Name: _______________________________________ Student ID:______________

You will need:
Kali (VM)
CYSCA2014InABox (VM)
Windows 95 (VM)
A computer with internet access

COS30015 IT Security

Lab 3 (week 3)

In this lab you will experiment with Buffer overflows in the C language.

1. Using VMware Workstation Pro, start up the COS30015 / **Kali Linux**.

Select Other… if needed
Log in as *user* (user)
COS30015user (password)

Don't need *root*. Now that we know a bit about security models, we're going to implement the principle of least privilege. *user* is a restricted account.

Let's write a C program:

Start the editor thus:

```
pico memtest1a.c
```

Type in the following code:

```
/* memtest1a.c*/
#include <stdio.h>
#define SIZE 8
void test(int*, char*);
int main(){
      int i = 0;
      char buf[SIZE];
      printf("Type in 5-20 chars into the text buffer\n");
      printf("Watch the value of i \n");
      printf("it will be corrupted when you exceed %i chars\n",SIZE);
      printf("Type \"q\" to exit:\n");
      printf("\t| i posn\t| buf start\t| buf end\t| i value\n");
      do {
            test(&i, buf);
            i++;
      }while(buf[0] != 'q');
      return 0;
}
void test(int *j, char* buf){
      scanf("%s",buf);
      printf("OK.\t| %u\t| %u\t| %u\t| %d\n",j, buf, &buf[SIZE],*j);
      return;
}
```

Use
Control+O to write to file (followed by ENTER)
Control+X to exit

Compile thus:

```
gcc -o memtest1a memtest1a.c
```

Name: ____ Student ID:______________

When the function is called, ***i*** is passed by reference. The ***test*** function can access this variable.
buf, being an array, is passed by reference, so the memory location is shared between the main function and ***test***.

and run: `./memtest1a`

Where in memory (the address) is ***i***? What is its value?

	Location (posn)	**Comments**
i		
buffer start		
buffer end		
buffer size		
Bytes between buffer start and i		

Each string entered is appended with a NULL characters (a 0 byte). If you type 8 characters, a 9th character (the NULL) will overflow the array and write over adjacent memory. If this memory is in use, it gets corrupted.

Play with the program (type stuff into it) and record your observations here:

Test No (i)	**Text typed into buf**	**Number of characters**	**Behaviour**
0			
1			
2			
3			
4			
5			
6			
7			
8			

When a buffer fills up, it writes forwards or backwards depending on the CPU and operating system.

Name: ______________________________________ Student ID:______________

Type in a few strings smaller than 10 characters – note that the integer ***i*** is counting correctly.

What is the distance (in bytes) between ***buffer start*** and ***i***?

If you try this in RedHat, the memory locations are further apart – about 13 bytes will overflow the LSB of i

n = (&i - buffer start) =

Try inputting that many characters (***n***) to overwrite the variable ***i***.
Try it. Does ***i*** change?

If it crashes, just start it up again and keep typing longer strings. Watch the value of i.

Try inputting n + 1 characters to overwrite the variable ***i***.
Try it. Does ***i*** change?

If a string is really long, you will crash the program (Linux will report a Segmentation Fault).
Code can be injected into the window between buffer overflow and segmentation fault, overwriting other parts of the program such as the ***return address*** and the ***EBP***

More background here:
http://www.tenouk.com/Bufferoverflowc/Bufferoverflow2a.html

2. Try this program:

On Intel architecture, the last name overflows over the first name. It's different on other CPUs. That's why crackers want to know about the hardware.

```
/* memtest2.c*/
#include <stdio.h>
int main() {
     char first[12];
     char last[12];
     printf("Type in your first name: ");
     gets(first);
     printf("Type in your last name: ");
     gets(last);
     printf("Hello %s %s\n", first, last);
     return 0;
}
```

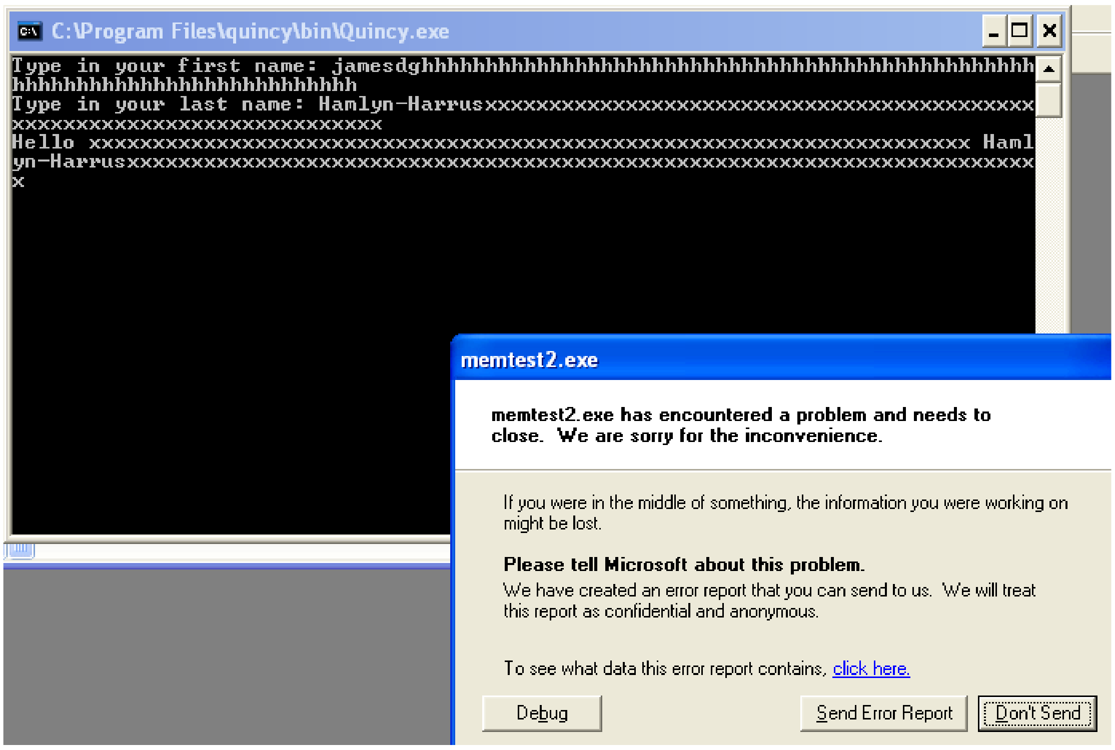
The compiler warning about **gets** will not stop compilation. You can ignore it.

Unlike ***scanf, gets*** lets you type in spaces and other non-printing characters, so is allows an attacker to enter executable code.

If you put in long strings, you can overflow one string with the contents of the other. If they are big enough, you will get a segmentation fault. If the size is just right, you will overwrite a function somewhere else in the program.

How many characters must you input into the last name to overwrite the first name (try it)?

3. You can also try these programs in Windows using the Quincy IDE (in the Win95 VM). If a 'segmentation error' occurs, Windows will pop up this message:



```
Type in your first name: jamesdghhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh
hhhhhhhhhhhhhhhhhhhhhhhhhhh
Type in your last name: Hamlyn-Harrusxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Hello xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Haml
yn-Harrusxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
x
```

memtest2.exe has encountered a problem and needs to close. We are sorry for the inconvenience.

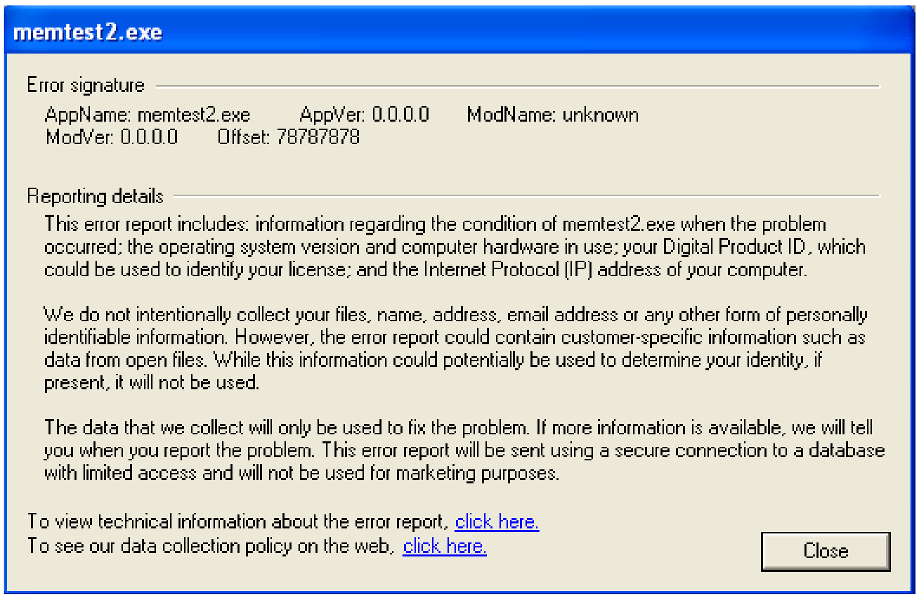
If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.
We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, click here.

If you click on click here, you will see the contents of the stack pointer, which contains the hex value of some of the string you typed in.

Name: ______________________________________ Student ID:______________



```
Error signature
  AppName: memtest2.exe    AppVer: 0.0.0.0    ModName: unknown
  ModVer: 0.0.0.0    Offset: 78787878
```

Reporting details

This error report includes: information regarding the condition of memtest2.exe when the problem occurred; the operating system version and computer hardware in use; your Digital Product ID, which could be used to identify your license; and the Internet Protocol (IP) address of your computer.

We do not intentionally collect your files, name, address, email address or any other form of personally identifiable information. However, the error report could contain customer-specific information such as data from open files. While this information could potentially be used to determine your identity, if present, it will not be used.

The data that we collect will only be used to fix the problem. If more information is available, we will tell you when you report the problem. This error report will be sent using a secure connection to a database with limited access and will not be used for marketing purposes.

To view technical information about the error report, click here.
To see our data collection policy on the web, click here.

Note: 78787878 is hex for xxxx.

Hackers can use this feedback message to figure out which bytes in their exploit string will be copied into the stack pointer. Changing these critical bytes to the location of some (of their) executable code allows the exploit to run their code. With Linux, it's a bit harder, but not much.

4. Re-write ***memtest1.c*** or ***memtest2.c*** to prevent excess characters being pasted into the char arrays:

HINT:

replace ***scanf("%s")*** with ***scanf("%10s")***

replace ***gets*** with ***fgets***
```
usage: char * fgets ( char * str, int num, FILE * stream );
```
eg. ***fgets(first, 12, stdin)***

Unlike gets, fgets leaves characters in the keyboard buffer.
In windows you can clean up the input stream with

```
        fflush(stdin);
```
In Linux you can do the same with
```
        char ch;  /*declare once only*/
        while ((ch = getchar( )) != '\n' && ch != EOF);
```
example code in ***safegets.c***
on Blackboard.

5. Using Google, look up the various "safe" string manipulating functions available in C. (e.g. strncpy, strncat)
What does the ***n*** do?

How is value of ***n*** determined?

6. Start up the COS30015 / **CYSCA2014InABox**
Surf to `http://192.168.100.210`
Click on CHALLENGES
Scroll down to Exploitation
Read the details of "The Fonz".
Click on the c file (let it open in GVim), and examine the code.

Note that two arrays DestBuffer and FixedVariable are adjacent to eachother. If we can overflow DestBuffer we might be able to write into adjacent variables.

On the host PC, open a browser and go to

Don't use the web browser – you can't do interactive input. Netcat (nc) is a universal Telnet-like client suitable for this.

https://github.com/CySCA/CySCA2015/blob/master/corporate_network_pentest/files/chaff/assessments/Light%20Reading/CySCA2014_Exploitation.pdf

Follow the Walk-through for "The Fonz". Open a command window and launch netcat:

```
nc 192.168.100.210 20000
```

Sending a bunch of **A**'s confirms that the overflow works.

What is the reported contents of FixedVariable?

Connect again and send a series of characters (alphabet) – you can "read" the position of FixedVariable by observing which characters get sent back.

Follow the walkthrough.

```
nc 192.168.100.210 20000
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Name: ____ Student ID:_______________

What is the reported contents of FixedVariable?

What ASCII characters do these HEX values represent?

HINT-look them up on an ASCII table at asciitable.com on the host PC.

Follow the instructions.

The server wants 0x73696851, which when converted to ASCII and reversed is?

Try it out:

```
nc 192.168.100.210 20000
ABCDEFGHIJKLMNOP<your 4 chars answer here>
```

What's the key?

7. On the host PC, surf to http://www.vividmachines.com/shellcode/shellcode.html and read up on injecting shellcode into Linux and Windows processes.

What does . *GetProcAddress()* do?

8. Using Google, look for tools and programs for checking a C program for potential buffer overflow vulnerabilities and memory leak testers. What are the names of some?

9. Shut down all VMWare images
and browsers and log off.

End of Lab.