COMP 4108 Assignment 2

Part A)

Find file PartAOutput.bash in the zip file for all the outputs as needed.

9.

   a. ISOLATED-COMPARTMENTS:

      i. This principle can help contain and prevent the spread of rootkits. By separating different system components and processes into isolated compartments (e.g., using virtualization, containers, or sandboxing), the potential impact of a rootkit can be limited. If a rootkit manages to infect one compartment, it would be much harder for it to break out and affect other parts of the system. This isolation can also make detection easier, as anomalies within a compartment might be more noticeable.

   i. LEAST-PRIVILEGE

      i. This principle can help mitigate rootkits by limiting the potential damage they can cause. If all processes and users operate with the minimum privileges necessary for their tasks, a rootkit that compromises one part of the system will have limited ability to spread or escalate its privileges. This makes it harder for rootkits to gain full system control. For example, if a rootkit exploits a vulnerability in a user-level application, it would be confined to that user's permissions, making it more difficult to modify kernel-level components or access sensitive system areas.

Part B)

Find files rootkit.c and insert.sh for the code implementations and documentation/comments

Also find PartBOutput.bash for the output of the code runs.

- **Using the correct __NR_x define:**
  For the x86-64 architecture, the correct define for execve is __NR_execve. This is typically defined in the kernel headers and is used to find the offset in the sys_call_table.
- **Accessing arguments:**
  As hinted, I used regs→di to access the filename argument.
- **Using current_* macros:**
  I used current_euid().val to get the effective UID, which aligns with the hint about using current_* macros from cred.h.
- **Modifying the hook for privilege escalation:**
  The hook checks if the effective UID of the user executing an executable is equal to the root_uid parameter. If it matches, it elevates the privileges to root (UID 0) using prepare_creds() and commit_creds() functions, as suggested in the hint.
- **Setting root_uid parameter:**
  The root_uid parameter is designed to be set to the user's UID (not root's UID) via the insmod command in insert.sh. This allows a specific non-root user to gain root privileges.

Part C)

Find files rootkit.c and insert.sh for the code implementations and documentation/comments

Also find the results of the code running in PartCOutput.bash

- Modifying the hook for file cloaking:
  - The hook modifies the struct linux_dirent buffer returned to the calling process.

  - It filters out any dirents (directory entries) for filenames that start with the magic_prefix.
  - This hides files and directories with names beginning with the specified prefix.
- Implementing the magic_prefix parameter:
  - I added a module parameter magic_prefix to allow setting the prefix for hidden files.
  - This parameter is provided when inserting the module using the insmod command in insert.sh.
- Buffer manipulation:
  - As suggested in the hints, I allocated a kernel buffer to copy and modify the dirent structure.
  - We iterate through the buffer, removing entries that match our hiding criteria.
  - The modified buffer is then copied back to user space.
- Testing the implementation:
  - I followed the steps outlined in the assignment:
    a. Edited insert.sh to set the magic_prefix parameter.
    b. Compiled the module.
    c. Created a test file with the magic prefix.
    d. Inserted the module as root.
    e. Ran ls -l to verify that the file with the magic prefix is no longer visible.
- Handling special characters:
  - As noted in the hint, when using '$' in the magic_prefix, we escape it in the bash shell using '$' to prevent variable expansion.