**6841 Project Report**

Intro note: all referenced files and repos will be linked at the bottom.

For my project I decided to make a Remote Access Trojan (RAT). This is a program that allows total control over a device. Some common functionalities of RAT’s are: Cookie Scrapers, Keyloggers, Clipboard Scrapers, Screen Capturing capabilities and Shells.

During the creation of my RAT I wanted to make a version of each of the above capabilities, however I have never written any form of server, GUI nor any form of malware before. So, everything I made during this session was brand new for me. I read lots of pre-existing RAT tools, primarily using a tool called SKYRAT as my base, I went through the code and used it to learn how server and malware programming goes, I also used their GUI as a base, however changed it to allow it to work with my own additions and changed it to a way I prefer looking at. I then went and made my own keylogger, as well as clipboard scraper. I then used SKYRAT’s Screen Capturing and shell capabilities as a base for my own.

**Shell:**

During this process I had many issues, as of the writing of this document my shell capability still has several minor issues, this includes only getting output to the main console back on the first command, despite commands continuing to run, and the closing not being smooth at all. As of writing (last updated ~ a week before due date) I am working on fixing these issues. Other than this, however the shell will work, it can get information such as Ipconfig and deliver this across the server so the malicious person can read the information, it can run all scripts that are runnable in Windows command prompt. While writing this, my windows defender was not happy about letting it run for testing, this became very frustrating as it would block functionality but not inform me leading me initially to believe it just didn’t work.

**Screenshot:**

This had the most iterations of all systems while working on the program. Initially, neither the infected nor the host were capable of creating any form of file. However, after much testing I got the client to be able to create a file, however the host was unable to read it. Eventually both were able to create the file, however the host would take nearly 15mins to be able to create a readable image file. I am still unsure why this was, however, moving away from being in its own thread it became nearly instant. It is because my internet was being incredibly slow during the test, however due to both being on the same pc I am unsure if this would have made a difference, however unthreading it (tested the day after) worked nicely. This allows the user to take a screenshot of the current active screen of the infected user. The file is also then named after the client ID.

**Keylogger:**

This was the first capability I made, the initial version would print the key code, however this was not useful. After trying to use the windows.h userinput structs, I decided that this approach was not feasible, after doing more research I found a youtube video on gamedev illustrating how to fix the problem I was having. Rewriting parts of the system and making it work with the server took a short amount of time, from there the system worked well. The file that the logger reads into currently only will read at the closure of the host, I have spent a very long time trying to fix this however I was unable to work out why this is, it is my understanding that using fclose() would write the text into the file however it only happens when the program stops fully. For special characters such as the return button and backspace button I decided to use tokens instead of symbols, for example return is written as: [return], this was done as symbols can be confusing and would require an explanation, an issue with this system would be if someone actually just wrote the key in this way, it would be impossible to know which has happened.

**Clipboard Scanner:**

This was the final capability I worked on, I used the windows.h documentation and was done very quickly. No issues were occurred during creation.

**Methodology for testing:**

I had a separate C file which was not connected to the server to test capabilities broken away from server programming, this allowed me to learn how the capabilities worked as well as test to ensure that an issue is on the server and not in the functionality of the capability. All functions were built and tested using this test first.

**Server Programming:**

I have never written a server in any programming language before, I used the SKYRAT server as well as the winsock2 documentation to write my server. Initially I had it sending text back and forth to ensure connection. An issue I have discovered is that the server can only communicate with the device its running on, I believe this is because of using “localhost” to run the socket on.

**Further additions:**

Initially the trojan file had lots of testing on a cmd window that opened up, however after I was confident everything was working correctly, using the windows.h documentation I worked out how to hide the cmd window. I then started working on it being capable of running on Linux, however as I do not have a linux system running I decided it would not be worth the time to rewrite the entire system and setup a test environment for the program. Potentially a further task for later.

**Final notes:**

During this project I have taught myself and become significantly more confident in server programming in C, as well as GUI programming. I feel I am a lot more confident in the ideas behind keyloggers and scrapers inside of a device, I also understand the backend of how shells work in a computer significantly more. Finally, I have learnt a lot about screen capture device, personally I believe this project has been an incredible help to my understanding of sockets, server programming and C programming in general. In the future I would like to add functionality for it to run on linux systems, as well as have it be able to close the shell correctly. On top of this, I want to work out how to make packets for packet testing in my recv functions as currently It just assumes everything is received correctly, despite the multiple threads running simultaneously potentially sending data over the socket and into the wrong place.

**Referenced files during the development:**

1. Skyrat: <https://github.com/pwnengine/SkyRat/tree/main>
2. Keylogging: <https://www.youtube.com/watch?v=n1SzitnRzY4>
3. Windows.h documentation: <https://learn.microsoft.com/en-us/windows/win32/api/winuser/ns-winuser-input>
4. Winsock2: <https://learn.microsoft.com/en-us/windows/win32/api/winsock2/>