ECE 422 / CS 461, Midterm Exam

Monday, March 6th, 2017

Name:_____

NetID:_____

- Be sure that your exam booklet has 15 pages.
- Absolutely no interaction between students is allowed.
- Show all of your work.
- Write all answers in the space provided.
- Closed book, closed notes.
- No electronic devices allowed.
- You have **TWO HOURS** to complete this exam.
- For every wrong answer, you will lose up to 1 point.
- The lowest points you can get for a question is 0.

Page	Points	Score
2	16	
3	16	
4	9	
5	6	
6	8	
8	6	
9	7	
11	11	
12	8	
13	5	
14	8	
15	10	
Total:	110	

Question 1: Multiple Choice ..... *32 points*

For each question, circle *all* that apply. Some questions may have more than one correct answer.

(a) (2 points) Which of the following is/are NOT a control flow hijack defense?

A. Canaries
B. Data Execution Prevention
C. Return Oriented Programming
D. Address Space Layout Randomization

(b) (2 points) Which of the following is/are NOT one of the virus phases?

A. Propagation Phase
B. Triggering Phase
C. Injection Phase
D. Immune Phase
E. Action Phase

(c) (2 points) The Belmont Report for ethical principles and guidelines for research identifies all of the following as core principles EXCEPT

A. Respect for Persons
B. Distributive Justice
C. Truthfulness
D. Beneficience

(d) (2 points) Which of the following describes the vulnerability that Shellshock took advantage of?

A. Environment variables can be inherited from other parties and will be executed by the Bash parser
B. Bash could update permissions on system files without sudo access
C. Bash could skip arbitrary instructions in system binaries
D. A vulnerability in OpenSSL allowed malicious users to steal private keys

(e) (2 points) Which of the following is a quality shared by ALL gadgets?

A. Ends in a RET instruction
B. Increments or decrements register values
C. Short, typically less than 5 instructions
D. Located in statically-linked library

(f) (2 points) Which of the following is an effective method for preventing insider attacks?

A. Using code walkthrough
B. Monitoring employee behavior
C. Restrict software installation privileges
D. Code Randomization

(g) (2 points) Which type(s) of malware require human assistance in order to replicate?

A. Worm
B. Virus
C. Trojan Horse
D. Bot

(h) (2 points) Bob reverse engineered a copyrighted software for personal gain. Which of the following best describes the situation?

A. Bob might have violated the Computer Fraud and Abuse Act (CFAA)
B. Bob might have violated an End User License Agreement (EULA)
C. Bob might have violated the Electronic Communications Privacy Act (ECPA)
D. Bob is fine since he is cool

(i) (2 points) Buffer overflow attacks do NOT work on the heap because memory is allocated at run time.
   A. True
   B. False

(j) (2 points) Which of the following is/are examples of two factor authentication?
   A. password, security question
   B. password, RSA SecurID Fob
   C. smartcard, RSA SecurID Fob
   D. password, fingerprint

(k) (2 points) SQL injections can be prevented with which of the following:
   A. ASLR
   B. Prepared Statements
   C. Same Origin Policy
   D. CGI

(l) (2 points) Which method of password cracking uses precomputed hashes to look up passwords more efficiently?
   A. Dictionary Attack
   B. Brute Force Attack
   C. Rainbow Table Attack
   D. Collision Attack

(m) (2 points) A website allows users to create usernames using any characters they want. It also displays this username on their profile page. What attack is this vulnerable to?
   A. XSS attack
   B. CSRF attack
   C. Collision attack
   D. Brute Force attack

(n) (2 points) Which of the following are laws regarding computer systems?
   A. Computer Fraud and Abuse Act (CFAA)
   B. End User License Agreements (EULA)
   C. IEEE/ACM Code of Ethics
   D. Electronic Communications Privacy Act (ECPA)

(o) (2 points) Which of these are principles you should follow while creating a password?
   A. Passwords should be long
   B. Passwords should be uniformly distributed
   C. Passwords should never be reused
   D. Passwords should be difficult to remember

(p) (2 points) Chroot jails are NOT capable of preventing malicious applications from network access.
   A. True
   B. False

Question 2: Short Answer . . . . . *23 points*

(a) (2 points) Define the Least Privilege access control policy.

(b) (3 points) Define the same-origin policy, and list one type of attack that circumvents it.

(c) (2 points) What's the difference between how arguments are passed between a system call and a user defined function?

(d) (2 points) In ROP, ret instructions are used to chain gadgets together. Are there other instructions that can be used for a similar purpose? Briefly justify your answer.

(e) (2 points) Name two choices for where to place the reference monitor when implementing system call interposing.

(f) (2 points) List the two key parts of Mandatory Access Control.

(g) (2 points) A large university wants to implement a central sign-on service to authenticate affiliated students and faculty. Give one way that this service may help security and one way that this may hurt security.

(h) (3 points) What is a Trojan Horse? List one example of a kind of software that may contain a Trojan Horse.

(i) (3 points) List two defenses against XSS.

(j) (2 points) Explain two differences between signature based approaches and heuristic analysis as malware countermeasures.

Question 3: AppSec MP Question ..... *32 points*

Consider the following function for parts (a), (b), (c), (d), and (e):

```
void vulnerable(char* arg) {
   char buf[2048];
   strncpy(buf, arg, sizeof(buf));
   printf(buf);
}

int main(int argc, char **argv)
{
 vulnerable(argv[1]);
 return 0;
}
```

Make these assumptions:

- The machine behaves just like the VM from MP1.
- All the defenses mentioned in lectures are off
- **buf** is located at 0xbfffeb2c
- When the program is run with a command line argument of "ABCD%p%p%p%p%p%p%p", it prints to the screen the following right after the printf call returns:
  - ABCD0x800(nil)0xbfffeb2c(nil)(nil)0x444342410x70257025

(a) (2 points) Which format specifier of printf is vulnerable. Why?

(b) (2 points) What command line argument should you provide to the program in order to print out the following?

- ABCD0x44434241

(c) (2 points) In the MP, why can't you write all bytes of the return address with ONE printf specifier from part (a)

(d) (2 points) If we give the program a command line argument of "ABCD%p%p%p%n%p%p", what will happen? Explain.

(e) (5 points) If we give the program a command line argument of "ABCD%p%59p%n%p%p%p". What character sequence will be in **buf** after printf returns? Only write down the characters up to and not including the null terminator. Explain.

You are asked to do return-oriented programming. Consider the following function for parts (f), (g), (h), (i), and (j):

```
void foo(char *arg)
{
  ...
  char buf[4];
  strcpy(buf, arg);
}
```

Make these assumptions:

- The machine behaves just like the VM from MP1.
- All the defenses mentioned in lectures are off
- You see the following information when the program arrives to the breakpoint at foo that you set earlier with the command `break foo`:
  - **buf** begins at 0xbffe3658.
  - (gdb) x/2wx $ebp
    0xbffe3660: 0xbffe3750 0x0805cc46

You are given the following gadgets. The first column is the address in hexadecimal representation followed by the instruction at that address:

```
#gadget1
8001750:  xor    %edx,%edx
8001752:  ret

#gadget2
8057360:  add    $6, %eax
8057361:  ret

#gadget3
8497561:  xor    %ecx, %ecx
8497562:  pop    %esi
8497564:  ret

#gadget4
8057cd0:  int    $0x80

#gadget5
8417365:  mov    $ecx, %edx
8417367:  pop    %ecx
8417368:  ret

#gadget6
8617372   pop    %eax
8617373   ret
```

(f) (8 points) Your task is to open up a shell. Assume these are the only gadgets that you can use, and **ebx** has already been set up for you.

Fill in the table below with gadget addresses. Assume stack grows upward, and the first box is where **buf** points to. You may not have to use all of the boxes, but you shouldn't go beyond.

Put down any necessary clarification here.

(g) (3 points) If ASLR is enabled, will your solution still work? Explain.

(h) (3 points) Ben Bitdiddle is trying to complete MP1.1.5 but is unable to launch the shell. Please help him to find the bugs. (Assume address 0xbffffba0 is accessible)

```
mov $0xb,%eax            #0xb: sys_execve number
mov $0xbffffba0,%ebx
lea 4(%ebx),%ecx         #ecx=ebx+8(argv)
xorl %edx,%edx
movl $0x6e69622f,(%ebx) #/bin
movl $0x68732f00,4(%ebx)  #/sh\x00
mov %ebx,(%ecx)          #argv[0]=/bin/sh
mov %edx,4(%ecx)         #argv[1]=NULL
int 0x80                #sys_execve()
```

(i) (3 points) Assume stack canary is enabled for MP1. You decide to use gdb to look at the canary value, and hard code that value in your buffer, will it successfully open up shells? Explain.

(j) (2 points) What is one pro and one con of a callback shell against a normal shell.

Question 4: WebSec MP Question ..... *23 points*

Please refer to the following python code for part (a)
The function `returnLastName` accepts a user id and return the last name of the user with the corresponding user id. The database table that stores this information has a total of 4 columns.

```
import MySQLdb as mdb
...

def returnLastName(id):
  db_rw = connect()
  cur = db_rw.cursor()
  cur.execute("SELECT * FROM inject3_users WHERE id='"+id
            +"'")
  row = cur.fetchone(); # Returns a tuple
  return row[1]
```

(a) (3 points) Write a value for id which will cause the function to return the MySQL version

(b) (2 points) For 2.2.1.3, Bungle took the md5 of the passed in `password` when running an SQL query. You had to find a value for `password` that had an md5 hash which caused a SQL injection. If you wanted to quickly and efficiently find a value for `password` with a hash that caused a SQL injection, you had to look for a specific pattern in the md5 hash. What pattern did you look for?

(c) Imagine that a website has implemented token based protection to protect against CSRF attacks. Assume the website is not vulnerable to XSS attacks. Assume there is a form to change your password. You can only submit this form while you are logged in. The following code is used to generate the CSRF token that will be used to protect this form:

```
import os, random
from hashlib import md5

def generateToken(username):
  m = md5()
  m.update(username)
  token = m.hexdigest()
  token = token[:16]
```

The username that is passed into generateToken is the username of the currently logged in user.

i. (3 points) Where are CSRF tokens stored? Explain the process that is used to protect a form with CSRF tokens.

ii. (3 points) Why is this method of token generation not secure? How could an attacker take advantage of this to perform a CSRF attack against the change password form?

iii. (2 points) How would you change generateToken in order to make it secure against attacks?

(d) Recall for 2.2.3, you had to write a payload function, and then deploy and execute it via the search form on Bungle. For this question, you will have to do something similar. Assume that the CSRF defense level is set to 1, and that XSS defense is set to 2. On Bungle, CSRF defense level 1 turns on token based csrf defense, and XSS defense level two recursively removes the script tag from the users input.

As a reminder, this is the url for Bungle: `http://bungle-cs461.cs.illinois.edu/`

If you are not sure about exact syntax of any Javascript or jQuery function, you may use the function in pseudo-code style.

i. (5 points) You have to write a function `payload` in this section. The function should prevent Bungle login requests from going through. Instead, when someone tries to login as a user, `payload` should stop that request, and instead login the person in as the user `sneaky` with password `password`. As a reminder, the id of the login button is `log-in-btn`, and the id of the csrf token is `csrf_token`

ii. (5 points) Write a function `makeLink` that will generate a link which will deploy and execute a function called `attack`. This function accepts no arguments. You can assume that that the function has been defined above the `makeLink` function. Make sure the generated link targets XSS defense level two. As a reminder, the endpoint on Bungle that is vulnerable to XSS attacks is the `/search` endpoint, which accepts a parameter named `q`.