

Assignment 06

RISHI KUMAR SONI

1001774020

Task 1: Choosing Our Target

Observation

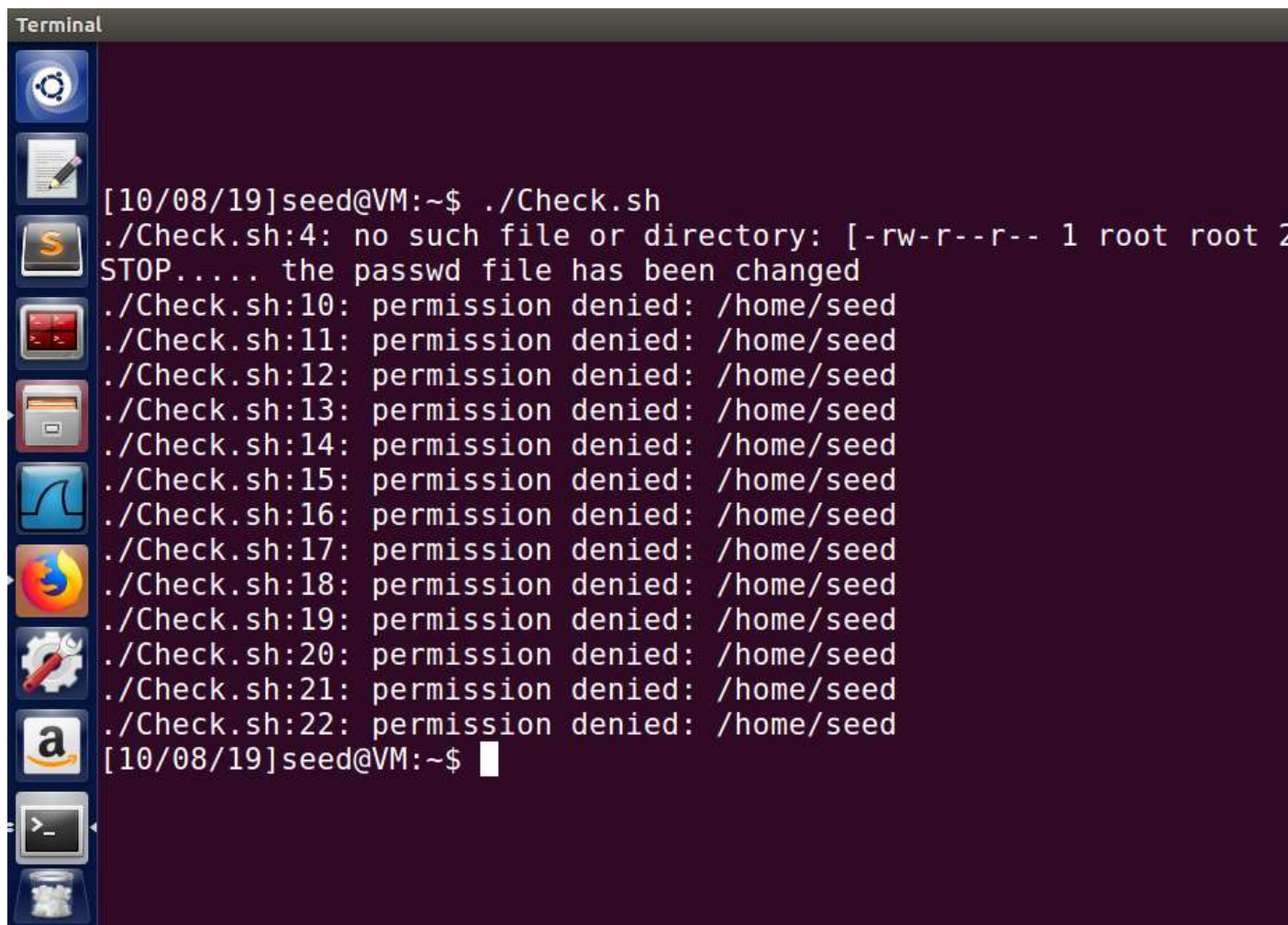
```
root@VM: /home/seed
root@VM:/home/seed# useradd -d /usr/Test -m Test
root@VM:/home/seed# cat /etc/passwd | grep Test
Test:x:1003:1003::/usr/Test:
root@VM:/home/seed# cat /etc/shadow | grep Test
Test:!:18181:0:99999:7:::
root@VM:/home/seed#
```

Explanation:

The third column in the file `/etc/passwd` denotes the UID of the user. Because Test account is a regular user account. If we change this entry to 0, Test will become root.

Task 2: Launching the Race Condition Attack

Observation



```
Terminal
[10/08/19]seed@VM:~$ ./Check.sh
./Check.sh:4: no such file or directory: [-rw-r--r-- 1 root root 2
STOP..... the passwd file has been changed
./Check.sh:10: permission denied: /home/seed
./Check.sh:11: permission denied: /home/seed
./Check.sh:12: permission denied: /home/seed
./Check.sh:13: permission denied: /home/seed
./Check.sh:14: permission denied: /home/seed
./Check.sh:15: permission denied: /home/seed
./Check.sh:16: permission denied: /home/seed
./Check.sh:17: permission denied: /home/seed
./Check.sh:18: permission denied: /home/seed
./Check.sh:19: permission denied: /home/seed
./Check.sh:20: permission denied: /home/seed
./Check.sh:21: permission denied: /home/seed
./Check.sh:22: permission denied: /home/seed
[10/08/19]seed@VM:~$
```

Explanation:

For this task, we disable the sticky protection mechanism. We have to show the Race Condition , it can be done by adding new line in the passwd file.Repeat.sh is the shell code that takes the value from the input.txt and give it input vulnerable program vulp.c.

We can see after multiple attempts at exploiting the race condition, our attack runs and the message is displayed.

Output

We can see after the race condition , our attack runs and message is displayed by check.sh.

Task 3: Countermeasure: Applying the Principle of Least Privilege

Observation

root@VM: /home/seed

```
root@VM:/home/seed# cat vulp.c
#include <stdio.h>
#include <unistd.h>

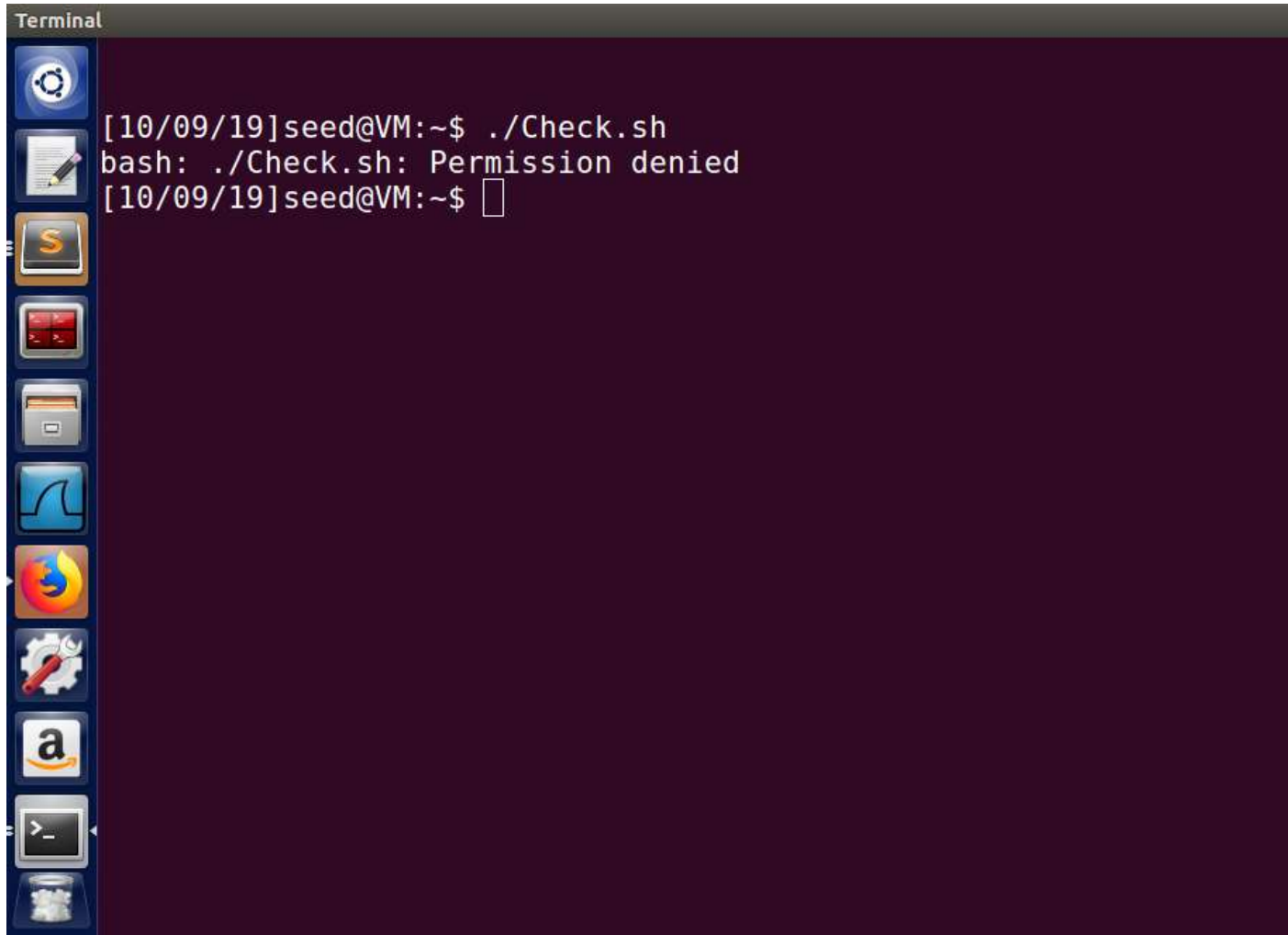
int main()
{
    char * fn = "/tmp/XYZ";
    char buffer[60];
    FILE *fp;

    /* get user input */

    scanf("%50s", buffer );

    uid_t euid = geteuid();
    uid_t uid = getuid();
    setuid(uid);
    if(!access(fn, W_OK)){

        {
            printf("No permission \n");
        }
    }
}
```

A terminal window titled "Terminal" with a dark purple background. On the left is a vertical dock with icons for various applications including a gear, a notepad, a terminal, a file manager, a web browser, and others. The terminal text shows a user named 'seed' at a VM prompt running './Check.sh', which results in a 'Permission denied' error from bash. The prompt returns to the user's shell.

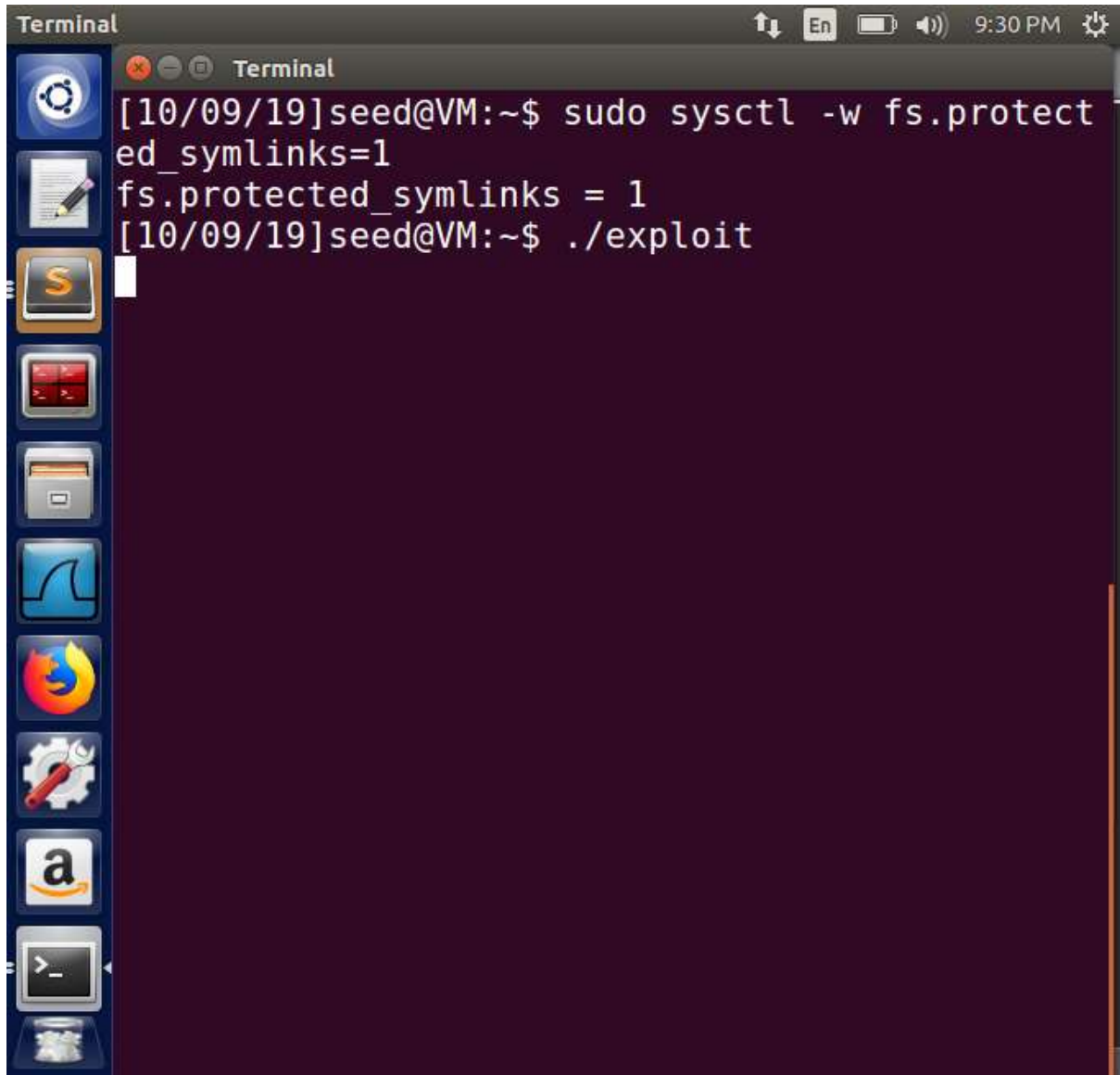
```
Terminal
[10/09/19]seed@VM:~$ ./Check.sh
bash: ./Check.sh: Permission denied
[10/09/19]seed@VM:~$
```

Explanation:

We modified the vulnerable program so that we downgrade the privileges before the checks and then revise the privileges at the end of the program. But after perform the task again, it doesn't work.

Task 4: Countermeasure: Using Ubuntu's Built-in Scheme

Observation

A terminal window titled "Terminal" with a dark purple background. The window has a title bar with standard Linux window controls and system status icons (network, keyboard, battery, volume, time 9:30 PM, and settings). On the left side, there is a vertical dock with various application icons: a gear, a document with a pencil, a yellow square with a dollar sign, a red square with white text, a folder, a blue square with a white line graph, the Firefox logo, a gear with a wrench, the Amazon logo, and a terminal icon. The terminal text shows a user named 'seed' at a machine named 'VM' in the home directory. They run 'sudo sysctl -w fs.protected_symlinks=1', which outputs 'fs.protected_symlinks = 1'. Then they run './exploit', and a white cursor is visible on the next line.

```
Terminal
[10/09/19]seed@VM:~$ sudo sysctl -w fs.protected_symlinks=1
fs.protected_symlinks = 1
[10/09/19]seed@VM:~$ ./exploit
```

Explanation:

In this , we turn on the sticky bit protection and perform the same the attack.Our attack is failed.

1. How does this protection scheme work?

A **Sticky bit** is a permission **bit** that is set on a file or a directory that lets only the owner/root of the file/directory or the root user to delete or rename the file.

2. What are the limitations of this scheme?

Limitations:

- Works only for the directories where sticky bit is on or working.
- It works in some of the cases only. Race condition would work in the directory or file which is owned by the root and which can be modified.