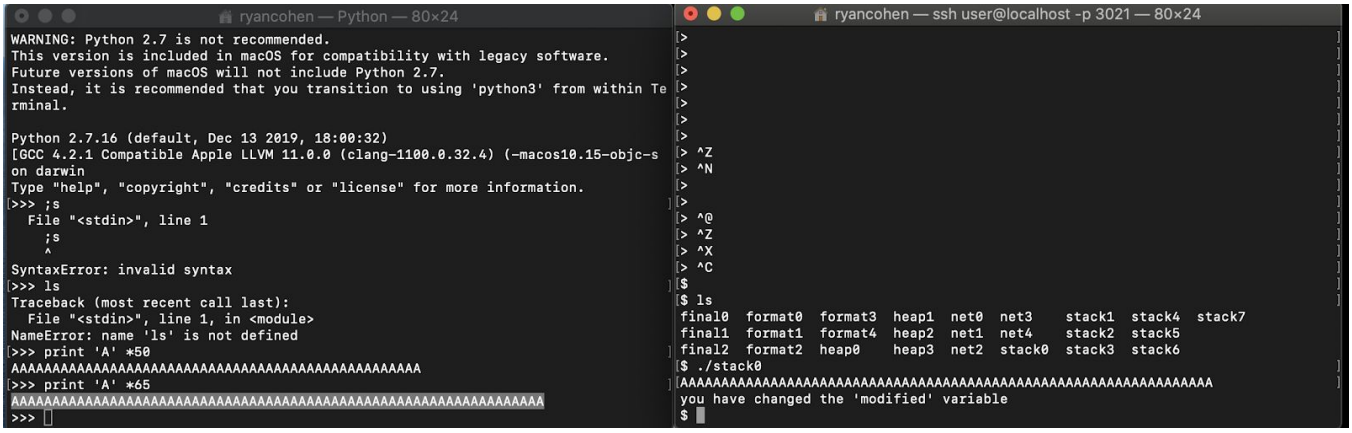
Ryan Cohen

Lab Writeup

1. Source Code:

```
1   #include <stdlib.h>
2   #include <unistd.h>
3   #include <stdio.h>
4
5   int main(int argc, char **argv)
6   {
7     volatile int modified;
8     char buffer[64];
9
10    modified = 0;
11    gets(buffer);
12
13    if(modified != 0) {
14        printf("you have changed the 'modified' variable\n");
15    } else {
16        printf("Try again?\n");
17    }
18  }
```

First thing I did was run the program. When running the program with or without arguments it will wait for a response. When typing in a response, it would say “Try Again?”. Looking at the code I knew from Line 14 I had to change the modified variable. To do that I had to perform a buffer overflow of over 64 as seen in line 8.



As seen in the screenshot, how I did that was to print a random character such as ‘A’ 64 times and then break the limit by printing ‘A’ one more time. I performed that process in python and then copied it over to the program on the right window, which I ssh’d into. At the end, I got the message “you have changed the ‘modified’ variable” correctly solving the problem.

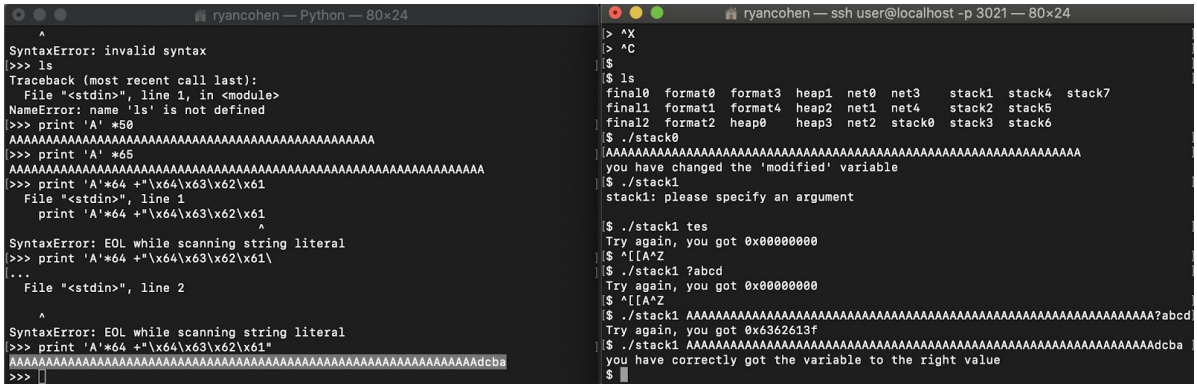
Important commands used:

print ‘A’ *65

2. Source Code:

```
1   #include <stdlib.h>
2   #include <unistd.h>
3   #include <stdio.h>
4   #include <string.h>
5
6   int main(int argc, char **argv)
7   {
8     volatile int modified;
9     char buffer[64];
10
11    if(argc == 1) {
12        errx(1, "please specify an argument\n");
13    }
14
15    modified = 0;
16    strcpy(buffer, argv[1]);
17
18    if(modified == 0x61626364) {
19        printf("you have correctly got the variable to the right value\n");
20    } else {
21        printf("Try again, you got 0x%08x\n", modified);
22    }
23  }
```

When I ran stack1 for the first time, it immediately asked for me to specify an argument. Running the program with an argument of “test” the program told me to try again, and it outputted 0x00000000. Seeing that made me remember the code I just looked at. On line 18, I saw that it had to be 0x61626364. I now knew that I had to overflow the buffer and include that bit at the end.



I accomplished that by printing A 64 times which filled the buffer. I then added the hex code in reverse as Protostar was in little endian. I put that as an argument for stack1 and I got the output of “you have correctly got the variable to the right value”.

Important Commands Used:

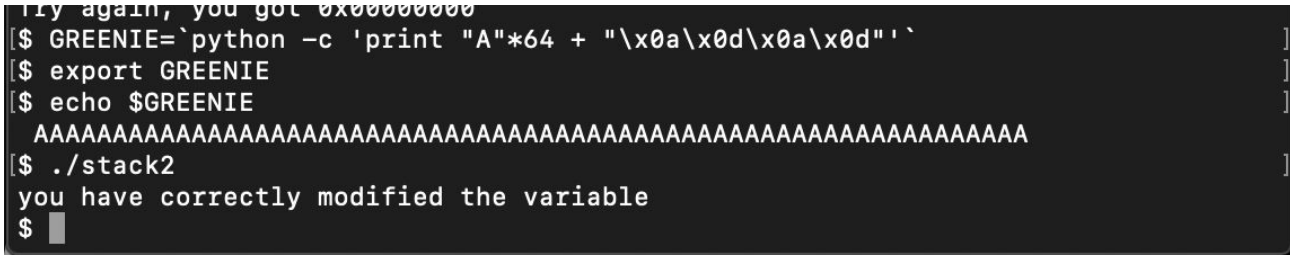
print ‘A’ *64 + “\x64\x63\x62\x61”

3. Source Code:

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

int main(int argc, char **argv)
{
  volatile int modified;
  char buffer[64];
  char *variable;

  variable = getenv("GREENIE");

  if(variable == NULL) {
      errx(1, "please set the GREENIE environment variable\n");
  }

  modified = 0;

  strcpy(buffer, variable);

  if(modified == 0x0d0a0d0a) {
      printf("you have correctly modified the variable\n");
  } else {
      printf("Try again, you got 0x%08x\n", modified);
  }
```

For this program when I first ran it, I got the error “please set the GREENIE environmental variable”. After researching how to do that, I set GREENIE to something random and then ran the program again. It then gave me the message “Try again, you got 0x00000000, modified. Looking at the code of the program it

looked very similar to the rest of the problems where the variable modified had to be changed. Taking a similar approach to the last problem I took python and set the GREENIE variable to overflow the buffer and then plus the 0x0d0a0d0a at the line so then modified would be the right variable.



With that, I got the correct message to appear “you have correctly modified the variable”.

Important Commands Used:

GREENIE= `python -c ‘print “A”*64 + ‘\x0a\x0d\x0a\x0d”’`

export GREENIE

echo $GREENIE