In his name

Sharif University of Technology

Computer Engineering Dep.

Data & Network Security

HW 1

Mehrad Milanloo

99105775

5. Part 5

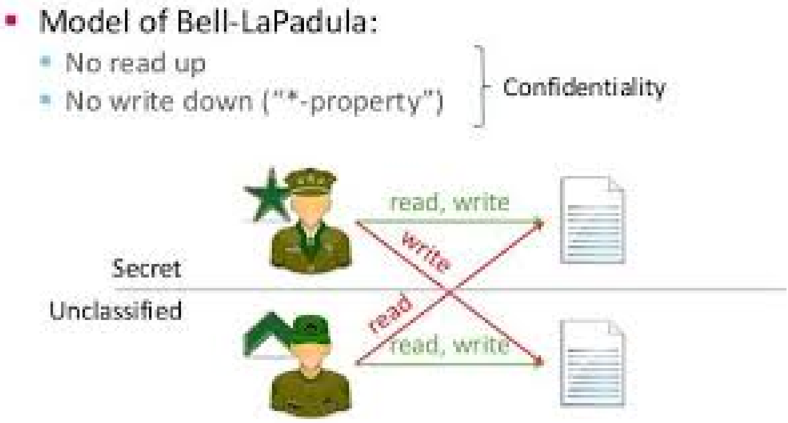
5.1 Theory Questions

5.1.1 Access Control

---

1. The Bell-LaPadula (BLP) model is designed to maintain data confidentiality by imposing two primary access control rules:

- **No Read Up**: A subject at a lower security level cannot read data at a higher security level (e.g., a user with 'unclassified' clearance cannot read 'secret' files).
- **No Write Down**: A subject at a higher security level cannot write data to a lower security level (e.g., a user with 'secret' clearance cannot write to 'unclassified' files).



- Model of Bell-LaPadula:
  - No read up
  - No write down ("*-property")

Confidentiality

Figure 1: BLP model

In Unix-like systems, the permissions are presented in the format `rwxrwxrwx`, where the first `rwx` is for the *user (owner)*, the second `rwx` is for the *group*, and the third `rwx` is for *others*.

`rwx`	`rwx`	`rwx`
user	group	other

Considering the `BLP model` and that the `secret` group is higher than the `unclassified` group, the permissions for the `secret` and `unclassified` files would be as follows:

permissions	owner	group	file name
`rw-rw--w-`	root	secret	secret_file
`rw-rw-r--`	root	unclassified	unclassified_file

2.

- The Linux system uses the `/etc/shadow` file to store encrypted user password information. The permissions for the `/etc/shadow` file should be set so that only the root user has read and write access.
- Users can change their passwords using the `/usr/bin/passwd` command. The `passwd` program needs to interact with the `/etc/shadow` file to update passwords. To do this securely, it is equipped with the `setuid` bit. The `setuid (set user ID)` bit is a special permission that allows users to run an executable with the file **owner**'s privileges, which in this case, would be the `root` user's privileges.
- The `passwd` program should have permissions such as `rwsr-xr-x`. The `s` in the user's execute permission place (`rws`) indicates that the `setuid` bit is set. This means when any user runs the `passwd` command, it operates with `root` level permissions, allowing it to modify the `/etc/shadow` file. When a user wants to change their password using the `passwd` command, a process is created with the `user`'s privileges. But because of the `setuid` bit, this process executes with `root` privileges, allowing it to write to the `/etc/shadow` file.
- The user invokes this process by typing `passwd` in the command line, and if they have `sudo` privileges and need to change another `user`'s password, they would precede it with `sudo` as in `sudo passwd username`.
- The `setuid` bit contains a security risk. If there is a vulnerability in a program with the `setuid` bit set, such as the `passwd` program, it could potentially be exploited to gain elevated privileges, in this case, `root` access.

3. Here's some methods that `SELinux` can help address the vulnerability:

- **Restrict `passwd` Execution Context**: `SELinux` can restrict the execution of the `passwd` command to certain contexts. For example, we could create a policy where the `passwd` command can only be executed by users in a specific role or with specific types, which would require `sudo` to run the command and deactivates the usual command execution without using `sudo` privileges.
- **Limit `passwd` Capabilities**: `SELinux` can control which capabilities a process gets. Even if a program has the `setuid` bit set, `SELinux` can override this and prevent the process from getting full `root` capabilities.

- **Define Strict Domains**: By defining strict domains, `SELinux` restricts the files and resources that a process can access, even if it is running with `root` privileges. For example, if the `passwd` command is compromised, it wouldn't necessarily have access to other critical system files.
- **Role Transition**: We can use `SELinux` to force a role transition when a user attempts to execute the `passwd` command. The user would have to have the proper role that is allowed to transition to the `passwd_t` type, and we could enforce that only users in a `sudo` or `admin` role can make this transition.

Resources

- How to Set File Permissions in Linux - Geeks for Geeks
- Bell-LaPadula Model - Ilahia College of Engineering and Technology
- Using SELinux - RedHat
- Privilege escalation using setuid - Creekorful
- Linux Privilege Escalation - Bordergate
- ...