Problem Definition:

What is a Rat :

Why:

Purpose:

Objectives:

Clear Scope

What did I plan to do

How did I go about doing it

Results:

What did I achieve:

Features and functionality

Originality

Conclusions:

Result Correctness towards intended purpose

Appropriate AI

Visualisation:

Algorithms

Reliability

Completeness

“Investigating the parallels between using a RAT Style Software for Malicious Purposes and Virtuous intent”.

To understand the scope of this project first you must be able to understand what a RAT is and what it implies.

RAT is an acronym, with two meanings distinguished by the context and purpose in which it was designed. The two meanings are: Remote Administration Tool, and Remote Access Trojan, while at heart the functionality of these two RATs are virtually identical, the different expansions represent the intent for how the software should be used. With further development of both these ideas, comes features exclusive to them – and features designed to enhance the experience.

Generally, a Remote Administration tool has moral and true intentions whereas a Remote Access Trojan is designed with baneful objectives.

RAT software is used for many reasons, the most common principled reason is as a support utility tool, this is quite commonly used by tech-support team in most major companies and establishments to provide remote support. Another common use is for file transfer from a remote location, or even remote access to a system, a common necessity induced by the COVID-19 Pandemic. RAT software usually makes up a smaller part of a larger system. This is also true for the less noble variants – which usually are installed with the intent of delivering a payload with even more malicious intent.

A RAT used for malicious purposes is also synonymous with the term “Back Door”, and is quintessential in most compromising attacks. here is a definition given by Virus Radar.

Here are some of the more famous RAT-style Software ranging from some of the most famous malicious trojans, to some of the industry standard support utility software including remote desktop style software. These can be written in nearly any programming language, with some of the ones shown here being Delphi, Python, java and even VBS. There are many factors which affect the choice of language used.

For this project Python was used. As shown on these graphs python has grown exponentially in popularity in the last few years, and is now one of the most used programming languages in the world. Usually, a lower-level language is chosen, and selected specifically for the target system, such as C# for windows or C and Bash for Linux, however Python has the benefit that it can be executed on any machine with a python interpreter making it a good cross platform solution.

The Implementation uses the Python Socket module as an interface to the Berkeley Sockets API, and opts for TCP Sockets as opposed to UDP. The reason for this is that TCP ensures the data is all received, and it is read in the order which it is written, which are both crucial to the implementation of many of the features of this program.

The application uses an object-oriented approach and was developed similarly to a Feature driven development model, where the client-server model was established first, and then the feature list was created and the features were iteratively added to the system.

In reality the feature design took more steps with a research phase, planning phase, design phase,

It was then built and tested in an isolated environment, before being integrated in a simplified abstraction of a client-server model, and then finally deployed into the main program.

Currently the system has the ability to perform all of the following functions:

Sending a console message, Sending a custom messagebox alert,

Remote shutdown of the target system, with an optional custom shutdown message,

Remotely locking the target system,

Remotely restarting the target system,

The program can make use of the Windows Telnet Client in order to perform functions like, Watching Star Wars: Episode IV: A New Hope.

Playing chess Via Telnet Connections,

Using a weather forecasting facility.

By default, on modern systems the windows telnet is disabled, so there also exists the functionality to remotely activate the telnet client.

The program includes a keylogger, which can record all of the keystrokes used by the target and saves it to a text file. This can be started or stopped on command.

Furthermore, the clipboard contents can be saved also, and when combined with the keylogger, a check can be performed on “CtrlC” and “CtrlV” shortcuts, which are used to copy and paste, - on entry of these shortcuts, the clipboard contents are written to the keylogger file. The keylogger runs on a separate thread, allowing passive capture of the input without halting the rest of the program.

These keylogs are retrievable with 2 main methods, one of which utilizes a gmail account to email the contents of a specified file – in this case the keylogs, however this works with other files too.

Multithreading has again been utilised to allow an “Automailer” to be set up, emailing the keylogger file contents every day at a scheduled time.

There also exists the option to exchange files, this can be sending folders from the client to the server or sending a file from the server to the client. This can be anything from images to zipped folders and more.

There is also a functionality to remotely run executables, applications, python files and other scripts like vbs. This can include the files send to the target as previously mentioned.

The next main feature lies in the ability to screenshot the target system, this could include multiple monitor set ups. From here, a screen recording function can be implied.

In a similar manner, the program enables the attacker to take a picture using the targets webcam, and also take a video recording, and play the recording back.

Finally, and possibly most importantly – the UoRAT enables the attacker to establish a “Shell”-like functionality, allowing the attacker to run commands on the Victims computer as if they had physical access to the computer, this is one of the most fundamental concepts for a good RAT style software.

Here is an image of running the client and server. You can see the server at the top, waiting a listening for a connection, then the client connecting to it.

Next is an example of running a command, on the left we have the server – displaying the list of possible commands, and a short description of what they achieve.

As highlighted by the arrow the command “-EpIV” is run which connects to a telnet server and plays an ascii animation of star wars EpIV as seen on the right, this is what the clients device would show.

While it is possibly to leave these files as python scripts, they can be turned into executables, and combined to have a front end program disguise the malicious intent of the true program, in this case I have used an executable from a Maze Game created in C++, with the client running as a threaded daemon, meaning that even once the maze game is closed, the client can remain running in the background.

Another Benefit of converting this program from a python script to an executable, means any requirements or dependencies are removed. Here you can see a comparison of all the libraries imports for the python script, vs the executable file which doesn’t even require a device with python on.