CS 370, Assignment 3 Report

Cody Malick
malickc@oregonstate.edu

December 2, 2016

Question 1

Steps taken:
1. used chmod u-s /usr/bin/passwd
2. Tested passwd again to see what permissions were needed
Error received: passwd: Authentication token manipulation error 3. Researched what capabilities passwd needs under the posix capabilities system
4. Capbilities needed: cap_chown, cap_dac_override, cap_fowner, all needed to be set to ep

From the man pages, the following capabilities grant the following permissions:

cap_chown: Make arbitrary changes to file UIDs and GIDs

cap_dac_override: Bypass file read, write, and execute permission checks.

cap_fowner: Bypass permission checks on operations that normally require the filesystem UID of the process to match the UID of the file. Set extended file attributes on arbitrary files. Set Access Control Lists on arbitrary files.

Question 2

cap_dac_read_search: Bypass file read permissions checks and directory read and execute permission checks.

We can demonstrate this capability by removing the 'ls' command's set-UID capability using:

Listing 1: Removing set-UID capability on ls

```
root@ubuntu: chmod u-s /bin/ls
```

Then, we can test out the lack of set-UID capability by running ls on a directory that requires root access, such as /root:

```
seed@ubuntu:~/ ls /root
ls: cannot open directory '/root': Permission denied
```

Now, we can add the cap_dac_read_search capability to ls:

```
seed@ubuntu:~/ sudo setcap cap_dac_read_search=ep /bin/ls
seed@ubuntu:~/ ls -a /root
.   ..   .bash_history   .bashrc   .bzr.log   .cache   .profile
```

We've demonstrated that, using the cap_dac_read_search capability, we can remove the need for ls to run at an elevated level.

Question 3

After the capability was added, tests b,d, and e failed. Let's break down what happened after the capability was added.

a

After: Passed
Observations: This test just tries to open the file initially. It passes.

b

After: Failed
Observations: This test should have failed, as we call cap_disable(), which allows the process to temporarily disable the privilege.

c

After: Passed
Observations: This test passes, as we have re-enabled the capability using cap_enable().

d

After: Failed
Observations: This test fails because we drop the cabability completely using cap_drop(), making it unrecoverable.

e

After: Failed
Observations: This test tries to reenable the capability after it was dropped. The test fails because, once the capability has been deleted from the process, it cannot be recovered soley by the process.

Question 4

In ACL, we would have to assign elevated privileges to a certain person or role in order to accomplish what we did here. Linux uses the 'sudo' command to make this relatively simple. While typing 'sudo' is fairly easy and convenient, it does not allow very fine grain control of the system. The more sudoers on a given system, the more security vulnerabilities there are.

Capabilites, while less conveniant, are more fine grained in their control of the system. Using capabilities properly would reduce the number of sudoers or super users needed on the system.

Question 5

The attacker would be able to use capability A if he managed to call the code to reenable it. Disabling the capability is only temporary. If the capability was completely dropped or deleted, however, he would not be able to unless he was able to compromise the code before the capability was dropped.

Question 6

I chose very limited options for apache in my apparmor profile. The primary scenario I tackled was apache, serving static webpages, needing very limited capabilities in general. I've added comments to the file as needed. While I've found other options online, just tackling a basic scenario for this exercise made sense.

```
# This is a basic apparmor profile for apache.
/usr/sbin/apache2 {

    # Capabilities
    # It should be able to stop itself
    capability kill,

    # It should be able to bind it's service to a network socket
    capability net_bind_service,

    # www folder should be accessible to serve static webpages
    /var/www/*

    # Apache specific apparmor files from the apache2 apparmor package
    #include<apache2.d>

}
```