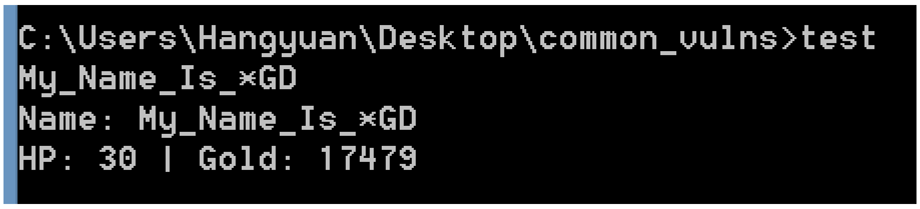
ELEC5616 AS3

Yuming JIANG 460444981
Hangyuan Yu 460469308

1.1 Savegames

(1) In this case, the characters health cannot be set using buffer overflow attack, but the gold ca be changed to a number greater than 9000. The process of buffer overflow attack will be described later. First, I will introduce how the operating systems allocate strings in the memory. In 64-bit operating systems (which we use to execute the program), strings can only be allocated every 4 bytes at one time. In the program "savegame", the length of the name string is 10. As a result, the system will prepare 12 bytes size for the name string to be stored. If the length of the name string is larger than the memory size, the buffer overflow will take place. Once there is buffer overflow, the next declared variable "gold" will be overwrited by the strings that is overflowed. In this case, if we set the name larger than 12 bytes size, for example 15 bytes, it will overflow 4 bytes, because the null separator accounts for one byte and the total number of bytes for the string is 13 bytes in the memory. Using this theory, we are able to change the "gold" number larger than 9000. We use the name "My_Name_Is_*GD". We find that the decimal value of of the overflow characters are shown and is greater than 9000, which is 17479.

The execution result is shown below:



```
C:\Users\Hangyuan\Desktop\common_vulns>test
My_Name_Is_*GD
Name: My_Name_Is_*GD
HP: 30 | Gold: 17479
```

(2) To mitigate the exploit, one easy step is to declare the input value name at last. This will make the exploit much more difficult. However, this cannot prevent the exploit at all. The way to prevent this exploit is to always do bounds checking. In addition, use compilers with certain protection will be a way to prevent this exploit.

(3) The exploit could be useful for more than just the game. When attackers implement the buffer overflow attack, they can let the overflow overwrite the return address and change it to point to the injection code that attackers want to execute. In this way, attackers may be able to get the access of a system. A good example of utilizing this exploit is the crack of Sony PSP 3000. The hackers stores a picture in the PSP, which makes PSP have buffer overflow. Then attackers are able to execute their program and downloaded game without the permission of the machine.

1.2 iCubeKinect

(1) We think it is not possible to use the symmetric cipher. Because it is very difficult for iCubeKinect to give the key to its user securely. And there will also be too difficult to handle a lot of keys if there is a large user base.
With using asymmetric cipher, iCubeKinect can simply hard coded the public key into the DVD-Reading-Machine or just store the public key in the cloud. When user want to verify their DVD games, they can get the public key very easily.

(2) We found that the cert is stored in the DVD. Anyone can get the cert from buying one DVD from iCubeKinect. With this cert, the hacker can store anything in one DVD and put the same cert into that fake DVD. With that correct cert, user can not find that the DVD is not from iCubeKinect.

(3) We think the "content_hash" can not be made static in the cert file stored in the DVD. The DVD-Reading-Machine should generate the "content_hash" with all of the content stored in DVD when user play the DVD with the machine. With this, the hacker can not change the content in the DVD easily even the hacker can get the correct cert from the DVD.
We think the security vulnerability can be less with the using SHA-512. Because, with MD5, the hacker can brute-force get the same MD5 hash result with hacker's own contents as the MD5 result with iCubeKinect's content. If change MD5 to SHA-512, the hacker will pay much more to find the correct hash result.

1.3 General Questions

(1) The reason why GCC provides -fno-stack-protector is that it can have certain stack protection towards the program. The technique "canary" is a mechanism to check and prevent stack overflow attack, which provides security protection while the program is running. It is usually implemented by compilers. It inserts a value which is difficult to counterfeit into a protected stack when the program is running (it is usually a random value in 32 bytes). The value of "canary" will be checked when the function returns the value. If the value is changed, it is considered to be the occurrence of buffer overflow.

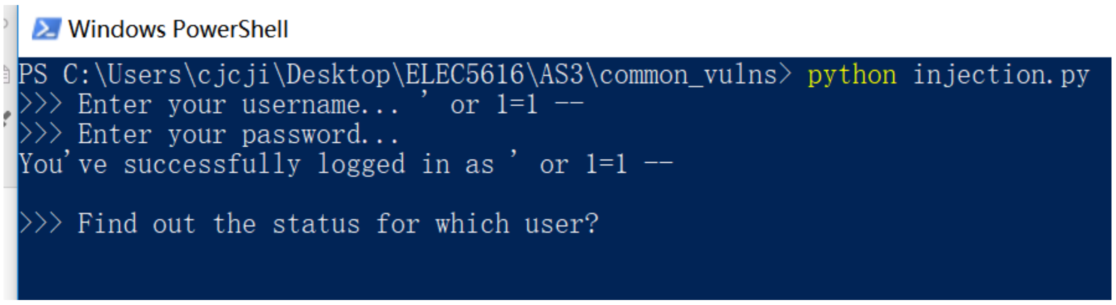
(2) If the game was written in Java, it can in some way decrease the risk of exploits. It is because Java has its own bound checks when assign a value to a variable. When there is buffer overflow attack, Java will pop an exception and automatically stop the program from executing. However, since the implementation environment of Java, JVM, is written in C and assembly language, which has the possibility to have buffer overflow attack.

(3) It is possible to obtain a BASH shell using buffer overflow attack. When

the attackers implements buffer overflow attack, they will prepare a constructed buffer which includes the shellcode inside the the buffer and stored it in a sensitive folder. Then a program that is written by the attacker will find the address which stores the constructed buffer. The address is recorded and will be used to overwrite the return address. And then the shellcode inside the constructed buffer in the address will be executed. The shellcode we use to test the exploits is shown as below ("A complete tutorial on the stack-based buffer overflow programming using C code on Linux opensource OS with real demonstration", 2017) :
"\x31\xc0\x31\xdb\x31\xd2\x53\x68\x55\x6e\x69\x0a\x68\x64\x55"
"\x55\x4d\x68\x41\x68\x6d\x61\x89\xe1\xb2\x0f\xb0\x04\xcd\x80"
"\x31\xc0\x31\xdb\x31\xc9\xb0\x17\xcd\x80\x31\xc0\x50\x68\x6e"
"\x2f\x73\x68\x68\x2f\x2f\x62\x69\x89\xe3\x8d\x54\x24\x08\x50"
"\x53\x8d\x0c\x24\xb0\x0b\xcd\x80\x31\xc0\xb0\x01\xcd\x80";
(source:
http://www.tenouk.com/Bufferoverflowc/bufferoverflowvulexploitdemo32.html)

2. SQL Exploits

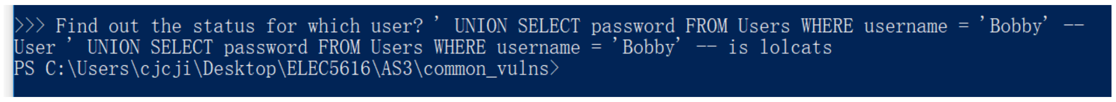
(1) The SQL statement to check the username and the password is: *SELECT 1 FROM Users WHERE username = '%s' AND password = '%s'.* In SQL, all the things after "--" will be seemed as comments. So the hacker can type in the username as "' *or 1=1* -- " and type in anything for the password place. The final SQL statement is: *SELECT 1 FROM Users WHERE username = ' ' or 1=1* -- *' AND password = ' '*. "-- *' AND password = '* '" becomes the comment, and the where check statement "*username = ' ' or 1=1*" is always true. So the hacker can log in without knowing the username and the password.



```
Windows PowerShell
PS C:\Users\cjcji\Desktop\ELEC5616\AS3\common_vulns> python injection.py
>>> Enter your username... ' or 1=1 --
>>> Enter your password...
You've successfully logged in as ' or 1=1 --

>>> Find out the status for which user?
```

(2) The status query is *SELECT status FROM Users WHERE username = '%s'*. The hacker can type in "' UNION SELECT password FROM Users WHERE username = 'Bobby' --". So the statement becomes *SELECT status FROM Users WHERE username = '' UNION SELECT password FROM Users WHERE username = 'Bobby'* -- *'*. The statement after the operator UNION will also be executed, because of this, the password for Bobby is shown.



```
>>> Find out the status for which user? ' UNION SELECT password FROM Users WHERE username = 'Bobby' --
User ' UNION SELECT password FROM Users WHERE username = 'Bobby' -- is lolcats
PS C:\Users\cjcji\Desktop\ELEC5616\AS3\common_vulns>
```

(3) This is not a difficult security problem to fix. It is so common because a lot of people who design the application do not know the knowledge about SQL and Database system.
There are four main ways to prevent this type of attack.

① When design the application, use parameterized statements to write all the query statements.

② Just escape all the characters that have a special meaning in SQL. For example, a single quote in a parameter must be replaced by two single quotes.

③ The developer can fix the input parameter to some strict pattern. If the user's input does not follow the pattern, the query will be denied.

④ The developer can also limit the permissions on the database login by the application to only what is needed. So some core tables can not be accessed by the hacker.

(4) I think SQL injection vulnerability is more severe than buffer over flow exploit. Because the developer who is dealing with the buffer over flow exploit always have the knowledge of c. They can handle the problem by themselves. However, when developers dealing with the SQL injection, the developers may only know javascript and php and have no idea about SQL and database. They should ask other experts in database field to help them solve the SQL injection problem.

References:

A complete tutorial on the stack-based buffer overflow programming using C code on Linux opensource OS with real demonstration. (2017). Tenouk.com. Retrieved 3 June 2017, from
http://www.tenouk.com/Bufferoverflowc/bufferoverflowvulexploitdemo32.html