LESSON NOTES

Intro to Linux

Security

2.2.2 Account Configuration and Management

Lesson Overview:

Students will:

- Understand how accounts are managed in terms of authentication and security as well as how the system configures the profile for each user once they login to the system

Guiding Question: How does a Linux system manage accounts and configure the session environment once a user has been authenticated?

Suggested Grade Levels: 9 - 12

Technology Needed: None

CompTIA Linux+ XK0-005 Objective:

2.2 - Given a scenario, implement identity management

- Default Shell
- Configuration Files
  - /etc/passwd
  - /etc/group
  - /etc/shadow
  - /etc/profile
  - /etc/skel
  - .bash_profile
  - .bashrc
- Account Management
  - passwd
  - chage
  - pam_tally2
  - faillock
  - /etc/login.defs

This content is based upon work supported by the US Department of Homeland Security's Cybersecurity & Infrastructure Security Agency under the Cybersecurity Education Training and Assistance Program (CETAP).

Account Configuration and Management

Authentication and Configuration

Once a user is created in a Linux system, various actions and configurations occur to establish an environment that is tailored to the user's needs. This environment ensures both the security of the user's data by isolating it and the secure storage of other users' data. When a user logs in or a system is booted up, a series of shell commands are run to establish settings, the look of the system, and even if the user has any aliases, or custom commands, how those will function. Typically, a Linux system runs a default shell, called Bash, in which these shell commands would be run. A shell is like having an interpreter between the user and the operating system, reading the inputs from the startup process and keystrokes to determine what needs to be done and how.

Before diving into setting up Bash startup or profile commands, a user needs to be authenticated prior to gaining access to a system. In most cases, a user will be verified via credentials aka a username and password. The passwd utility is used to set a password for the user. Using the command **`passwd`** alone will give you the option to change the password for your own account, while sudo rights will allow the changing of other accounts in the system. So, for example, if we go back to user John Smith from the previous lesson, we can assign John's account a password or a new password using the command **`sudo passwd -u john`**. Once we press Enter, the system would then ask us to enter the password for John. The passwd utility offers several options such as deleting passwords, setting expiration constraints, locking accounts, and setting the warning days for when account passwords would expire or when the account would be locked.

The chage command offers a more readable view of a user's password information such as when it was changed or if/when it will expire. So, if we wanted to see password information on John Smith's account, we could use the command **`sudo chage john`** to get that readable view of John's password information. Modifications are allowed using the chage command as well as using various options. Most Linux systems will have an /etc/login.defs file that is associated with an account the moment it is created. This file will establish directives for passwords such as character requirements, expiration dates, etc.

While managing user accounts, pam_tally2 and faillock utilities both provide useful tools for viewing and managing failed login attempts. The pam_tally2 tool allows more options than just viewing and resetting login attempts. Those options include disabling the standard passwd and usermod utilities.

When it comes to storing user information, Linux offers a variety of ways for authentication data and associated files and information to be accessed and modified which will vary on the task at hand. For example, creating new users requires less access than running a password audit. The /etc/passwd file is where account information is stored for each user as well as the system accounts which are often created by applications to access the files and directories needed to function. The /etc/passwd file will contain information on the username, user ID, group ID, potentially the full name of the user, the home directory, and the default shell used.

For system accounts, the default shell will typically say nologin or false because those are acting in the background of the user who is using the application. An application would not need to actively login.

This also prevents a malicious user from trying to login to the system under an applications user account. If they were able to even get to the authentication step or tried changing users while in a system, the nologin or false settings in the /etc/passwd file would take effect and either give them an error and not login or boot them from the system entirely. In the past, the /etc/passwd file also contained password information. That has since been changed and now the file only contains an x if a password is set for the user. The /etc/group file contains data similar to the /etc/passwd file but it is tied to the groups on the system. Groups classified as a system group will have similar rules in place to prevent users from logging in as if they were an application. If a password is used for the group it will be noted by an x just as it is in the /etc/passwd file as well.

Since the passwd and group files only denote whether a password is being used or not, there still needs to be a place where Linux can securely store that password data. The /etc/shadow file, or /etc/gshadow when looking at groups, is where credentials are securely stored. The /etc/shadow file will contain information on the usernames and related password information. The password itself will not appear in plaintext, but instead shows in a salted and hashed format. The hashing algorithm used varies between Linux system and the version used, but in any case, it ensures the stored credentials are secure and not easily read or copied. If a "!", "!!", or "*" is present instead of the hashed password, it means the user or group does not have a password set or cannot use a password to login.

In addition to setting up user credentials during account creation, Linux systems also initialize a range of configurations and directories for each new account. The /etc/profile file sets the global environment variables and paths that are common to all users' login sessions. When a user logs in, the /etc/profile file is read and executed to configure the initial environment variables, such as defining default system paths, setting variables for the shell prompt, and configuring system-wide aliases or functions.

The etc/skel directory, or skeleton directory is often used in systems to copy a set of directories and files to the account when created. When a new user account is created, the content of /etc/skel is automatically copied to the user's home directory, providing them with a pre-configured environment. The purpose of this is to ensure that each new user starts with a standard set of files and configuration settings, such as default shell configuration files, startup scripts, and predefined directory structure. This is typically where the bash files would be located.

The .bash_profile file is executed upon login. The primary purpose of the .bash_profile file is to customize the behavior and environment of the Bash shell for a specific user. Users can use this file to set environment variables, define aliases and functions, modify the prompt, and execute various commands or scripts upon login. It is often used to initialize the user's shell session with specific settings and configurations customized to the needs of the user. The .bashrc file has the same purpose but is applied immediately and is not a part of the login process, meaning relaunching the terminal would show the changes.