GOD-VIEW

Introduction

What is GOD-VIEW?

GOD-VIEW is a networking application, with the name referring to a view from up above, the all seeing eye. GOD-VIEW is the third iteration of developing a Remote Access Tool, driven by the motivation of understanding the web, it's foundation, but more specifically the communication between computers in networks & processes. This might seem simple, but it has a multitude of layers below when it comes to development of a truly great program. Gathering from my previous attempts which started with the Eagle-Eyes (https://github.com/Alvin-22/Eagle-Eyes) project & continued by NeoRAT (https://github.com/Alvin-22/NeoRAT), both of which are available on Github as open source projects with major flaws. But these projects provide the foundation of how communication between networks & processes can look.

In practice GOD-VIEW hosts a TCP socket that the client connects to & a HTTP server that communicates with the server through a websocket allowing real time communication.

What Does GOD-VIEW Achieve?

The GOD-VIEW program supports a large amount of powerful features, that I will discuss later on. But on a foundational level, GOD-VIEW allows for secure, fast & non-limiting communication between it's server & clients. It accepts connections using the TCP transport protocol & works with streams of data. It keeps these connections alive by sending beacon messages in intervals & provides persistent attempts to connect / reconnect if it fails to establish a connection or in the case that the connection is ever closed with the server. This might seem non-descriptive or difficult to understand what it means, but it will all make sense as I will in detail walk through everything the application archives & how the program works.

Who can use GOD-VIEW?

The program makes availability one of its core foundational beliefs. GOD-VIEW supports the Windows, Linux & Mac operating systems.
Both the server & client program supports these operating systems. The server also supports a Desktop GUI application that requires Google Chrome installed. This GUI application can also be used in a browser, this does not require Google Chrome installed. This desktop / web GUI supports all major browsers except for Internet Explorer with device support from 320x320+ (iPhone 5+) & supports mobile specific interactions when right clicks for example is not possible.
The program also supports both the IPV4 & IPV6 protocols & can be run from any directory on the filesystem (all paths are accepted, relative or absolute).
Overall GOD-VIEW is very flexible & supports all the major operating systems & different ways to interact with an application, be it through the terminal, desktop or web.
GOD-VIEW also supports running the program with an interpreter or as an executable.

What Makes GOD-VIEW Different?

The things that make GOD-VIEW different can be both seen as positive & negative. The major difference is the availability as mentioned. It also provides a neat & well thought out design, with soothing color schemes & responsive action-reaction design thinking. It provides for beginner-to-advanced usage, but always, even when used in the terminal with describing colors & never leaves with the “what is happening” interaction.
The desktop / web is built using sustainable methods & complete custom design.
The availability is not the only thing that is different, GOD-VIEW provides an extremely powerful zero-limitations of actions when it comes to performing a command to 1 or 100 clients at the same time, however taxing / complex the command is, with no extra delay between responses from every client. GOD-VIEW also never writes / stores anything on the client’s filesystem.

How was GOD-VIEW Built?

GOD-VIEW was built using the Python (blessing & a curse) programming language, following the object oriented paradigm, for the client / server & supports Python 3.7+.
The program also utilizes third party libraries, which are required, with specific intentions to avoid bloating the program. These can be installed using pip / direct download link to a .whl file. The program uses multiple formats to store data, using sqlite3 for the local database, text files, csv files & the PNG format for images.
To build the executable file requires pyinstaller to be in the path variable.

The desktop / web GUI was built using ReactJS with TypeScript**[front-end]**, redux for global state management & styled components for the custom design. It communicates with the server using websockets **[back-end]**.
As mentioned previously, the desktop GUI requires Google Chrome installed & if it’s not available will default to web GUI.

What are GOD-VIEW's Features?

Operating Systems

- Windows
- Linux
- Mac

IP Versions

- IPV4
- IPV6

Usage

- Terminal (Command Line)
- Desktop Application
- Web Application
    - All Major Browsers
    - Responsive Design (320x320+)
        - Mobile friendly
    - Login Protection

Data Transport

Protocol

- Transmission Control Protocol (TCP)
    - Transport Level Protocol

Encryption

- Asymmetrical AES128
- [Not SSL]
    - Self-Signed Certificates Require OpenSSL
    - Requires Certification Expiration Date
    - Requires DNS Hostname

Compression

- Deflate Compression
- PNG Image Compression
- [Not JPEG]
    - Can be Less Effective
    - *Implementations difficulties*
    - Supports Lossy Compression

Encoding

- UTF-8
  - Windows Code Page Support
- Base64

Serialization

- JSON
- [Not Pickle]
  - A lot Easier Usage
  - Allows Remote Code Execution

Surveillance

GOD-VIEW supports multiple ways of gathering large amounts of information, with 30 data points upon initial connection, this is among other things the country, operating system, antivirus & privileges of the connected client.
There is also more specific information that can be gathered about the clients which are about the GPU, CPU, Memory, Disk, Network & IO. **[Sysinfo]**
GOD-VIEW provides the ability to upload files from the server or an URL & the option to execute this file upon upload. It’s also possible to download files or directories to the server & execute the file upon successful download. **[Upload / Download]**
The recover commands allow you to get the WIFI passwords stored on the client's computer, the browser history & bookmarks of most browsers. **[Recover]**
You can capture a screenshot of one or all monitors & have the options to display the image on the server. You also have the option to take a snapshot from the webcam, working the same as taking a screenshot, but without the option of taking a single snapshot of all connected webcams. **[Screenshot / Snapshot]**

Action

The action namespace holds commands with the intent to perform specific system actions.
You can shutdown, restart, logout, hibernate & standby the clients computer. **[System]**
You can alert the client’s computer with a messagebox, of which you can set the title, the text & specify the symbol of the messagebox. The symbol can represent a question, information, danger or warning. **[Alert]**
The process command allows you to list all running tasks & the clients open network connections, providing information about them. You then have the option to kill anyone of these processes. **[Process]**
The clipboard command allows you to copy data to the clipboard, empty the clipboard & get the data of the clipboard. **[Clipboard]**
The browse command allows you to enter one or more URLs to be opened in the default browser of the client system. **[Browse]**

Execution

When it comes to the execution namespace, it holds some of the most interesting & vital commands. It allows inline injection of keyboard / mouse actions or running a script from a file. The language of this injection is GOD-VIEW’s own. It’s very simple & not very strict.

The language consists of key, value pairs, separated by a space.

press [Key / Button]
hold [Key / Button]
release [Key / Button]
type [Text]
position [X,Y]
move [X,Y]
scroll [X,Y]
mhold [Mouse Button]
mrelease [Mouse Button]
click [Mouse Button]
dclick [Mouse Button]

Unique flow control commands

- **sleep [Seconds]**
- **repeat [Times]**
    - Repeats all subsequent actions for Z times, this is recursive & allows nested repeat statements.
- **[position / move / scroll] random [min X, max X,min Y,max Y]**
    - Can be used by any command taking an X & Y parameter, and requires instead of X & Y four values, for the minimum & maximum random value for the X, respectively the Y.

The injection command can be used to perform automated keyboard / mouse activity, supporting recursive loops & random coordinates, making it quite a powerful tool. **[Inject]**

Next there is the python command, allowing the inline execution of Python code. Everything that would be written to the terminal on the client's computer will instead be read on the server, making it seamlessly natural. This command also supports, like injection, execution from a file. **[Python]**

Finally, arguably the most important command in GOD-VIEW, is the shell command, allowing the remote execution of terminal commands. A reverse shell, supporting all commands. Commands that do not return any data will time out, after Z amount of time, therefore interactive programs like Python or Node would simply timeout. **[Shell]**

Management

The management namespace is about simplifying tasks & specifying how the program shall operate. This is done with the blacklist command by allowing the rejection of specific connecting IP addresses. The alias command allows the translation of a word written using the server's terminal to a corresponding value. Then there is the very powerful autotask command, giving great freedom & automation, for the simple reason that any command that interacts with clients is supported. It will perform the request upon connection of the client & write the results to a text file. Finally, also one of the most important & influential commands is the environment command. It holds a number of variables that have a great impact on how the program will run, for example alerting an email upon client connection. **[Blacklist / Alias / Autotask / Environment]**

Connection

The connection namespace focuses on the handling of anything that specifically handles the connection or the file on the client's filesystem. Starting off by targeting persistence of the client program, by having the program run even after the computer shuts off or the user logs out. This is done with the autostart directory, to schedule a task, or by adding an autostart registry key, these features are Windows specific & won't work on any other operating system & would require other implementations. **[Autostart]**
The session & delete commands allow specifying which clients to either interact with or to remove. **[Session / Delete]**
Additional quite straight forward commands are disconnect, reconnect & uninstall. Disconnect might seem like the delete command just mentioned, but has the specific difference in that it makes the client exit itself, which won't have the client reconnecting until it's rerun, which delete does not always guarantee. Uninstall removes all registry keys made for persistence along with any autostart files & finally removes itself. **[Disconnect / Reconnect / Uninstall]**
Lastly we have an important command, only supported in Windows, again, which is escalate. It will try to perform an user authentication bypass (UAC) bypass, rerunning the program with administrative privileges without ever having to ask the user. **[Escalate]**

Utility

The utility namespace is very straight forward, with a couple of commands that support the general flow of the program. The build command allows you to build GOD-VIEW into an executable program, being able to build any file, specify the icon & if it should have a window or be a background process. **[Build]**
Next is the note command, this simply writes the data to a text file, with an attached timestamp. **[Note]**
The who command is especially neat for the terminal, as it will list every single data point of a specific connected client. **[Who]**
The gui command will simply reopen the desktop application, to be used in case you close the window. **[Gui]**

Live

Live is one of the most challenging namespaces, these are the most powerful commands GOD-VIEW has to offer, providing live streams of data, each with their own socket connection. This holds a difficult challenge of having different types of client connections to the server if the server is to be only hosted on a single socket, or if multiple sockets should be opened on the server for each one of these live streams of data. You also have the problem of not wanting the client to be able to dictate when a certain stream of data should be opened, as the client has to connect to the server first. The solution was a token-like system, generating a universally unique id on the server side, allowing the connection with this token to be accepted if sent by the client within X amount of time, of which the token is valid, this requires the server to always initiate the request of a specific connection type from the client & is a great solution for the reason of only requiring a single server socket.

The audio command enables the microphone, allowing a live audio stream of the microphone to the server & be directly streamed to the default output device. Making the client a hot-mic.

The next command is desktop, it provides an extremely powerful & relatively smooth stream, the FPS is not going to be the highest, the performance will be based on the screen size & the power of the computer. It's optimized for performance, but because it uses PNG compression, it does not support lowering the resolution, which can make the stream performance very image-size dependent. The stream is optimized by splitting up the work of taking the screenshot & sending it, making it more resource intensive, but with performance improvements, but is still throttled by the TCP-protocol bottleneck. **[Desktop]**

The modules command provides an overview of all running live streams, with the ability to close any stream. **[Modules]**

The webcam command works very similar to desktop & usually has very good performance, this is because the webcam size is usually a lot smaller, making the frame rate much higher. This is only supported on Windows though, because it's not utilizing opencv, to avoid bloating the program on the client side, instead a library called VideoCapture is used, which uses the Windows DirectShow API. **[Webcam]**

Finally we have the logger commands, keylogger & clipper. These commands record keystrokes & the clipboard data to be written to files on the server, providing client side timestamps & fair representation of performed actions. **[Keylogger / Clipper]**

Multiplex

The multiplex namespace has only two commands, all & close. These will simply add every connected client to the session or remove every single client from the session. **[All / Close]**

Terminal

This namespace refers to commands that are primarily used in the terminal of the server, as it's features are utilized by default in the desktop / web GUI. Alternatively are they specifically to be used only by the terminal. The clear command simply clears out the terminal, the help command lists all available commands, listing their names, arguments & all the data necessary about the command to be used. Then we have the list command, which lists data about all the connected clients & finally the exit command which simply closes the program. **[Clear / Help / List / Exit]**

○

The Different Ways of Running GOD-VIEW

The GOD-VIEW program accepts a few command line arguments. These are to specify the type of way you want to run the program, but cannot be given the freedom to be changed like environment variables during runtime. These are the -TERMINAL argument, which only runs the terminal of the server, without the desktop / web GUI. Then there is the -WEB argument, which will run the program with the terminal, but also hosts the web GUI, this will limit some functionality, but can be used in the case that the primary usage will be with a browser. The disabled features are to show screenshots / snapshots taken, nor can the desktop, audio or webcam command be used. By default GOD-VIEW runs as both a terminal & desktop / web GUI.

You can specify the address with -IP, TCP port with -PORT & the HTTP port by specifying -GUI_PORT. You may also specify the -SECRET & -SALT of the asymmetric encryption key to be used by the server.

The desktop / web GUI does not support exactly every command, you cannot exit from the GUI, it’s only available to the terminal, you can also not clear the terminal window of the server on the desktop / web GUI.

There are slight limitations when running the client as a non-executable file. You cannot do anything that requires the executable file to exist. This includes autostart, reconnect, uninstall & escalate.

Program / File structure

host.py

- Starts the server threads, this includes hosting the TCP server, the messaging thread & the desktop / web HTTP server.

bot.py

- Starting the main thread, connecting to the remote server, which is started by running the host.py file.

archive

- Contains all program data, server logs, database file & client specific files. The files are automatically generated & the file structure organized as such, this file can be deleted at any time, without ever disturbing anything during runtime.

build

- Files that are related to running & installing the program & acts as a miscellaneous directory.

gui

- The necessary desktop / web GUI production build files, required to be in the root directory of which the program is run.

exe

- Stores the executable build files generated using the build command.

client

- Stores client specific behaviour & implementations. To support all the client functionality & to fulfill requests.

server

- The server specific files, but also the GUI application. This is a large majority of the program, as this handles everything, not just following directions like the client.

shared

- Files that both the client & server can share, to reuse the same implementations, this has a lot to do with sending & receiving the data & shared state.
-

GOD-VIEW Outside of the Local Network

To be able for remote connections to be made to the server & to host the server for the internet is quite simple. With three alternatives, either use a port tunneling service like ngrok, which practically does the port forwarding for you. The second choice is to do the port forwarding of your router yourself. The third option would be to remotely host the program & enable the -WEB command line argument.

Ngrok

- Download ngrok at https://ngrok.com/
- Make sure ngrok is in the path variable or in your current directory
- **ngrok tcp -region=eu 5658**
  - Running this makes the TCP server available to the internet.
- **ngrok http -region=us 8565**
  - Running this makes the HTTP web GUI available to the internet.

After this, you would simply change the Static.IP & Static.PORT in the GOD-VIEW/client/state.py file to the IP & port ngrok redirects traffic from, for example tcp.eu.ngrok.io & 1832. Note that if you're using ngrok to access the web application, it does not support HTTPS, so you have to use the HTTP version. This is sadly the library limitation of which is used for the GUI.

What GOD-VIEW Doesn’t Implement

There are things not implemented, this is because it might not fit the flow of the program, it requires third-party libraries not worth using for the features it would bring. The Python language has difficulty implementing certain features neatly, or it’s difficult to implement cross platform solutions for.

Real-Time Messaging Protocol (RTMP) server, supporting real time streaming. There are great servers for RTMP, one of them being Node-Media-Server (https://github.com/illuspas/Node-Media-Server), but uses NodeJS which makes it difficult to implement interactions with the server. But it also requires large streaming libraries, most commonly used being FFMPEG, which is a big hassle. But the GOD-VIEW GUI still allows you to read RTMP streams supporting flv.js (https://github.com/Bilibili/flv.js/) using the sidebar.

Another feature not implemented is a true remote desktop, most likely not terribly difficult to implement using the pywin32 library, but because it might clutter the flow of the program & also focuses a lot on specifically Windows only functionality, it’s not been tested or implemented.

Toast notifications, with my will of not using the pywin32 library being strong as it would simply put so much focus & possibility to Windows only functionality. GOD-VIEW worked around these problems with notifications by implementing server terminal connection notifications, the possibility for email notifications & GUI notifications with audio sound & a visual alert.

GOD-VIEW avoided a lot of the malicious intent features, such as more extensive password grabbers / cookie stealers, specific features targeting DDOS attacks & reverse proxies. Simply because of the lack of need for any of these features, interest in them. It does not include anything to do with encryption of files, as it’s simply not the intent of the program. Avoiding more ~basic features, such as the active window for the keylogger / clipper & fun functions, changing wallpaper, playing sounds, CD-popout along with other things, are not implemented as I cannot be bothered, a lot of these require operating system specific implementations or Windows only implementations & the pywin32 library. Still wanting to hold most functionality cross platform.

Lastly, what's not implemented is any virtual machine (VM) detection & obfuscation. This is also not the intent of the program, but also, especially VM detection from what I've seen are implementations supporting Windows only. The program does not try to obfuscate anything, it’s written in Python, not C, C++, C# or Delphi. It can be done, but has very little purpose to be supported by GOD-VIEW.