Felipe Dessoy Caraballo 2024

The Black-List

This project involves developing a C/C++ program to monitor and respond to Bash commands in real-time based on user privilege levels. Below is an outline that breaks down each component of the project, along with a sample blacklist of commands that can download or access webpages without requiring `sudo`.

Project Outline: Real-Time Command Tracking and Response Program

Objective:

Develop a C/C++ daemon that:

- Tracks Bash commands in real-time.
- Performs specific actions based on user-defined privilege levels (1-4).
- Logs or blocks specific commands, defined in a blacklist, based on privilege level.

Requirements:

1. **Privilege-Based Control (1-4)**:
    - The program takes a single integer argument (1-4) on startup to determine privilege level.
    - Based on the privilege level, it will log commands, shut down upon certain commands, or ignore commands as specified.
2. **Privilege Levels**:
    - **Level 1**: Logs all commands and shuts down the computer when any command is executed (using the `SIGQUIT` signal).
    - **Level 2**: Logs commands, uses a flagging system for some commands, and shuts down if a blacklisted command is executed.
    - **Level 3**: Only logs commands without taking any actions.
    - **Level 4**: Does not log any commands but allows for modification of tracked commands (modification mode).
3. **Password-Protected Privilege Escalation**:
    - The program can escalate privileges if the correct password (`"1234"`) is provided.
    - It should also support de-escalation without a password.
4. **Daemon Operation**:
    - The program must run in the background and avoid creating log files in Level 3.
5. **Bonus Requirements**:
    - Ability to change privilege levels during runtime.
    - The program should allow actual system shutdown on Level 1 or 2 when a blacklisted command is executed, beyond using `SIGQUIT`.

Implementation

1. **Privilege Level Functionality**:
    - Implement a logging system for Levels 1, 2, and 3 that avoids creating files at Level 4.
    - Blacklist handling for Levels 1 and 2, including immediate shutdown upon encountering a blacklisted command at Level 1.
    - Flag system in Level 2, which might involve setting flags for dangerous commands without immediate shutdown (e.g., alerting user or logging with a warning).
2. **Privilege Escalation and Modification**:
    - Implement a simple password check for escalation (with `1234` as the default).
    - De-escalation should be possible without a password.
    - Implement privilege level switching and adjust functionality on the fly (e.g., a signal handler could listen for a custom signal to trigger a mode change).

Pointers

1. **Main Program Flow**:
    - Initialise the program based on the privilege level.
    - Daemonize the process to run in the background.
    - Implement command tracking with appropriate responses based on the privilege level.
    - Include password-based privilege escalation logic.
2. **Privilege Level Functions**:
    - `log_command()`: Logs commands unless the level is 4.
    - `shutdown_system()`: Handles shutdown on blacklisted commands for Level 1 and 2.
    - `flag_command()`: Flags specific commands under Level 2, potentially prompting user action.
    - `modify_blacklist()`: Allows blacklist modification in Level 4.
3. **Signal Handling**:
    - Use signals for handling shutdown (`SIGQUIT`) and possible dynamic changes in privilege levels.

Blacklist of Commands (Non-`sudo` Commands for Downloading/Accessing Webpages)

These commands can download or access webpages without requiring `sudo` privileges, therefore bad:

1. `curl`
2. `wget`
3. `lynx`
4. `links`
5. `nc` (netcat)
6. `telnet`
7. `ftp`
8. `scp` (remote copying)
9. `rsync`
10. `git clone`

These commands can access the network, download files, or transfer data, making them potential security risks in restricted environments.