s390 words.

“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.

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 socket programming to connect the machines.

Currently the System the has the functionality to perform the following procedures. It can send a console message to the client, it can remotely shut down the client, it can copy and retrieve files and folders. It has the functionality of a non-interactive reverse shell, it can navigate through the file system, disconnect the device and send an email from the device.

The future Functionality and ideas for this system are as follows. The ability to send a file, the ability to run a file or script, take a screenshot, run a keylogger, obtain clipboard contents, retrieve the keylogger files, lock the pc. There is a possibility of adding extra Linux functionality too.

Some more features that could be implemented are: A Graphical user interface, either fully interactive or a Psuedo UI made in the console.

A time out feature.

An Acknowledge // Accept Connection Request

And a feature allowing the client to terminate at any point.

This is what the code looks like when it is run, the connection is established, then it prompts the client whether to run the malicious or virtuous mode, if the client choses virtuous mode, a password is generated, and the client must type it in to confirm the connection. This is not the case for the malicious mode. As shown here you can see the client has typed the code correctly and the keys matched.

Here you can see a message is typed into the server window which is then transmitted to the client window.

Here is an example of the reverse shell style implementation, with the command DIR run to demonstrate.

This shows the process to copy a file from the device and save it to a new location.