# Project 3: Common Vulnerabilities

**Steven Tridgell \_\_\_\_\_\_\_\_\_**

**Nicholas Carydis 310183626**

## Savegames

* 1. **Set the characters gold or health to a number greater than 9000 by utilising a buffer overflow. How did you achieve this? Explain using reference to bytes and ASCII as to what the exact value was that you achieved**.

Setting the name to: stephenabcZZZZZZ gives Gold: 1515870810.

* 1. **How could this exploit be prevented?**

By limiting the number of characters read in the input stage to 9 chars + '\0' character.

* 1. **Could this exploit be useful for more than just the game? Could it be used to gain access to a system? If not, why not? If so, where might it be used?**

Yes. This is a typical buffer overflow. It can be used to potentially execute malicious code which can completely compromise the system especially if the game is running with higher permissions that the user currently has.

## iCubeKinext

* 1. **Why does the iCubeKinect system use an asymmetric cipher to verify their DVD games? Would it be possible to use a symmetric cipher instead?**

If a symmetric cipher was used then the key would have to be in all gaming consoles otherwise the game would not be playable. This would essentially give everyone in the world the key defeating the point of using any cryptography or signature. An asymmetric cipher is used so that only the official developers can sign the DVD.

* 1. **What problem exists in the iCubeKinect verification code? How could you make the machine execute any arbitrary DVD?**

The code is only hashing 52 bytes, 'metadata' and 'content\_hash'. Malicious code could be put in the metadata, then using a birthday attack assuming thousands of available certificates the 'content\_hash' is changed as a nonce until the MD5 matches one of the valid MD5's then its RSA signature can be copied, making it valid.

* 1. **How would you fix it? Would the security vulnerability be made less serious by using either a stronger hashing scheme (such as SHA-512) or a different asymmetric cipher?**

No, SHA-512 or a different cipher would not change the security. It can be fixed by hashing the key-id at the front. ie) char \*cert\_hash=MD5(cert.key\_id + cert.metadata + cert.content\_hash); This would prevent a birthday attack assuming that the key\_id is unique.

## General Questions

* 1. **Why is it necessary for us to provide the ﬂag -fno-stack-protector to GCC? What is a canary in terms of a buﬀer overﬂow and how can a canary prevent a buﬀer overﬂow exploit?**

–fno-stack-protector disables the stack protection mechanism included with GCC. (Why? Possibly for compatibility/performance reasons?)

A canary is a value stored before a function return pointer and in some cases control code that is also stored in another piece of memory elsewhere. This existence of this value is checked before execution of the control code, to ensure it has not been altered. If the value has been altered, the control code is not executed as a buffer overflow may have occurred (and the pointer may have been changed to point to alternative malicious code).

The reason a canary works to prevent stack-based overflows is due to the order of memory that is accessed to overwrite the return address (or other pointer) in the stack; as the stack holds more accessible memory near the top, it is likely that an attacker will write to the stack from a lower memory address to a higher memory address. As the canary is at a memory address one lower than the protected values, it is likely it will be affected by any unauthorized memory write operations.

* 1. **If the game above was written in Java instead of C, would the savegame still be exploitable?**

No? Java has stack protection??? Check

* 1. **Imagine you were exploiting a program that was running with escalated privileges (i.e. could read sensitive files, modify other users settings and so on) is it possible to obtain a BASH shell using buffer overflows? Be sure to explain what shellcode is and how the shellcode is executed.**

Call system() or some other C function?

Look here: <http://www.phrack.com/issues.html?issue=49&id=14>

## SQL Exploits

* 1. **Show how it is possible to log in as any user by performing an SQL injection attack on the username/password login page.**

username = ….

password = a' OR 1==1 –

* 1. **The website has been clued in on their major security problem and prevented the previous attack. Is it possible to use the status query to work out the password of one of the administrators Bobby?**

ANSWER HERE

* 1. **How can these attacks be prevented? Is it a difficult security problem to fix? Why is it so common?**

Sanitising all inputs prior to using them in any SQL queries is an easy way of preventing this avenue of attack. (how is this done?) Although this is not a difficult security problem to fix, it is common due to laziness, oversight and/or generally bad programming practices.