Computer Science Department

Computer Security – COMP 432

Group No: 10

UNIX OS Security

Name & ID:

- Ahmad Aljamal 1150409
- Mohammad Alqyeem 1150357
- Wafaa Abdel Mohdi 1151709

Instructor: **Dr. Hafez Barghouthi**

• Abstract:

In this paper we'll cover some of the services provided by UNIX based operating systems and how these services are implemented in the UNIX family of operating systems.

• Introduction:

The operating system of a computer or a network is the core of all communications and functionality, and since the operating system controls all of the resources and data on the system, and securing operating system security is a crucial component of an overall security program, and the prevalence of malware, viruses and root kits, increased the importance of OS security, and because of the major consequences that result if system has been attacked, like none of the key functions the OS does will be available and data would be compromised.

Since more than 64.9% of the Servers that run the internet are running Unix/Linux and more than 40.41% Desktop/Laptop run on a Linux kernel, Unix/Linux security is a hot topic.

Since the operating system controls a wide range of functionalities from local operation and encryption to networking and processes handling, to have a secure operating system, the OS should handle all of these aspects to ensure full security

• Literature:

In UNIX there are several layers of interaction which are occurring between the computer hardware and the user. The first layer is kernel, which runs on the actual machine hardware and deals with all the connections with the hardware. All the commands and applications in UNIX relate with the kernel, rather than the hardware directly.

The hardware constitutes the second layer. On top of the applications and commands is the command-interpreter program, there is another layer which is called shell which interacts between user, user's applications, and the existing UNIX commands.

Under the UNIX, the operating system consists of many of the utilities along with the master control program, the kernel. The kernel is a powerful program which helps the UNIX to start or stop program and handle the file system and other common lower level tasks which every programs shares.

- **Discussion:**

I. Identification & Authentication

The operating system must be able to distinguish between different users, and verify users claimed identities. Identification and authentication are important to other services in the system. Passwords are by far the most common authentication method in UNIX based operating systems.

UNIX based operating systems identify and authenticate users according to username/password. The OS provides restriction on passwords which have to be at least 8 characters; it stores passwords in an encrypted form using a modified version of DES algorithm in a file (/etc/passwd) according to the following format:

username:encrypted password:UserID:GroupID:user's full name:home directory:login shell.

Because the /etc/passwd file is available to all users on the system this would make the system vulnerable to brute-force attacks, that's why UNIX now implements shadow passwords which doesn't directly store the password in a user accessible file. Instead the system stores the password in /etc/shadow which can only be read by the root (some distributions add Password aging).

II. Access control

Standard UNIX systems prevent the unauthorized use of system resources (e.g., files, memory, devices, etc.) by promoting discretionary access control. Permissions are divided into three categories: owner, group, and other. However, privileged accounts can bypass this access control. UNIX treats all system

resources consistently by making no distinction between files, memory, and devices; all resources are treated as files for access control purposes (this is UNIX philosophy).

The UNIX file system has a tree structure, with the top-level directory designated as "/". Some of the second-level directories are standards. For example, "/bin" contains system executable, "/dev" contains devices, "/usr" contains user files, etc. Each directory contains a pointer to itself (the '.' file) and a pointer to its parent directory (the '..' file) these created when creation of this directory (executing mkdir command). In the top-level directory, the '..' file points to the top-level directory. Every file (and directory) has an owner, a group, and a set of permissions. UNIX identifies block devices (e.g., disks) with the letter 'b' and character devices (e.g., modems, printers) with the letter 'c'.

When a user or process creates a new file, the file is given default permissions (default mask or mode). For a process-created file, the process specifies the default permissions. For user-created files, the default permissions are specified in the startup file for the user's shell program. File owners can change the permissions (or mode) of a file by using the umask command.

Every file and every directory has 3 types of access, being read access, write access and execute access for 3 types of groups: user, group and other. The first group is the group of the owner of the file. The second group contains access rights for a group of users. The third set of access rights is for any other user (not being the owner and not belonging to the group having access rights to the file or directory). These types have the following values:

- Read access value 4
- Write access value 2
- Execute access value 1

Password commands "passwd" is a magic command, because it sets password for any user even the shadow file is for root; it has "s" in the permission in the user

domain “rws--xr-x”, this will make the effective uid will be equal to root, if “s” put in group domain “rwx--sr-x” so effective gid will be equal to admin.

Another use of this technique, if you write a command has “s” in group domain, so the same group members can use it to write to each other terminals, but writing directly blocked. So this technique can be exploited as vulnerability, to hack a specific user & takes control or stole data.

- **Changing file ownership**

Changing user or group ownership of a file is done with the GNU chown command (change owner). The owner of or any member of group of owners of the file can do any of the three access modes. Although both types of ownership are changed with the same command, they are independent of each other. E.g. you need not be a member of the group that owns the file in order to be able to change it. Your own group will be considered as "other", and if permissions allow, you can change the file.

User and group ownership can be changed in one command:

chown newuser:newgroup file

- **The root user**

So every file is owned by somebody. And so is every process. If you want to handle a file or a process, you have to be the owner. It is clear that some actions need to be undertaken to circumvent this situation. Who will clean up the mess? Who will modify the system files and services? On a UNIX system, this force is called the "super user" or "root".

The root account should always be protected with a password, and the root user is not obliged in any way to communicate this to the other users. This prevents people from reading each other's mail, from harassing other people and generally prevents a great deal of accidents.

The root user (system administrator) should only use the root status when necessary, and only when concentrated. Root status gives full control over the system, so you should be careful when "being" root. Should you need to become root, always log in as a normal user and then use the "su -" (switch user) command, which will give you root status when no options are given.

III. Availability & Integrity

One aspect of availability is whether a system restarts securely after failure. Traditional UNIX systems boot in single-user mode, usually as root. And, unfortunately, single-user mode allows literally anyone sitting at the system console to execute privileged commands. Thus, single-user mode represents security vulnerability in traditional UNIX. Depending on the flavor of UNIX, the security administrator has one or two options for closing this hole. First, if the operating system supports it, the security practitioner should configure the system to require a password before booting in single-user mode. Second, tight physical controls should be implemented to prevent physical access to the system console.

System restarts are also relevant to system integrity. After an improper shutdown or system crash, the UNIX fsck command will check file systems for

inconsistencies and repair them (either automatically or with administrator interaction). Using the fsck command, an administrator can detect unreferenced inodes, used disk blocks listed as free blocks, etc.

Although there are many ways to supplement UNIX file system integrity, one method has become so popular that it deserves to be mentioned here. Developed by Gene Kim and Gene Spafford of Purdue University, Tripwire is an add-on utility that provides additional file system integrity by creating a signature or message digest for each file to be monitored. Tripwire allows administrators to specify what files or directories to monitor, which attributes of an object to monitor, and which message digest algorithm (e.g., MD5, SHA, etc.) to use in generating signatures. When executed, Tripwire reports on changed, added, or deleted files. Thus, not only can Tripwire detect Trojan horses, but it can also detect changes that violate organizational policy.

IV. Audit

What to do when the system security bypassed? Can we know who did it?

Unix OS stores every security relevant event in the log files, if any suspicious of attacking signs found, then these log files can be checked for any suspicious actions that were done lately.

These log files can help reduce the attacking possibility because the attacker knows that the events are recorded and he will not take the risk of exposing his identity unless he is an expert. Also these files can be used for investigation any incidents and can be used as legal evidence in criminal trials.

The log files can be located on the system such as the directory /var/log/ which contains all log files, note that the location of the log files can be different depending on the Linux distribution used.

Some example of the log files:

- lastlog: records the last time a user logged in
- utmp: records information of the current logged in users and the status of the system
- btmp: records failed login attempts
- sulog: records all attempts to execute the su command

V. Security facilities for users

Of course each user on the system wants to be left alone and deny other users from configuring his files or create some files in the user directories that the user does not need. In order to do that, UNIX provides each file with its own permission and the user that created it, so each user is responsible of his files, and the user also cannot configure any files on other users' directories unless the other users want to. These configuration are applied to the normal users, the commands (chmod, passwd) are used to change the files permission and to change the user password.

Any OS must have an administrative account that can configure the system to what it is needed, Unix provides one administrative account called "root" account that have all privileges on the system that can do:

- Changing files permissions
- Delete any file
- Control users accounts
- Execute any program
- Shutdown the system
- etc.

If the system was been attacked and the attacker was able to access the root account, then the attacker has all the permissions that is needed to do whatever he wants on the system, this violates the separation of duties principle. Also when having the access of the root account, the OS will not ask for authorization when doing some actions, which violates the complete mediation principle.

• Conclusion

Traditional UNIX implements some of the components of operating systems security to varying extents. It has much well-known vulnerability; out-of-the box configurations should not be trusted. Furthermore, add-on security tools can supplement core UNIX services. With proper configuration, a UNIX system can be reasonably protected from would-be intruders or attackers.

Designing and implementing a truly secure program is actually a difficult task. The difficulty is that a truly secure program must respond appropriately to all possible inputs and environments controlled by a potentially hostile user. Developers of secure programs must deeply understand their platform, seek and use guidelines ,and then use assurance processes (such as inspections and other peer review techniques) to reduce their programs' vulnerabilities.

• References

- "How Unix Security Works". Tille.garrels.
- “INTRODUCTION TO UNIX SECURITY FOR SECURITY PRACTITIONERS”. Jeffery J. Lowder.
- https://www.ukessays.com/essays/information-technology/importance-of-unix-operating-system-information-technology-essay.php