Report for: Re-engineering, securing and decommissioning software

Name: Jamie Cropley

P Number: P15188432

Module: IMAT2605 - Object Oriented Design and Development with C++

For attention of: Peter Cooke / Rafael Ktistakis

Date: 09/01/2017

Summary:

This report entails the re-design of the provided software which is a Connect4 game, it is played by means of two players connecting through a server remotely to play each other over a networked connection.

This report also investigates the issues with the software especially that of security issues, then re-designs such issues using an object orientated programming approach in the formation of a UML Diagram design involving redesigned classes containing methods and attributes.

Additionally, from a hypothetical context the report explains the process of decommissioning the existing software and implementing the new software, and explains the problems with this along with proposed solutions.

Page count:

Page count does not include Sections 4 and 7, and this page therefore total pages = 6

Contents:

1 Introduction

This report is needed to outline, plan, analyse and re-structure in a UML diagram the coursework software provided which is a game of Connect 4 which can be played by two players over a formation of a computer network by means of client ← → server ←→ client. Although most of my background knowledge is in C++ the coursework software is written in C therefore rewriting in terms of the UML diagram towards an Object Orientated Programming (OOP) approach is essential.

I am aiming to do this through the objectives outlined in the course spec and the contents of this report, especially in terms of what I can improve in the software with the security elements of it, I believe these are of the most importance to the software becoming better improved because the software is being used over a network. My aims and objectives will therefore mainly revolve around the security aspects of the software and how I go about improving them throughout this report.

I plan to achieve this through planning which would involve me going into some detail of the choices I will make on my Object Orientated design, then go onto the analysis of the existing security issues with the software and how I can improve such issues. Then I would detail my changes in the form of a redesign of the software via a UML based OOP type diagram, where this should also help with the decommissioning and transition into the new software.

2 Planning

My first initial thoughts on the software were that it was split into two projects where one project is the client and one is the server. From briefly overlooking the file structure and compiling such files, it was apparent to me from the onset that most of security issues most likely arisen from the server and that the client was going to be easier to organise in terms of OOP on a UML diagram as it had a more organised file structure where the code was separated out into various separate files and functions whilst the server just consisted of one source code file all within in one function.

My initial plan was to compile the files for the server and the client in the software and hope for some errors to come up in the debugger in visual studio 2015 along with obvious security issues and where I noticed obvious errors and security issues I would divide these into separate classes in terms of Object Orientated Programming and such in the form of a UML diagram for my proposed redesign. However, this was not doable, as I had no errors come up in the visual studio debugger when I compiled both the server and the client.

Now my plan is going forward is to find such security issues detailed in section 3 of this report in a more abstract and lateral way and perhaps using different software to analyse all the code part by part to see exactly what it is doing and exactly where it is flawed. I decided to focus on the security aspects the most in terms of the UML redesign because firstly I think this would be the most important in terms of a client to server and server to client software base, additionally when I ran the code I could clearly see there were no encryption at all what so ever which leads me to believe this area of the software is worth investigating further and thus redesigning in terms of a UML and towards the Object Oriented design of the software, additionally going this in depth into the code of the client and server should hopefully help me to find other issues with the code in terms of the redesign and improving it overall.

3 Security - Risks

To initially discover the security risks within this software I initially thought about it in a simplistic way, the software is made up of a client and server therefore data is being passed between the client and the server, the information that is being passed is of importance in terms of privacy, for example an IP address of the user and the server, the game moves and its data therein, none of these kinds of data show any form of encryption at all throughout the code which is of great risk to the user, especially the IP address which can be used to perform all sorts of different types of attacks on their computer. Session hijacking was also something that caused some concern for me in terms of a man in the middle attack, this could be prevented if more than 3 IP's are being used throughout additionally with a secure password at both ends for each move.

I then thought about it in a secure programming context in terms of the C programming language, where the most common type of security issue found in this programming language is buffer overflow, this was made obvious to me from the start where I could see that strings were passed to the type of char causing buffer overflows throughout the code files network.c and main.c (server only)

I then used a piece of software called Flawfinder on Linux/MacOS terminal to analyse the entirety of all the code files to confirm my initial impressions of the code, I did it on all files but network.c and main.c (server only) where the only ones which confirmed that there were buffer overflows throughout these files. This would be of a concern in a security context because this allows for that of a malicious hacker to purposely conduct such a buffer overflow attack, allowing them to inject their own code or sets of instructions into the buffer which then just wait for it to be called again by the user before executing.

Overall the following security issues with the client and server software need to be considered throughout my UML redesign amongst every code file and when attempting to mitigate such issues:

- Unencrypted data:
    - Player moves.
    - IP address of users and server.
    - General data about the game.
- Session hijacking, man in the middle attack
- Buffer overflow attacks.

3.1 Security – Mitigation

Below is an overview of OOP design and classes for the client and server:

	Classes are in their detailed form	Classes names are
1	Encrypt important data throughout the software and make it require an encryption/cryptographic key.	Encryption
2	Important data that is prevalent throughout gameplay	Game Data
3	Session hijacking / man in the middle attack prevention.	MitM Prevention
4	Prevention of Buffer overflow attack's.	Buffer Overflow Prevention

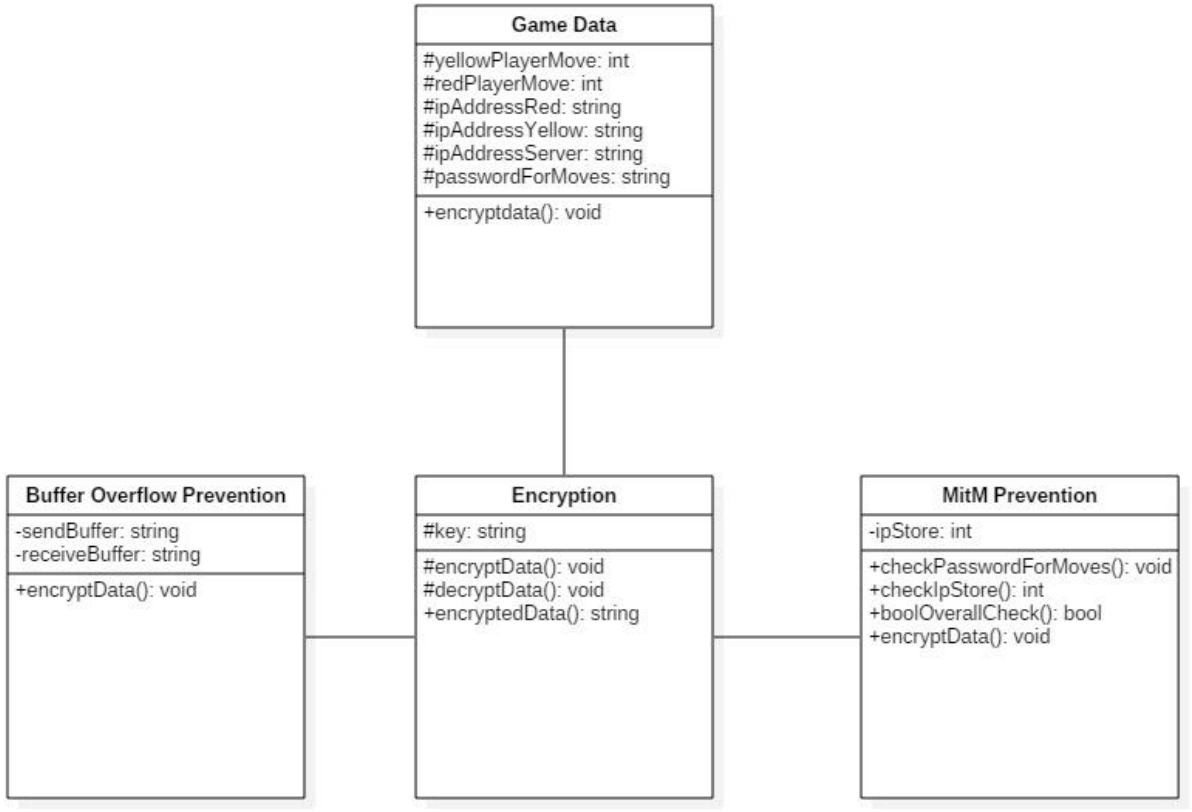
Each class will contain the following attributes and types for the client and server:

Name of class:	Attribute name:	Type:	Reason for type:	Public, Private or protected:
Encryption	key	string	Mixture of characters	protected
Game Data	yellowPlayerMove	int	Can store relative to grid coordinates	protected
	redPlayerMove	Int		protected
	ipAddressRed	string	Not actually used to calculate anything.	protected
	ipAddressYellow	string		protected
	ipAddressServer	string		protected
	passwordForMoves	string	Mixture of characters	protected
Buffer Overflow Prevention	sendBuffer	string	Type string manages own memory as opposed to type char.	private
	receiveBuffer	string		private
MitM Prevention	ipStore	int	IP addresses via client 1 & 2 and server = 3 (!=>3)	private

The above information should suffice for me to decide upon the overall UML design and its appropriate methods therein, therefore giving me a starting template to manipulate, and add to aspects like methods, objects etc… on an ongoing basis if need be.

4 UML

Below is the overall UML diagram which exhibits the classes respective to security mitigation.



The above diagram shows implemented methods as follows:

Method	Reason for Method
encryptData	Encrypts data
decryptData	Decrypts data
encryptedData	For handling encrypted data
checkPasswordForMoves	Asks each player for a password
checkIPStore	Checks there is no more than 3 ip addresses throughout the software
boolOverallCheck	Contains a sort of program that would change a bool to true if ip addresses are no more than 3 and both players have entered the correct password.

5 Transition

When decommissioning a piece of software there is usually a lot to consider especially in respect of the users. With this software, it would most likely not be used in a business environment with it being a game however things like costs and usability would still come into it. For this decommissioning scenario, I will explain the transition to the new software / game in a business context however because I believe this gives a more structured approach to implementing it.

The biggest issue I believe I would find initially with the users of this, would be them using the new software, therefore perhaps a policy, agreement or indication to them before they use it would suffice in terms of educating them about computer security and how the new version utilises new and better security methods better thus helps to protect their private and personal data. The biggest change to the software is its security therefore I believe this would be the best area to focus on in terms of changing over the software and decommissioning the old version of it.

With the new implementation of this software, it begins to go towards a more Object Orientated Design in terms of how it is coded therefore when changing the software over to the new version, if anyone who begins to keep the code up to date and maintain it especially in terms of security and compatibility, would need to be knowledgeable of Object Orientated Programming to prevent the possibility of creating accidental new security flaws and bugs.

The software usage most likely will need to be monitored initially to ensure that the users have no trouble with it, and that the security flaws are corrected and improved. Additionally, it is highly possible although the person coding my design may not find any bugs its highly likely that the more people who use the software the more chance there is of them finding bugs in the software, for example a user might click somewhere on the screen that no one has ever clicked before including the designer and myself thus finding a new bug that would be need to be investigated further and fixed.

Overall the initial functionality of the old vs the new application remains the same with the only key differences being of its new security features, the only thing affecting the user in terms of change would be need of a password for their initial moves to confirm validity of their machine in terms of the IP address.

6 Conclusion

I have redesigned the elements of this software that needed improving the most in terms of security flaws that I found throughout the client and the server. I found that there were 3 main areas of concern when it came to security which were data being publically available and non-encrypted, a man in the middle attack being able to be performed easily and a few buffer overflow errors.

Although the buffer overflow errors were obvious from the onset, I had to think about the other issues laterally in terms of a client and server and how they interact with each then by looking through the code and running it to confirm my findings.

I redesigned these elements towards an Object Orientated approach, by specifying 4 different classes, each class contained attributes and methods that were related to the improvement of the security software features.

Overall this allowed for a quicker and more effective way to deal with these security issues as well as allowing for any necessary future updates to be carried out in a much easier and understandable way.

7 Bibliography

(2017) *Buffer Overflow.* [Online] Webopedia. Available from: http://www.webopedia.com/TERM/B/buffer_overflow.html [Accessed 09/01/2017].