Offensive Cybersecurity Framework

NIGHTMARE FRAMEWORK

CYBERSECURITY

Author: Pablo Diez
Contact: pablodiez024@proton.me

Offensive Cybersecurity Framework

INDEX

INTRODUCTION

Nightmare Framework: A Powerful Offensive Cybersecurity Framework

Nightmare is an advanced **offensive cybersecurity framework** specifically designed to conduct **Advanced Persistent Threat** (**APT**) **simulations** and **penetration testing** on high-level systems. Its modular architecture and **powerful toolset** make it the ideal choice for professionals and cybersecurity firms looking to carry out comprehensive security assessments.

This framework is intended for use in **controlled, ethical environments**, enabling **Red Teamers**, **ethical hackers**, and **security consultancies** to identify vulnerabilities and **strengthen system defenses**.

Key Features:

- 🛡️ **Secure Communication**: Key exchange using asymmetric encryption (RSA) and message exchange using symmetric encryption (AES).
- 🐞 **Advanced persistence**: Windows registry (**without** administrator **privileges**) and services (**with** administrator **privileges**).
- 👇 **Dumping**: Password, cookies, software, WiFi passwords, processes, etc.
- ⚙️ **Modular architecture**: loading modules (DLLs) directly from the registry to avoid leaving traces on disk.
- 👁️ **Graphical interface**: Manually developed using only WinAPI calls (no external frameworks).
- 📋 Among others.

Nightmare is available on demand and offered with a Non-Disclosure Agreement (NDA) to ensure ethical and professional use of the source code.

FEATURES

Nightmare Framework

🧩 Dynamic Dependency Resolution in Memory

My Remote Access Tool (RAT) is engineered for maximum efficiency, portability, and stealth. One of its key architectural strengths lies in how it handles third-party dependencies: fully in memory, without requiring traditional system-level linking or DLL deployment.

The client executable weighs just 93KB, yet it includes full support for powerful libraries like libsodium (encryption) and cJSON (data serialization) —thanks to our in-memory dynamic dependency resolution system.

What This Means for You?

- ✅ **No File Footprint**
  - My RAT loads essential libraries directly from encrypted memory blobs. No DLLs are written to disk. This minimizes system footprint and reduces the risk of detection by file-based security tools.
- ✅ **Superior Evasion Capabilities**
  - Unlike traditional malware that relies on LoadLibrary or visible DLLs, our tool mimics the Windows loader internally—resolving functions from memory-only payloads. This makes it far more evasive to signature- and behavior-based antivirus (AV) and endpoint detection and response (EDR) systems.
- ✅ **Portable & Self-Contained**
  - Everything required to run—encryption routines, networking features, and JSON handling—is embedded and resolved in memory at runtime. This enables seamless deployment, even in restricted or sandboxed environments.
- ✅ **Minimal Registry Usage, Admin-Free Deployment**
  - No installation or admin privileges are required to run the client. All external libraries are loaded dynamically and resolved from memory, allowing the tool to operate independently of the standard Windows loader—making it fully compatible with restricted environments and strict group policies.

FEATURES

Nightmare Framework

🧩 Dynamic Dependency Resolution in Memory

How It Works (Simplified)

Instead of depending on Windows to load external libraries, our client:

- **Embeds libraries** internally as encrypted memory blobs.
- Loads and links them at **runtime**, simulating the **Windows PE loader**.
- **Resolves functions** via a custom in-memory resolver (equivalent to GetProcAddress).

This is done transparently, with no additional DLLs or system changes required.

Key Benefits to Your Organization

Benefits	Impact
⚙️ Self-contained binary	Easier deployment & minimal integration work
🥷 Low detectability	High resistance to modern AV/EDR due to memory-only ops
🚫 No external DLLs needed	No risk of missing or misconfigured runtime dependencies
🧩 Runtime flexibility	Supports modular updates and function resolution on the fly
📦 Lightweight footprint	Entire client < 100KB with full crypto and JSON feature

FEATURES

Nightmare Framework

💾 Persistent Cache

The **persistent cache stores** data that is used **repeatedly** to **avoid unnecessary queries**, **reducing system load and improving response times,** the framework allows to execute the commands withou using the temporary cache. This type of cache is used in **two key areas**:

- **Geolocation Data**: In the **GeoLoc Execution feature**, when a location is **selected on the map**, the **results of geolocation queries** are **stored persistently**. This **prevents** the need to **make external requests** every time **geographic information is needed**, **reducing traffic**.

- **Port Scan Results**: The **results of custom port scans**, which **can be slow and resource-intensive**, are **stored in the persistent cache**. When a future session performs the same scan, the **previous results are reused**, **avoiding** the need **to perform the scan** again unless explicitly cleared or overwritten by the user.

🗃️ Temporal Cache

The **temporary cache** stores the **results of functions** whose data is **not expected to change frequently**. This approach **optimizes execution** time by avoiding repetitive executions of functions that have already produced results while **still allowing access to this data when needed**. The following functions are stored temporarily in this cache:

- **Network Scan**
- **System Information**
- **Antivirus Check**
- **Stored WiFi Credentials**
- **Persistence Status**
- **Browser Cookies and Password Dumps**

This **persistent** and **temporary** cache system ensures that frequently used and **relevant data** is readily available without the need to repeatedly execute functions, thereby improving the overall efficiency of the RAT.

FEATURES

Nightmare Framework

📋 Default Commands

The **default commands** allow us to specify a series of functions to execute **automatically** when a **session is established**. Once up to five functions are configured, they **will be triggered sequentially** as soon as the connection is made and the handshake is complete.

This feature is highly useful for **automation**, especially from a **network administrator's perspective**. Once access is gained, the tool can be deployed, and all **predefined tasks will run automatically** in an encrypted and controlled manner, streamlining the process.

🚨 Global Execution

Similar to the "**default commands**" feature, this function enables the **GLOBAL execution** of up to five configured tasks. It streamlines **automation** by allowing all required functions to run sequentially, eliminating the need to execute each one individually.

🗺️ Geo Execution

Geo Execution enables the execution of up to five functions based on the **device's geographic location** or **ISP**. This is particularly useful for organizations with **interconnected offices** across multiple **locations** (VPN, Proxy, Tunnel, etc.). Once access is gained to devices in different regions, functions can be **automatically triggered according to their geographic details**. The available options for location-based execution include:

- **Country**
- **Region**
- **City**
- **ISP** (based on **latitude** and **longitude**)

This feature allows for tailored execution based on precise location or network provider, enhancing automation across diverse environments.

FEATURES

Nightmare Framework

📋 Search Functions

The framework offer a function that enables session filtering based on various criteria. When running the server in GUI mode (graphical interface), a popup allows you to filter by:

- **ID** (Session ID)
- **IP Address** (Internet Protocol Address)
- **Country**
- **PC Name**
- **Operating System**
- **CPU** (Processor)
- **GPU** (Graphics card)
- **RAM** (Memory RAM)
- **ISP** (Internet Service Provider)

In **CLI mode** (**console**), the filtering options are more **limited**, allowing you to filter only by: PC Name, country, region, city and ISP

This flexibility allows for easier session management depending on your environment.

📋 Remote Shell

The **Remote Shell** function allows executing **PowerShell** commands remotely. In **GUI mode** (Graphical Interface), the framework has an **interactive shell** window with **command history**, providing a **fully interactive experience**. In **CLI mode** (Console), however, there is **no command history** or the ability to navigate through previously typed commands.

This feature **creates a process** where both the **input and output** of commands are **controlled by the server**, allowing for seamless execution and retrieval of previous outputs. For example, if you run **`$p="example"`** and then execute **`echo $p`**, the output will display the assigned variable's value, in this case, "example".

FEATURES

Nightmare Framework

📥 File Download

The framework has a **dedicated function for downloading** files from the remote machine, available in **two modes**:

- **GUI** (**Graphical Interface**):
  - Open the **remote explorer** (explained below), where you can **easily download** files through an **intuitive interface**.

- **CLI** (**Console**):
  - Simply **specify** the **file** you wish **to download** and the **destination path** will be a specific path in a dir called "**DATA**".

This function includes **file size verification** and **displays** a **loading bar** in both **GUI and CLI** modes for **better user experience**.

📤 File Upload

The framework has a **dedicated function for uploading** files to the remote machine, available in **two modes**:

- **GUI** (**Graphical Interface**):
  - Open the **remote explorer** (explained below), where you can **easily upload,** files through an **intuitive interface**.

- **CLI** (**Console**):
  - Simply **specify** the **file** you wish **to download** and the **destination path** where it should be saved and its done.

This function includes file **size verification** on the client side and displays a loading bar in both GUI and CLI modes for better user experience.

⚡ Exec Function

This only works for CLI and let you execute a command without shell mode

FEATURES

Nightmare Framework

🪟 System Information

The **System Information function** provides **key details** about the **hardware** and **software** of the **remote system**, including:

- **Operating System**
- **PC Name**
- **RAM**
- **CPU**
- **GPU**
- **Main Disk Space**
  - **Free Space on the Main Disk**

This information is **cached temporarily**, preventing the **re-execution** of functions when the output is **unlikely to change**, **thus reducing unnecessary load** on the **remote system**.

🔒 Persistence (Without Admin)

Persistence **without administrator** permissions is based on a simple **registry entry** and can be **detected in startup applications**.

☣️ Persistence (With Admin)

The **persistence mechanism** with **admin-level permissions** is achieved through the **creation** of a **Windows service** that is configured to **start automatically** when the **system boots up**. This service is **executed under** the '**NT AUTHORITY/System**' account, which is the **highest privilege level available in Windows**, **providing unrestricted access** to the entire system.

By running with these elevated permissions, the service ensures that the RAT remains **active and functional** even after **system reboots**, without requiring manual intervention. This method **offers** a **robust** and **stealthy** way to **maintain persistence** on the target system.

FEATURES

Nightmare Framework

✅ Check Permissions

This simple function **allows us** at any time necessary to **observe what level of permissions** we have on the **remote machine.**

🚫 Block & UnBlock 🔓

For this function the framework require administrator permissions on the system in order to successfully lock or unlock the keyboard and mouse on the remote system.

This can be useful when running the RPD function (explained later) so that the user does not interfere with the control.

🥷 Chromium Browsers Dump

The "**Chromium Dump**" feature is **designed** to **extract and retrieve** sensitive data, such as **passwords and stored cookies**, from **Chromium-based browsers** such as **Chrome, Brave, and Edge**. By interacting with the browser's local data files, it **accesses the user's encrypted data** and then uses the **browser's internal decryption services** to **retrieve** the **information**. This process involves **interacting** with **system resources**, including the **browser's ElevationService**, to **securely extract and decrypt the data**. The function is very effective at collecting critical information, while **maintaining system integrity** and ensuring that the data recovery process is secure and efficient.

The **entire decryption** process is **handled server-side**, **minimizing** the **footprint left** on the **remote system** and **helping evade detection** by security solutions. Retrieved data is **structured and stored** in organized **directories** within the server environment, allowing for quick access and analysis. Additionally, the operation leverages temporary processes and controlled resource management to ensure that no residual artifacts are left behind after extraction, preserving operational stealth and system stability.

FEATURES

Nightmare Framework

🎯 Gecko Browsers Dump

The “**Gecko Dump**” feature is designed to **extract and retrieve sensitive data, such as passwords and stored cookies**, from **Gecko-based** browsers such as Firefox, **LibreWolf** and **WaterFox**.

In this case, the **decryption process** is done from the **client-side** because the program uses the i**nternal DLLs of each browser** to access the functions that **decrypt the passwords** of all detected profiles that have the necessary files.

By using the browsers' own dll's the framework **avoid importing functions directly** into the binary that **can be conspicuous for antivirus software**.

WiFi Passwords Dump

A process is used to retrieve saved Wi-Fi credentials on a Windows system by connecting to the wireless management service, listing all available network interfaces, and extracting the SSID and password from each stored profile. The information is parsed from XML data and saved in memory. Proper cleanup of resources is performed at the end to maintain system stability.

🔄 Processes Dump

The client registers all the processes in execution at the moment this function is called, once it has all the processes listed it generates a JSON in order to serialize the message to send it to the server so that the server can save it and display it comfortably.

💻 Software Dump

By means of registry entries, all the software installed in the remote system is collected together with its version so that a JSON is generated to send it to the server in order to display and save it comfortably.

FEATURES

Nightmare Framework

🚨 Dump All Function

The **Dump All function** enables the **automatic collection** of **all** sensitive **information** in a **single operation**, designed specifically to streamline and automate data gathering processes. This function extracts the following data:

- **Chromium passwords and cookies**
- **Gecko passwords and cookies**
- **Saved WiFi credentials**
- **Running processes**
- **Installed software**
- **Exodus wallet data**
- **Monero wallet data**

💡 Display Message Box

The MessageBox function provides a **straightforward capability** to **display custom pop-up messages** on the remote system. Initially **incorporated during** the **development phase**, it has remained available for specific use cases. This feature **can be leveraged to simulate system errors**, provide **deceptive prompts** to **influence user behavior** (such as encouraging the execution of a previously uploaded .exe or .bat file), or **create controlled distractions** during engagement. Although not critical to core operations, it offers additional flexibility for social engineering tactics within authorized scenarios.

💻 Screenshot Function

Generates a **screenshot** of the **main screen** that will be **saved** in the user's **temporary folder UNTIL** it has been **downloaded** by the server, once it has been downloaded it is **automatically deleted**.

This is useful to **know what the user is doing** without having to open an RDP session.

FEATURES

Nightmare Framework

🔴 Record Mic Function

This function allows us by means of **WMI** to **generate** an **audio** file of the **amount of seconds that we want**, the requirements for this is that the system has an **available microphone**.

Automatically it uses the **functional microphone** or **configured as main** microphone.

🔎 Scan Network Function

This function performs an **ARP-based** scan to **identify all devices** connected to the local network. When the RAT is executed in **GUI mode**, a **visual network map** is generated to **provide an intuitive overview** of the detected devices, enhancing navigation and targeting capabilities.

The network map is **developed in C++** and benefits from **hardware acceleration** to ensure smooth rendering and optimal performance.

🔎 Scan Host Function

This function **requires a prior ARP scan** of the **network devices**. In GUI mode, the option to perform a port scan is only enabled once the initial network scan has been completed, ensuring a smooth workflow. In CLI mode, while the function can be invoked manually, **a warning will indicate if a device scan has not yet been performed**.

The port scan process **may take some time**, depending on the range selected, as it **allows** users to **specify** both the **starting** and **ending ports** for a more targeted and efficient scan.

Ⓜ Monero Wallet Function

This function use the system registry to identify the locations of the Monero .keys files. Once detected, it download these files for further analysis

FEATURES

Nightmare Framework

Exodus Wallet Function

This function identifies **common installation paths** to locate the Exodus wallet's **.seco** files, which contain **critical wallet information**. Once located, the function downloads these files for **subsequent analysis**.

These capabilities were **implemented** as part of the **development process**, with the **understanding that some organizations may hold cryptocurrencies** as part of their **legitimate business operations**, ensuring compliance with **relevant legal frameworks**.

Anti-Virus Detection

Using **Windows Management Instrumentation** (WMI), this function identifies all **antivirus software** installed on the system. It collects detailed information about each antivirus, including whether **real-time protection** is active, the s**oftware's update status**, the **installation path**, and the **date of the last update**.

Remote Desktop Protocol (Custom)

I have developed an **ultra-light Remote Desktop Protocol** (RDP) designed specifically to **avoid detection** by antivirus software. This custom protocol **operates over TCP**, ensuring it remains **unobtrusive** and **minimizes** the chances of being flagged by security systems. The protocol is designed with **efficiency** in mind, **maintaining** a **lightweight** structure to ensure **quick** and seamless **communication**.

It utilizes **five** distinct **message** types to facilitate the exchange of data between the client and server. These messages are crafted to ensure that the **communication remains efficient** while also reducing the likelihood of being detected by traditional security mechanisms. The protocol's design prioritizes stealth and performance, making it an ideal solution in environments where security and speed are paramount.

FEATURES

Nightmare Framework

📷 WebCam Function

This function opens a dedicated window to display the **live feed from the client's camera**. It has been developed in **C++** to **ensure** maximum **compatibility** and **configurations**. Before attempting to access the camera, the function **performs checks** to determine if the **device is available** or currently in **use by another process**, ensuring stable operation and avoiding conflicts.

If no camera is detected or if it is already occupied, the system handles the situation gracefully by notifying the server, maintaining operational reliability. This feature prioritizes both performance and compatibility, making it a robust solution for interacting with client hardware .

🎙️ Real-Time Audio

I have developed a **custom socket-based system** that, upon the client's connection, begins streaming audio data in **real time** directly to the server. The server then plays this audio through the default output . This method ensures low latency and continuous transmission, allowing real-time monitoring of the client’s audio environment. The system is designed to be lightweight and efficient, minimizing resource consumption while maintaining high reliability during the streaming process.

📁 Explorer

This function is available only in GUI. A custom client-server file service has been developed, enabling a set of operations such as LIST, DOWNLOAD, UPLOAD, and creating folders. All communication is serialized in JSON to ensure easier message parsing and readability. Upon initialization, the client sends a structured dataset containing information about all available disks to properly populate the main window.

The primary table (listview) supports two viewing modes: Disk and File view, allowing seamless navigation between disks, folders, and files.

FEATURES

Nightmare Framework

💎 Force UAC

This function is somewhat intrusive, as it continuously prompts the user for administrator privileges through the UAC (User Account Control) dialog. It operates in a loop, persistently requesting elevation until the user grants administrator rights, ensuring that the necessary permissions are eventually obtained.

🟫 CPU Monitor

This function is exclusively available in the GUI mode. When opening the control panel for an individual session, a secondary connection is established to continuously monitor the system's CPU usage. The client sends CPU usage data, measured as a percentage, to the server every 2 seconds, ensuring real-time performance tracking within the session interface.

🌩️ RAM Monitor

This function is exclusively available in the GUI mode. When opening the control panel for an individual session, a secondary connection is established to continuously monitor the system's RAM usage. The client sends RAM usage data, measured as a percentage, to the server every 2 seconds, ensuring real-time performance tracking within the session interface.

♻️ Clear Cache

The sole purpose of this function is to clear the persistence cache. Since the system automatically utilizes the cache whenever it is available, there is no option to disable its use. Therefore, if the operator wishes to discard the cached data or refresh it with updated information, it is necessary to manually clear the cache through this function.

FEATURES

Nightmare Framework

💎 Idle Function

The primary purpose of the idle function is to monitor the connection latency, ensuring it remains below 500ms. While its use is optional, it is mainly integrated into the GUI to continuously verify that the connection remains active, responsive, and functioning correctly.

🔌 Re Connect

This function is designed to force the client to reconnect to the server in cases where the connection has become desynchronized. It achieves this by creating a new socket and establishing a fresh connection to ensure stable communication.

🕸️ Uninstall

This function is responsible for cleaning up all artifacts generated by the server, including DLL entries in the registry, persistence mechanisms, memory allocations, and any other related resources. Its purpose is to ensure a complete and secure removal of any traces left on the system.

🚪 Close session

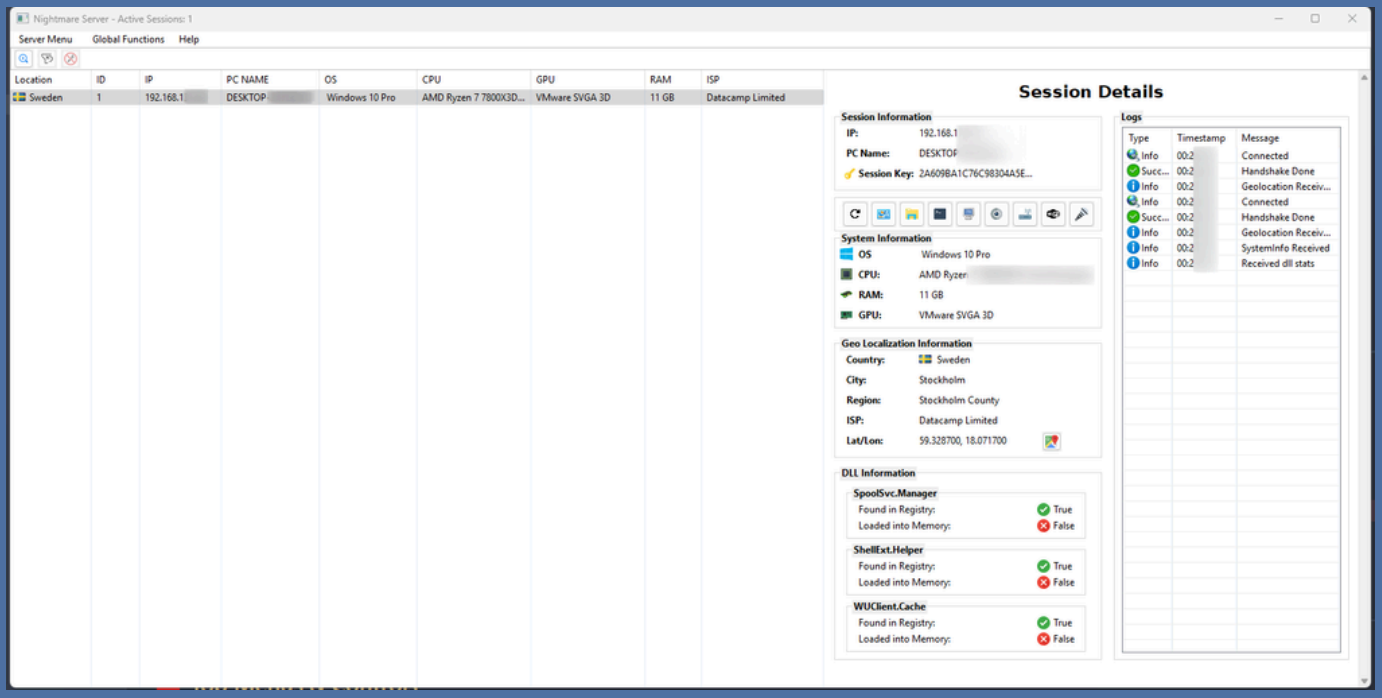
This function instructs the client to terminate the connection and unload all modules previously injected into memory. If persistence has been configured, the system will automatically re-establish the connection after a reboot, with all relevant modules restored from the registry.

With this, I have provided an overview of the general functions offered by this tool. There are additional, more specific functions available both in the GUI and CLI modes, which are thoroughly documented in the project's source code, available in Markdown format.

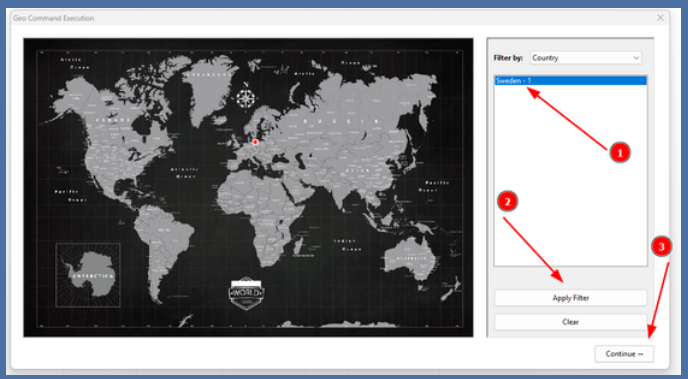
INTERFACE

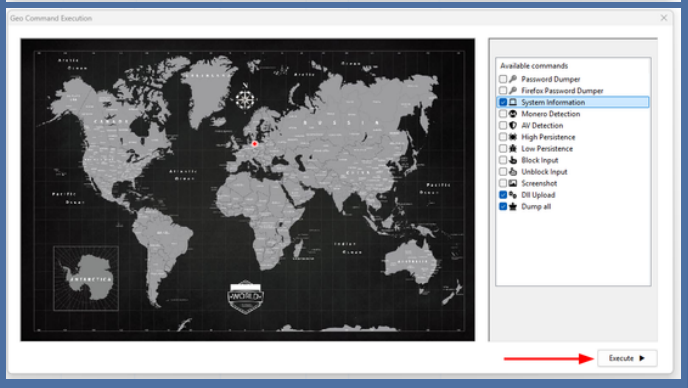
Nightmare Framework

Main Interface

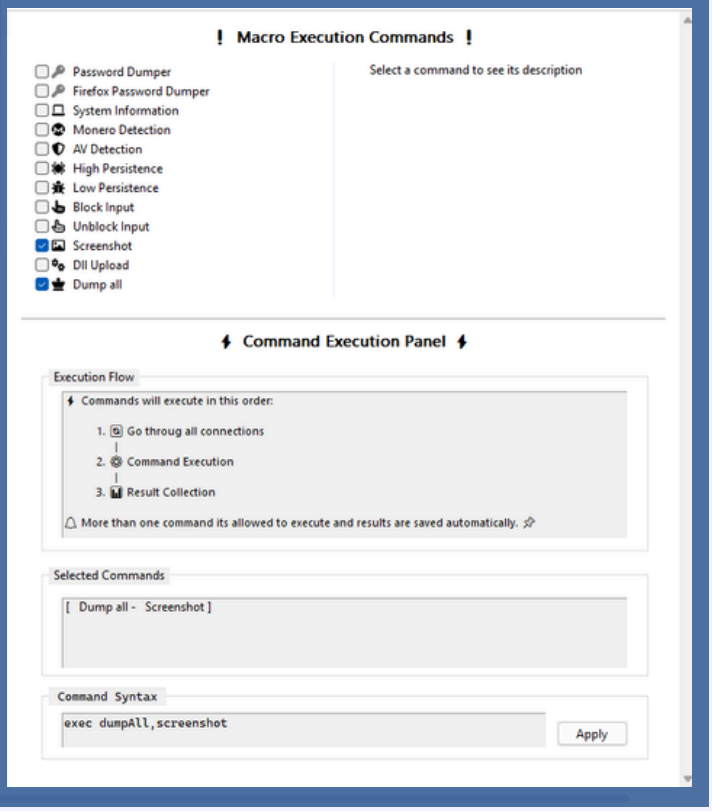


Geo Command Execution





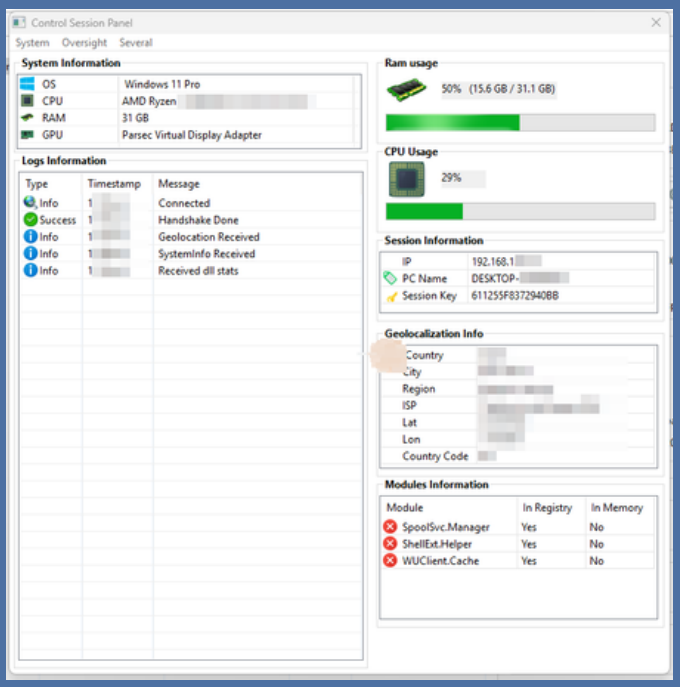
Global Execution



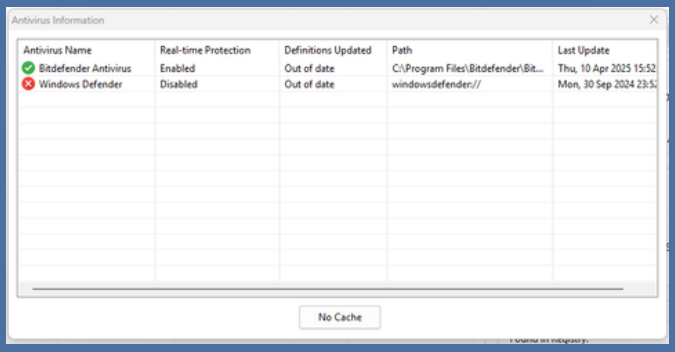
INTERFACE

Nightmare Framework

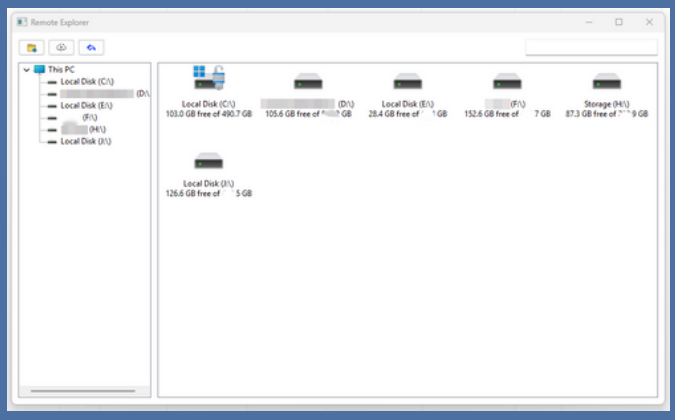
Session Control Panel



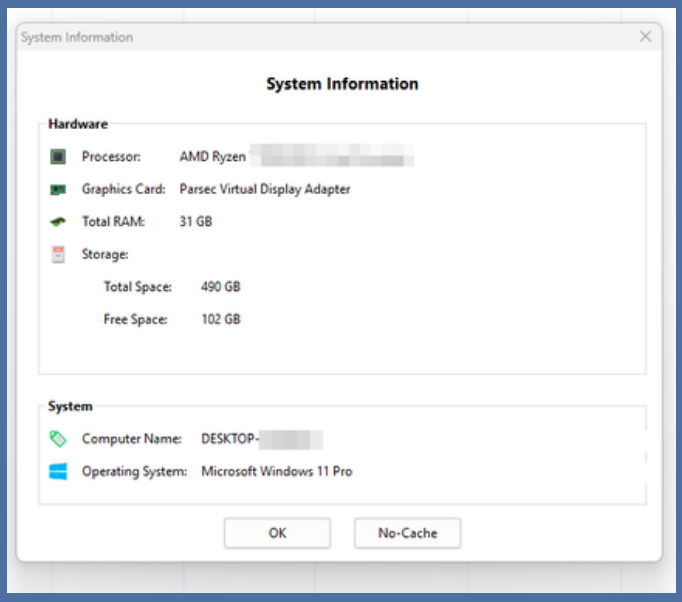
Anti-Virus Function



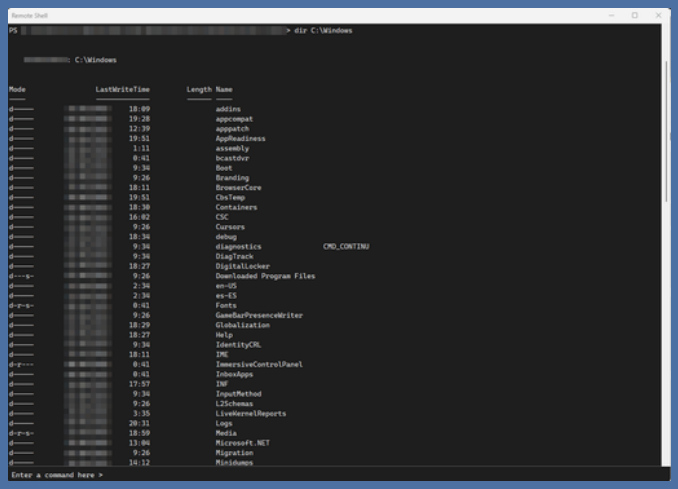
Remote Explorer



Remote Shell



Remote Shell



INTERFACE

Nightmare Framework

Process Dump

Processes Information

PID	Process Name
0	[System Process]
4	System
236	Secure System
280	Registry
1384	smss.exe
1952	csrss.exe
1764	wininit.exe
2052	csrss.exe
2116	services.exe
2136	Lsalso.exe
2148	lsass.exe
2216	winlogon.exe
2348	svchost.exe
2380	fontdrvhost.exe
2384	fontdrvhost.exe
2460	WUDFHost.exe
2512	svchost.exe
2560	svchost.exe
2624	dwm.exe
2876	svchost.exe
2884	svchost.exe
2892	svchost.exe
3028	svchost.exe
3036	svchost.exe
3044	svchost.exe
2168	svchost.exe
3088	svchost.exe
3180	b[illegible]
3200	b[illegible]
3208	svchost.exe
3400	svchost.exe
3456	svchost.exe
3656	svchost.exe
3664	svchost.exe
3756	svchost.exe
3792	svchost.exe
4092	svchost.exe
4104	svchost.exe

Software Dump

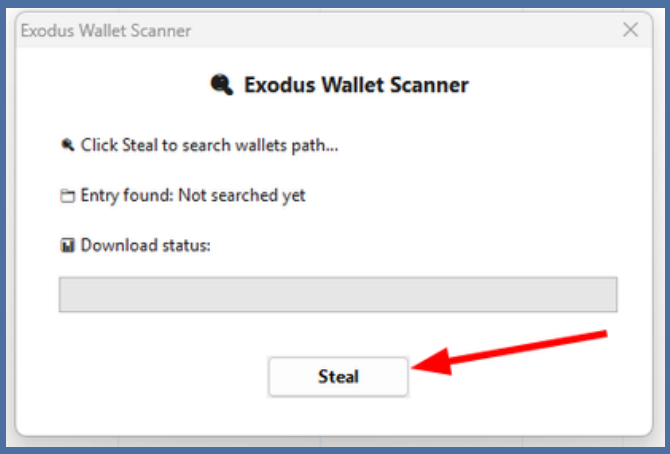
Software Information

Name	Version
010 Editor 15.0.1 (64-bit)	15.0.1
[illegible]	2025.5.0
[illegible]	24.09
[illegible]	24.10.34
[illegible]	27.0.40.173
[illegible]	27.1.1.12
[illegible]	0.2.3
[illegible]	9.5.0
Git	2.45.1
[illegible]	3.0.4.0
Mozilla Firefox (x64 es-ES)	137.0.1
Mozilla Maintenance Service	133.0
[illegible]	0.45.0.0
Parsec Virtual USB Adapter Driver	0.3.10.0
[illegible]	1.3.3
[illegible]	
[illegible]	
[illegible]	[illegible]
[illegible]	
[illegible]	
[illegible]	
[illegible]	
[illegible]	
[illegible]	
[illegible]	
[illegible]	[illegible]
[illegible]	
[illegible]	
[illegible]	
[illegible]	
[illegible]	
[illegible]	
VLC media player	3.0.21
[illegible]	8.1.25-0
NVIDIA Nsight Systems 2022.4.2	22.4.2.1
[illegible]	17.5.0
Microsoft Visual C++ 2013 x64 Additional Runti...	12.0.40664

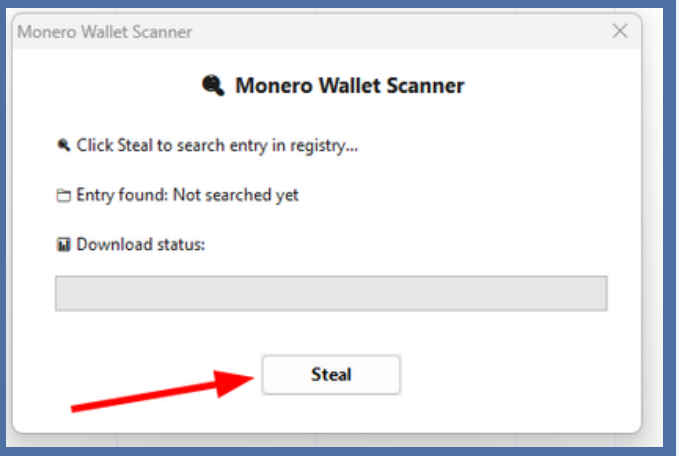
INTERFACE

Nightmare Framework

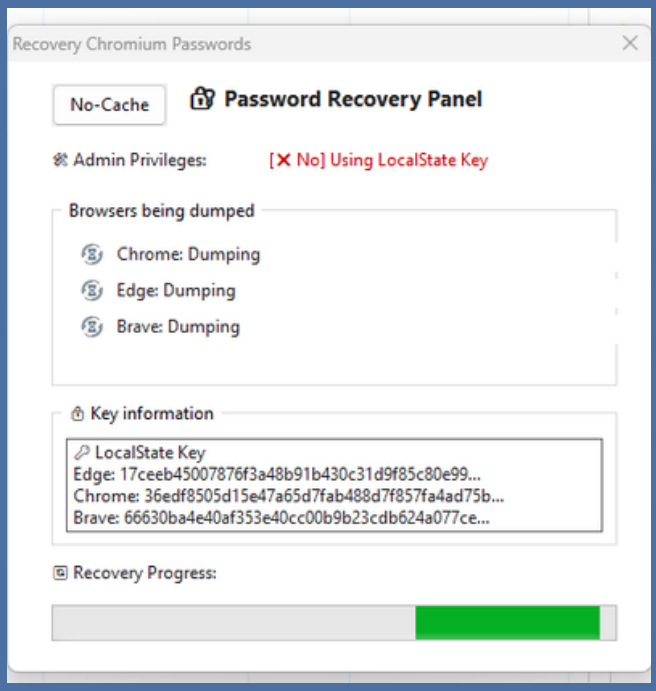
Exodus Wallet

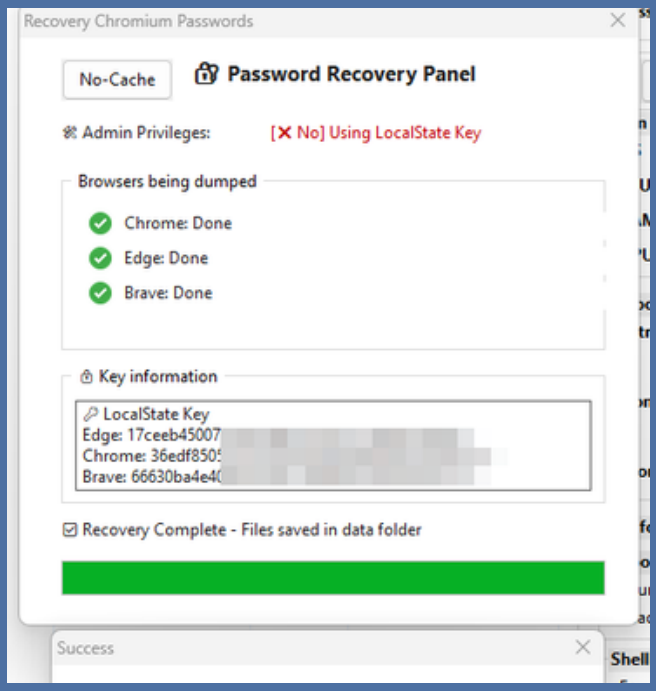


Monero Wallet



Password Dump

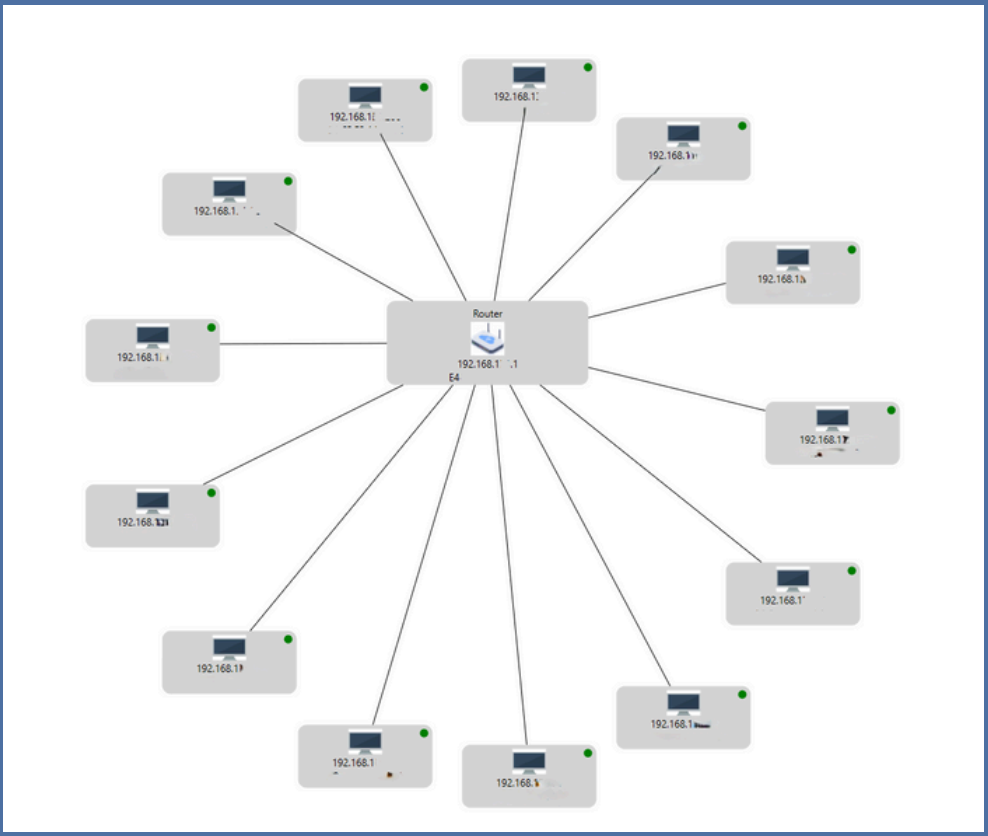




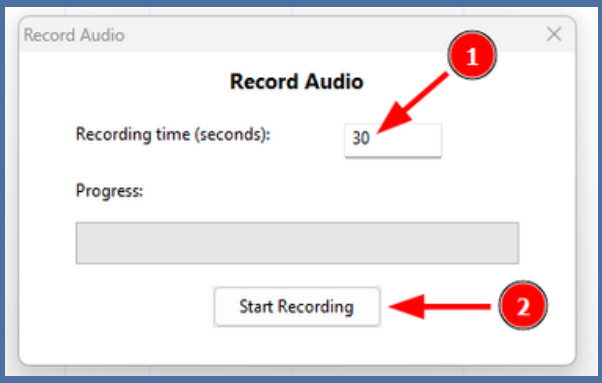
INTERFACE

Nightmare Framework

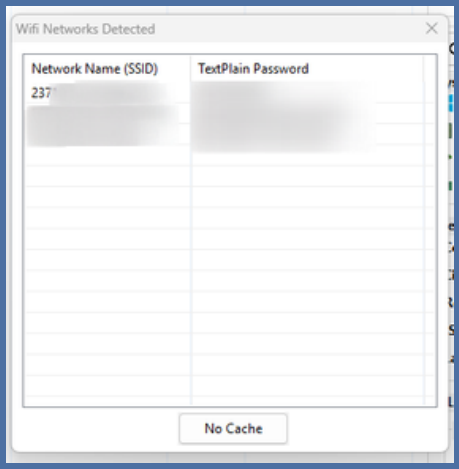
Network Map



Record Audio



Wifi Dump



LICENSE & USE

END USER LICENSE AGREEMENT (EULA)

This software is licensed, not sold. By using, copying, or modifying this software, you agree to the following terms:

1. **Authorized Use**: You are granted a license to use this software only in environments you own or where you have explicit written permission. Unauthorized use, including using this software for malicious activities, is strictly prohibited.
2. **Purpose**: This software is intended solely for educational and security research purposes within controlled and ethical environments, such as penetration testing or Red Team activities. It is not intended for illegal activities.
3. **Prohibited Use**: You may not use this software for:
   - **Unauthorized surveillance**
   - **Accessing systems or data without permission**
   - **Distributing malware or spyware**
   - **Engaging in illegal or unethical activities**
   - **Violating local, national, or international laws**
4. **Liability Disclaimer**: The developer of this software are not liable for any damages, legal consequences, or misuse resulting from the use of this software. By using this software, you accept full responsibility for its use and any associated risks.
5. **No Warranty**: This software is provided "AS IS" without warranties of any kind. The developers do not guarantee the software's performance, merchantability, or fitness for a particular purpose. Use it at your own risk.
6. **Redistribution and Modification**: You may not redistribute, sell, or modify the software without written permission from the developer. Any modifications are for personal or educational purposes only and cannot be distributed commercially.

By using this software, you confirm that you understand and agree to abide by these terms.

TERMS OF SALE

Availability of Full Code

The full source code is available upon request. Interested parties should contact us to obtain further details regarding the purchase process.

What the Purchase Includes

Upon purchasing the source code, the buyer will receive:

1. The **complete source code** of the project.
2. A **standalone GUI-based** compiler that **simplifies** the **setup** process, automatically **installing** all **required dependencies** (except the Visual Studio installer for C/C++ compilation, which must be installed manually).
3. **Full documentation**, including detailed **instructions** for compilation, setup, and usage of the software.

This package ensures that buyers have everything necessary to properly compile, configure, and operate the software with minimal technical barriers.

Post-Sale Support

Post-sale support is available for a limited **period of 30 days after purchase**. During this period, i will provide **technical assistance** in case the buyer encounters difficulties with installation, configuration, or operation of the software.

Additional support, updates, or any customizations of the software will be subject to an additional fee and will be agreed upon separately between the buyer and seller.

Additional Conditions

The code must not be used for illegal or malicious activities, as outlined in the EULA (End User License Agreement). Any misuse of the software, including unauthorized modification or redistribution, may result in the revocation of support and the software license.

PRICING

💵 Pricing

The complete package has a fixed price for standard (non-exclusive) purchases.

Negotiations are only available for companies or individuals requesting exclusive rights to the software.

📜 Sales Conditions

- A standard purchase grants a non-exclusive license to use the software under the terms outlined in the End-User License Agreement (EULA).
- If exclusive rights are requested, a separate negotiation will take place to determine the appropriate terms and pricing.
- **Important**:
  - If the code has already been sold previously in a non-exclusive manner, the buyer requesting exclusivity will be informed of the previous sale.
  - Should the buyer still wish to proceed, an exclusivity contract will be signed, and the software will be permanently withdrawn from sale thereafter.
- Once exclusivity is granted, no further copies will be offered, sold, or distributed under any circumstance.
- The exclusivity agreement will include clear clauses regarding:
  - Declaration that the code was sold with prior non-exclusive licenses (if applicable).
  - Confirmation that no future license sales, transfers, or distributions will be made.

🛡️ Additional Notes

- **Source Code Warranty**: The code is delivered exactly as described, without hidden modifications.
- **Transfer of Rights**: In exclusivity sales, the buyer acquires exclusive commercial use rights, while the intellectual property (authorship) remains with the original developer, unless otherwise agreed in writing.
- **Post-Sale Notice**: A public notice will be published confirming the exclusivity sale.

CONTACT INFO

📩 Contact Information

For inquiries, purchases, or further information, please use the following contact methods:

- 📧 **Email**: pablodiez024@proton.me
- 🌐 **GitHub**: https://github.com/an0mal1a
- 🌐 **LinkedIn**: https://www.linkedin.com/in/an0mal1a/
- ❗ **Important**:
  - Only serious, business-related inquiries will be answered.
  - All communications are treated confidentially and securely.
  - No unsolicited messages or unrelated proposals.
- **PGP Public Key**: (Optional, if you want encrypted communication, dont forget to add as file your public key)

-----BEGIN PGP PUBLIC KEY BLOCK-----
xjMEY859TxYJKwYBBAHaRw8BAQdAzPs2wj/f5Utap5yxzYjSrjdCFpEShgDB
XbWpULPUO/HNL3BhYmxvZGllejAyNEBwcm90b24ubWUgPHBhYmxvZGllejAy
NEBwcm90b24ubWU+wowEEBYKAD4FAmPOfU8ECwkHCAkQhOsaM7FUNFADFQgK
BBYAAgECGQECGwMCHgEWIQT7DfyqYfcts2wHL2uE6xozsVQ0UAAAQWoA/1K0
1yZvbMciaaMdVkECXRKCvkHRcZs//WQ+8S2koT5TAP0a91niXzBVEGFSdWO+
lioByU0EzVWdu7jZeLeMUYlnAc44BGPOfU8SCisGAQQBl1UBBQEBB0CpBT6a
6V2Z+NoKedCYWDjaWa5jqdaKHbLyOV/kOuQzBQMBCAfCeAQYFggAKgUCY859
TwkQhOsaM7FUNFACGwwWIQT7DfyqYfcts2wHL2uE6xozsVQ0UAAAJcwA/Rvr
/sp78c7/A9g2qlQlLxvjCF+s+IBCSqRGReB6wdcMAP4xli4Ahfp2w6C+hsxB
TI2/+Sdkxm0mPlnQ4xhe48/lCw==
=UqQY
-----END PGP PUBLIC KEY BLOCK-----