux is to create an archive of old logs to reduce disk space
    - Logrotate is responsible for this task
  - If a user needs to inspect log files, it is better to use proper commands to avoid opening a very large text file
    - e.g., head, tail, grep, less, more, etc.

Journald

- Journald is a systemd unit designed to acquire, store and show logs
- Since the process of the services are all handled by systemd, their stdout and stderr are automatically captured by Journald to save the proper logs
- Journald is available on the vast majority of Linux systems
- Journald by defaults logs the kernel and all the applications (services) that logs to stdout and stderr
  - Default system utilities logs in Journald
- Journald has a command line utility called `journalctl`
  - journalctl can be used to get logs for a specific unit with a proper command
    - i.e., `journalctl -u <unit>`

File Logging

- Programmers may choose to not print the logs to stdout and stderr but to save them directly into a file
- Usually, log files are stored in the `/var/log` directory
- Programmers may choose to log wherever they want
  - Wherever the software has write access permissions
  - Usually, the log file location can be specified in the configuration file of the service

SSH

- The SSH (Secure Shell) server daemon is a crucial component of secure remote administration and file transfer in computer networks.
- Operating as a background process on a server, the SSH daemon enables secure communication between devices over an insecure network, such as the Internet.
- It utilizes strong encryption algorithms to ensure data integrity and confidentiality during the exchange of information.

SSH

- It utilizes strong encryption algorithms to protect data integrity and confidentiality during the exchange of information.
- The SSH server daemon listens for incoming connection requests on a specified port (port 22) and authenticates remote users or systems using various methods, including passwords, public-key cryptography, or other authentication mechanisms.
  - The password mechanism is enabled by default, but it is highly insecure since usually passwords are much shorter than public-key certificates
- Once authenticated, users gain access to a command-line interface, allowing them to execute commands, transfer files, and manage system configurations securely.

Open Ports

- Network ports are the primary entry points for attacks.
- Even with a firewall, it's advisable to close unnecessary ports.
  - Attackers may circumvent the firewall in various ways.
- When a program binds a port in listen mode, choosing the listening address is possible.
  - It's important to bind only to the correct address.
  - Generally, binding to 0.0.0.0/0 is poor practice unless the service needs to be available across all network interfaces.
  - Binding to the loopback interface (127.0.0.1) is a safer practice, as it restricts access to the local machine only.
- To check for open ports, use the command `netstat -tulnp`.