**Assignment 2**

1. What is purpose of:
   1. On a Linux system, the /etc/passwd file is used to store user account information such as username, user ID, group ID, the name of the user, the home directory, and the preferred shell. It is called the passwd file because originally the encrypted user password was stored in the file as well. However, security concerns were raised and the password information now resides in the shadow file /etc/shadow.
   2. In modern Linux systems, the /etc/shadow file is used to store all password information for a user account. This information includes the username, the encrypted password, the number of days since the password has been changed, the number of days before the password can be changed, the number of days after which the user is required to change their password, and other password related information. Even though the one-way salted hash would be very difficult to crack, the /etc/shadow file is read/write protected for additional security. Only the root user is capable of reading or writing to the shadow file.
   3. In Linux, setuid (or SUID) is an abbreviated form of Set User ID. Normally, if a program were to be executed, the program would inherit the permissions from the executing user. If your user can only access files in your home directory, the program could only access files in your home directory as well. SUID, however, is a special type of file permissions which allows a program to inherit permissions from the executing user as well as the owner. Therefore, if you execute a program that is owned by root, the program can access the files in your home directory (your permissions) as well as any file that root can access (root’s permissions). The SUID attribute is important for files such as the /etc/shadow because it allows non-root users to run tools such as passwd which lets the user change their password.
   4. The chroot command changes the root directory of a process and all its children. If the root directory has been changed, the process is unable to access any files that exist outside of the new root directory and its subdirectories. For example, if you used chroot and changed the root directory for Process X, Process X can *only* access files from your home directory and subdirectories.
2. The ls command with the –l switch displays the long listing of information about a provided file or directory. The information includes native file system attributes such as the file type (link or directory), the owner, group, and user permissions, last modification timestamp, etc.. The lsattr command, however, displays information about a given file’s extended file system (e.g. ext2, ext3) attributes. Examples of extended attributes include append only, immutable, secure deletion, and undeletable.
3. In the Android system, applications must request permission from the user to access personal information and device functionality such as camera and GPS. The problem, however, is that permission requests are very broad. For example, an application may only need to read the user’s contact list, but the permission request will give the application permission to read, create, and delete contacts. If the application need only read contacts, the application should only be able to read contacts and nothing more. The challenge, however, is making the permission request process fluid while preserving the least privilege principle. It would be tedious and cumbersome to require the user to respond to three prompts (“Can Application X read contacts?”; “Can Application X create contacts?”; “Can Application X delete contacts?”) when an application wants to access contacts. Granular permissions allow a dedicated user complete control over a system, but a casual user may become frustrated by the tedium.
4. The standard UNIX permission model allocates read, write, and execute permissions to a file for the file owner, the owning group, and everyone else. This model works on a very basic level, but the model does not scale very well into, for example, an enterprise network of UNIX systems. The scaling issues can be resolved with Access Control Lists. An access control list allows permissions to be allocated to sets of users and sets of user groups. This is contrary to the standard model which only allows permissions to be set for a single group. Furthermore, suppose there is a directory that needs to be accessed by Quality Assurance agent and a Development team but no one else. The standard UNIX model does not allow for this type of permission allocation. An access control list, however, does by granting access to both the Development group and the Quality Assurance group.
5. When executing a program, the Real User ID (RUID) is set to the user ID of the user executing the program. In the case of a SUID, however, the Effective User ID (EUID) is the user ID of the SUID user ID. For example, in the case of the passwd tool, the RUID is your user ID, but the EUID is root because passwd SUID attribute is root.
6. Similar to any other illegal trespassing, an attacker who has compromised a system needs to ensure they cannot be detected nor traced. The type of attack wildly changes the types of cleaning an attacker needs to perform. One common method is to sanitize the system logs. For example, suppose that an attacker compromised the remaining user account of a terminated employee and used the credentials to log into the system. The login will be recorded in the /var/log/auth.log. An analyst that notices a terminated employee account in the user log would be a huge red flag that an unauthorized person gained access to the system. Another important log is the bash\_history log file stored in the user home directory. The bash\_history file stores a record of every single bash command executed by the user. An analyst could easily see what an attacker had done by looking at the log. In order for the attacker to edit the logs, the attacker will require root permissions. Secondly, an attacker must be sure to hide or remove all files that were created. Temporary executables should be removed securely to further reduce chances of recovery and detection. If the attacker needs for a file or executable to stick around, the attacker needs to hide and protect that file. Simple ways to hide a file include placing it in an obscure directory and prepending the name with a dot which signals the operating system to hide the file from normal listing commands. In the event that the file was found, it could be desirable to prevent a file from being removed. By setting the extended attribute immutable, the file is given an extra layer of removal protection. The permissions for hiding and deleting files range from basic user permissions to root permissions depending on system directories.