Name: _____

ECE568S: Midterm Exam
Examiner: C. Gibson

DURATION: 110 Minutes

1. Please use a **pen** to complete all of your answers to the midterm.

2. Do not turn this page over until you have received the signal to start.

3. This exam is closed book and closed notes. One 8.5"x11" double-sided aid sheet containing your own personally-created notes is permitted. Use of calculators, computing and/or communicating devices is not permitted. Work independently.

4. Do not remove any sheets from this test book. Answer all questions in the space provided, unless otherwise indicated. No additional sheets are permitted.

5. The value of each question or part-question is noted next to the question. The total of all marks is shown below.

6. Please write your name and student number in the space below, and write your name on the top of each sheet.

First Name: _____

Last Name: _____

Student Number: _____

Q1. _____ / 20 Q3. _____ / 30

Q2. _____ / 20 Q4. _____ / 30

Total: _____ / 100

Name: _____

1. Short-Answer Questions (20 marks)

Please *briefly* answer the following questions, in the space provided. If you require more space, please indicate that you are continuing your answer on the back of the sheet, and put the question number next to the continuation of your answer.

a) What sequence of `gdb` command would you enter to report information on the previous stack frame? **(4 marks)**

b) Your company is developing software for a new CPU, for which the "no operation" (`NOP`) instruction has a machine code of 0x00. Explain why this would cause a problem with the buffer overflow technique used in Labs 1 and 2. **(2 marks)**

c) Continuing from the previous question... If you wanted to reimplement a buffer overflow attack using the same technique as in Lab #1, what ***three*** key properties would an attacker require from an alternate command to replace the use of `NOP`? (Everything else is the same as the environment you have been using in the labs.) **(**3 x 2 marks = **6 marks)**

d) HMAC offers an improvement over a basic MAC, by preventing at least one type of attack. Briefly explain the nature of the attack on a MAC that an HMAC prevents (*i.e.*, describe the weakness and what that might let an attacker do). You do not need to provide a detailed description of how HMACs are implemented. **(5 marks)**

e) Describe, in point form, the steps required to obtain a X.509 public-key certificate, signed by a Certificate Authority (CA). **(3 marks)**

Name: _____

2. Buffer Overflow (20 marks)

The following programs each run as described, but each has one or more security vulnerabilities that can be exploited using techniques we've discussed in class. All programs compile and run as described.

a) (5 marks) Consider the following program:

```
01 #include <stdio.h>
02
03 #define     PROGRAM_NAME_LEN 4
04 char        program_name[PROGRAM_NAME_LEN];
05
06 void printMessage ( const char * message )
07 {
08       printf("%s: %s\n", program_name, message);
09 }
10
11 int main ( int argc, char * argv[] )
12 {
13       // Do we have the right number of command-line arguments?
14       if ( argc != 2 ) return (-1);
15
16       // Use snprintf() so that we don't overflow "program_name"
17       snprintf ( program_name, PROGRAM_NAME_LEN, argv[0] );
18
19       printMessage ( argv[1] );
20       return (0);
21 }
```

When executed, the program prints up to the first 3 characters of its name, and whatever message is provided in the first command-line argument:

```
$ prog_2a "Hello world"
pro: Hello world
$
```

Briefly describe the security vulnerability (or vulnerabilities) in the above code.

Program Line(s)	What approach would an attacker use to exploit it?

b) **(5 marks)** Your company is developing software on a system in which the process stack grows in the opposite direction from the environment we have been using in the labs (*i.e.*, both the stack and the array addresses grow from smaller to larger addresses). Starting with the provided template, write a short function that could be compromised with a buffer overflow attack.

```
void foo ( const char * cpoint )
{
     char buffer[128];




}
```

c) **(10 marks)** Continuing from the previous question... Assume that `foo` is called from `main`. Complete the following two stack diagrams, showing the relative locations of all local variables (`cpoint`, `buffer`, etc.), and all saved return addresses for every stack frame. You do not need to show any other elements on the stack. In Diagram 2 circle the return address that has been overwritten.

Diagram 1
Show the state of the stack at the very start of the call to `foo`

Stack Frame for _____	
Stack Frame for `foo`	
Stack Frame for _____	

Diagram 2
Show the state after your buffer overflow attack has run (before the return)

Stack Frame for _____	
Stack Frame for `foo`	
Stack Frame for _____	

Name: _____

3. Other Vulnerabilities (30 marks)

For the following two parts, consider the following string used in a Format String attack (the string has been spit onto two lines for readability):

0x4320	12	0x4321	96	0x4322	42	0x4323	*shellcode*

%18x	%hhn	%145x	%hhn	%110x	%hhn	%145x	%hhn	\0	

Assume that the string causes an address of 0xcd3ccd3c to be written in little-endian format:

Address	Value
0x00004323	0xcd
0x00004322	0x3c
0x00004321	0xcd
0x00004320	0x3c

a) **(5 marks)** Modify the Format String that is shown above, so that it still writes an address of 0xcd3ccd3c, but in big-endian format:

							shellcode

e) **(10 marks)** Modify the original format string (from the top of the page) to make it as short as possible (assume the shellcode is already as small as possible):

							shellcode

How many characters has this saved? _____

f) **(15 marks)** You are asked to consider a system that has the Non-Executable flag set on the memory pages occupied by the stack. Your task is to implement an exploit similar to Lab #1, but, in place of the usual shellcode, you are to use the Return-Oriented Programming (ROP) approach we discussed in lecture.

Your buffer overflow has succeeded in placing the following values onto the stack:

Absolute Address	Relative Address	Variable	Value
0x1039304	%EBP+0x54	y	0x0
0x1039300	%EBP+0x50	x	0x1039298
0x1039298	%EBP+0x48	buffer	/bin/sh\0

If we were able to place executable code on the stack, we might wish to execute the following:

```
movl 0x48(%ebp),%ebx     # argv[0]
leal 0x50(%ebp),%ecx     # argv
movl $0x0,%edx           # NULL (env)
movl $0xb,%eax
int  $0x80
```

The existing program code and libraries contain the following four code fragments (*widgets*):

```
...
0x10200  movl $0xf3,eax
0x10202  int $0x80
0x10205  movl %ebp,%eex
0x10207  ret
...
0x11700  call 0x12384
0x11705  subl $0x1,%eax
0x11707  movl 0x48(%eex),%ebx
0x1170a  ret
```

```
...
0x12100  leal 0x50(%eex),%ecx
0x12103  movl 0x60(%ebp),%ebx
0x12106  movl $0x1,%edx
0x12108  ret
...
0x13400  subl $0x1,%edx
0x13402  movl $0xc,%eax
0x13406  ret
...
```

Fill in the list below with the sequence of return address(es) that you should place onto the stack, in order for your exploit to properly run. (You just need to list the address(es): you do not need to show the stack or how to put the values onto it.)

RA1:		RA4:	
RA2:		RA5:	
RA3:		RA6:	

Name: _____

4. Encryption (30 marks)

a) **(5 marks)** Your company is designing the control system for an aerial drone. You are required to use block encryption for sending commands to the drone (*e.g.*, "lower the landing gear", "set engine speed to 50%", etc.). Each command is three blocks in length, and the drone needs to decode any received commands as quickly as possible. What encryption mode would you recommend, and why?

b) **(6 marks)** The aerial drone captures video and saves it to a local solid-state hard drive, encrypting the video with a stream cipher as it is written to the disk. The operators of the drone can request any random segment of the video, at any time (e.g., "send a 15-minute video clip, starting 150 minutes after take-off"). What type of stream cipher would you choose for this application? How would playback work? (You can assume that the remote operator has the same key that was used to encrypt the data.)

c) **(9 marks)** Consider a basic SP network, as discussed in class. Explain the purpose of the S-box, the P-box and the key. Do not explain how they are implemented: explain their purpose for existing in the design.

d) **(10 marks)** Consider a simple exchange based on public-key cryptography: Alice picks a shared secret, **K**, encrypts that value with Bob's public key, and sends the message to Bob; Bob then decrypts the message with his private key and receives the value **K**; Alice and Bob then use **K** as the encryption key for their subsequent conversation. What two important problems exist with this algorithm?