Adversary Emulation Framework

Mentor: Robert Y. Campbell

Ellicott City, Maryland

[robel.y.campbell@gmail.com](mailto:robel.y.campbell@gmail.com)

Mentor: Alcides Alvear

Gurabo, Puerto Rico

aalvear@uagm.edu

Student: Carlos A. Roque Fontanez Gurabo, Puerto Rico  
[croque16@email.uagm.edu](mailto:croque16@email.uagm.edu)

Student: Kiara Rivera DomenechGurabo, Puerto Rico  
[krivera786@email.uagm.edu](mailto:krivera786@email.uagm.edu)

Student: Genesis Resto Garcia  
Gurabo, Puerto Rico  
[gresto19@email.uagm.edu](mailto:gresto19@email.uagm.edu)

*Abstract*— The digital realm is constantly besieged by ever-changing malicious software that easily penetrates all protective barriers, operates maliciously without the user's awareness, and covertly extracts confidential information. Gaining insight into the workings of these harmful programs empowers us to counteract them more effectively [1].

Keywords— Adversary Emulation Framework, Malicious software, Malware attacks, Dynamic evasion, In-memory execution, Encrypted payloads, Implant, Syscalls Variables, Downloading Cradles, Event Tracing for Windows (ETW), Amsi Patching, In-memory execution, Implant Management

# Introduction

The Adversary Emulation Framework aims to address the constantly evolving nature of malware attacks by developing a framework that offers a structured and strategic approach to managing complex networked systems and offers various options to expose malware evasion techniques. The framework employs dynamic evasion, in-memory execution, and encrypted payloads to enhance its effectiveness and resilience. The implant uses various evasion tactics to avoid detection and collect victim information for a secure database. The future work includes creating a special tool called an implant that covers evasion techniques, collecting system information from the victim, and performing debugging to uncover new attack vectors for malware evasion. This framework will provide defense systems with better insight into the workings of harmful programs and empower them to counteract them more effectively. This is being explored the assembling program together with the Arduino development board. To inform us malware Attacks continue to evolve at an alarming rate, leaving defenders struggling to keep up with the latest techniques used by malicious actors. Antimalware systems are designed to detect these attacks, but their effectiveness is undermined by a lack of knowledge and understanding of lasting evasion techniques. Using behavior analysis these systems are constantly updated with the lasted signatures and techniques to stay ahead of the curve. It’s so important to know that defense systems are designed to detect and prevent malware attacks and defenders. Therefore, as a malware creator, it is essential to always be on the lookout for new techniques and tools to stay one step ahead of defense systems. We decided to create an innovative command and control framework to provide more effective dynamics in the complex world of malware and avoid detection. Using techniques incorporating a Dynamic Evasion, using ETW/AMSI Patching & Hardware Breakpoints, in memory execution using shellcode frameworks to create a spectacular and undetectable malware that can be penetrate even the most secure systems. As a malware creator, it is vital to find ways to create malicious code that is complex and varied enough to avoid detection by anti-malware defense systems. By exploring new techniques and tools, you always have the upper hand over defense systems and guarantee the success of your malware attacks.

# Related Work

In the realm of information security preparedness tools, Metta [16] stands out as a robust solution for enhancing readiness. Leveraging Redis/Celery, Python, and Vagrant with VirtualBox, Metta conducts adversarial simulations, allowing for comprehensive testing of host-based instrumentation. Depending on the Vagrant setup, it extends its assessment to network-based detection and controls. Metta efficiently processes actions specified in YAML files, utilizing Celery to organize and queue these actions for individual execution without necessitating user interaction.

MITRE's CASCADE [10], a research initiative, addresses the need for automating investigative tasks performed by a "blue team" to assess suspicious activities on a network using host data. The CASCADE server [15], detailed in the repository, excels in user authentication, analytics execution, and investigations. It employs Splunk/ElasticSearch-stored data for analytics and alert generation, triggering a recursive investigative process. This process explores related events, including parent-child processes, network connections, and file activity, generating a graph that visualizes event relationships. The server further enhances its capabilities by incorporating information from the Adversarial Tactics, Techniques & Common Knowledge (ATT&CK) project and allowing customization to minimize false positives. CASCADE's platform-agnostic query language facilitates simple analytics expression, automatically translated into Splunk or ElasticSearch queries, fostering adaptability and user-driven analytics creation.

A distinct approach is taken by APT Simulator [17], a Windows Batch script designed for simulating a compromised system. Unlike other adversary simulation tools, APT Simulator prioritizes simplicity. Users can seamlessly execute the Batch file as an Administrator after downloading and extracting the provided archive, without the need for additional servers, databases, or virtual machine agents.

In the planning problem domain of adversary emulation, Miller et al. [8] outlined strategies for handling uncertainties in target systems when utilizing CALDERA [24]. Their work contributes valuable insights into addressing uncertainties in the planning phase of adversary emulation.

# Techniques and components

## Revision Phase

* Command and Control Framework, employs dynamic evasion, in-memory execution, and encrypted payloads to enhance its effectiveness and resilience. The implant uses various evasion tactics to avoid detection and collect victim information for a secure database. This framework will provide defense systems with better insight into the workings of harmful programs and empower them to counteract them more effectively.
* Dynamic Evasion is a defense technique to our system capable of adapting to evading attacks in real time. This allows a faster and more effective response to constantly evolving threats.
* Event Tracing for Windows (ETW) and Anti-Malware Scan Interface (AMSI), these are two security mechanism systems built into Windows operating systems using a monitor to detect malicious activities. The ETW is an event logging tool that allows developers and systems administrators to analyze information about system activity in real- time. AMSI is an interface that allows anti-malware applications to scan content of scripts, macros and other files that can be used to carry out malicious attacks. These techniques refer to the modification of these security mechanisms to evade detection by security systems.

# Design

One of the limitations of this project in the real world could be to find effective evasion techniques for all operating systems at the same time. Most of the evasion techniques are for computers running Windows operating systems. Many of the techniques that are known today are assumed not to apply to Linux and this could present a limitation to this project. It is also complicated by the fact that software versions are always being updated and this allows that without realizing it we use obsolete software and may not work in the future. It is also important to keep in mind that the Arduino must be a protected one, because if it is confiscated it could create serious problems.

## Performance Criteria

* Detection Rate: How accurately the framework can identify and expose malware evasion techniques used by attackers.
* Evasion Coverage: The range of evasion techniques covered by the framework, such as dynamic evasion, in-memory execution, and encrypted payloads.
* Resilience: The ability of the framework to maintain effectiveness even in the face of advanced defense mechanisms employed by target systems.
* Data Collection: The framework's success in securely collecting and storing victim information, including system information and network traffic data.
* Attack Vector Discovery: The framework's ability to uncover new attack vectors and evasion techniques through debugging and analysis.
* Performance Overhead: The impact of the framework on the performance of target systems, striving for minimal resource utilization and system slowdowns.
* Integration and Compatibility: The framework's compatibility with existing security infrastructure and tools, ensuring smooth deployment and interoperability.
* Effectiveness of Countermeasures: The framework's ability to provide actionable insights and recommendations for enhancing defense systems against malware.
* Scalability: The framework's capability to handle large-scale networks and systems for emulation and analysis purposes.
* Continuous Improvement: The framework's future work, including regular updates to address emerging malware techniques and security challenges.

# System Integration plan

In our system implementation we have an Implant, Server, Interface & Database.

## Implant

#### Employ various evasion tactics to avoid detection.

#### Results of executed instructions received by the server must be sent back to the server.

#### Must have the ability to load new modules or code dynamically.

#### Implementing persistence to survive reboots.

#### Collect victim information for database.

## Server

* Send instructions to the implant and receive the output.
* Provide implant templates for customization.
* Store victim information in a database using secure methods.
* Manage Module Handler to host modules for execution on the implant.
* Provide secure and reliable communication channels.

## Interface

#### Provide operator ability to generate implants.

#### Dashboard to manage one or multiple implants.

#### Ensure secure authentication and authorization for operators.

#### Output the results from instructions executed on the implant.

#### Provide search and filtering capabilities to quickly locate specific information.

## Database

#### Storing and managing data related to victims, and operators securely.

#### Enforce consistency and validation to avoid errors and corruption.

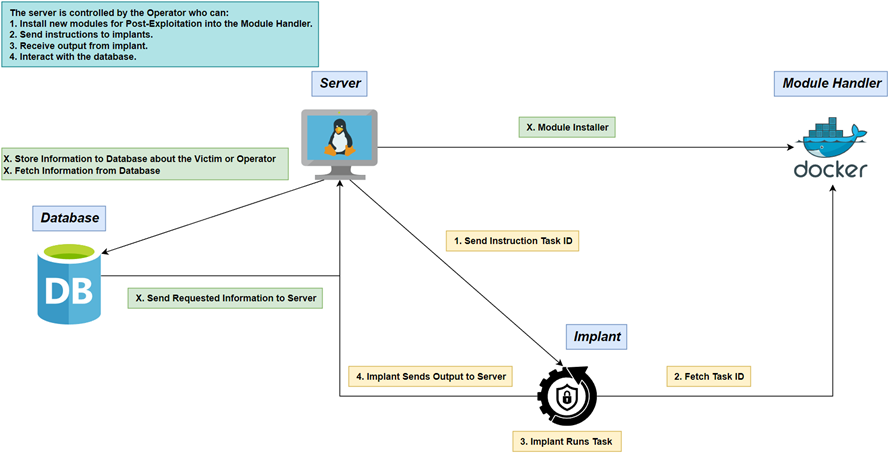


Fig.1. Concept Diagram

A computer screen shot of a computer

Description automatically generated

Fig.2. Database Scheme

Diagram

Description automatically generated

#### A screenshot of a computer Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Results

The system create a special tool called an implant. If the operator already has this tool, the system will take it in and manage it. If the operator doesn't have one, the system will make one, manage it, and get it ready to run. Then, it sets up a way for the tool and the server to talk to each other. If it is the first time the implant registers, it should collect system information from the victim. If it is not the first time, the implant should wait for the server to send commands before proceeding with any actions. If the server has tasks for the implant, it gets directions from a part called the Module Handler, using a TaskID as shown in Fig.4. The implant does the task and sends the results back to the server. This can be done as many times as the operator wants. If the server doesn't have any tasks, the implant just waits for new instructions.

## Implant

1. Initialization

1.1 Implant Identifier

The Main function initializes essential variables, setting the implantId to the host machine's name using Environment.MachineName. This unique identifier is crucial for subsequent interactions with the command-and-control server.

1.2 System Information Compilation

The function is called GenerateJson(), compiling detailed system information into a JSON format. This includes operating system, hardware details, and network information, crucial for registering the implant with the control server.

1.3 Registration Process

The RegisterImplant(victim\_information) method is invoked, sending the JSON string to the control server. The server responds with a unique token, ensuring secure communication and authentication for future interactions.

2. AMSI Bypass and ETW Patching

2.1 AMSI Bypass

The AmsiHBP.Start() call initiates AMSI bypass, allowing the implant to execute its payload without detection from antivirus software.

2.2 ETW Patching

The EtwPatch.Start() function patches ETW, preventing the implant's activities from being logged and tracked by system monitoring tools.

3. Main Command Execution Loop

3.1 Core Loop

An infinite loop continuously checks for new commands from the command-and-control server, ensuring real-time interaction and responsiveness to incoming instructions.

3.2 Command Object

A new Command object is instantiated in each iteration, storing details of the command received from the C&C server.

3.3 Exception Handling

A try-catch block ensures graceful handling of exceptions or errors during command retrieval or execution, maintaining the implant's operational status.

4. Command Retrieval

4.1. Retrieval Mechanism

Commands are retrieved from the C&C server using either HTTP or SMB protocols, depending on system configuration.

4.2. JSON Response

The implant constructs a URL or uses SMBClient to retrieve a JSON response containing command details, parsed to extract relevant information.

5. Command Parsing and Execution

5.1. Parsing JSON Response

The implant parses the JSON response, extracting and interpreting command specifics such as Command, Args, and ImplantUser.

5.2. Command Handling

A switch statement handles different command types, allowing modular and organized execution of diverse actions on the victim's system.

5.3. Result Communication

Results of command execution are sent back to the C&C server using either SMB or HTTP, depending on the initial command retrieval method.

5.4. Error Handling

Comprehensive error handling within try-catch blocks ensures stable behavior, preventing crashes or unpredictable actions in case of issues during command execution.

## Server

The server functions as a sophisticated command and control center, orchestrating a covert implant system with precision and versatility. Users can seamlessly send instructions to the implant, a discreet and remotely controlled entity, and receive real-time outputs, facilitating seamless interaction with the target environment. To enhance adaptability, the server offers customizable implant templates, allowing users to tailor the implant's functionalities to specific requirements. The system prioritizes security, employing robust methods to store victim information in a protected database, ensuring the confidentiality and integrity of sensitive data. Furthermore, the server boasts a sophisticated Module Handler, serving as a centralized hub to host an array of modules for execution on the implant. This modular approach streamlines the management of diverse functionalities, enabling users to deploy tailored tools as needed, enhancing the overall effectiveness and stealth of the implant system.

## Interface

## As shown in Fig. 5, in accordance with the specified requirements, the developed interface exhibits a comprehensive set of features catering to the operational needs of operators in the context of implant generation and management. The system offers operators the capability to seamlessly generate implants through an intuitive interface. A centralized dashboard facilitates efficient management of one or multiple implants, affording operators a streamlined overview of the entire implant ecosystem. Security is a paramount consideration, with robust authentication and authorization mechanisms implemented to ensure that only authorized personnel can access and manipulate the system. Furthermore, the interface provides a mechanism to output the results derived from instructions executed on the implants, enhancing transparency and facilitating post-execution analysis. To enhance usability, the system incorporates sophisticated search and filtering functionalities, empowering operators to swiftly and accurately locate specific information within the interface.

## Database

#### As shown in Fig. 2, a robust and secure database system plays a pivotal role in the storage and management of critical data related to both victims and operators within a given context. By implementing stringent security measures, such as encryption and access controls, databases ensure the confidentiality and integrity of sensitive information, safeguarding it against unauthorized access or potential breaches. Moreover, databases contribute significantly to the maintenance of data consistency and accuracy through the enforcement of validation rules. These rules help prevent errors and corruption by ensuring that only valid and properly formatted data is stored. In doing so, databases become a reliable foundation for organizations to build upon, fostering trust in the accuracy and reliability of the data essential for supporting victims and facilitating efficient and effective operations.

# future work

In the realm of cybersecurity, the future development of evasion frameworks demands a multifaceted approach to ensure adaptability and effectiveness against ever-evolving threats. Firstly, there is a pressing need to broaden the framework's scope by expanding its compatibility to include evasion techniques tailored for diverse operating systems. This expansion enhances the framework's versatility, making it applicable across a broader range of systems and thereby increasing its overall utility. Secondly, a continuous commitment to research is imperative to keep pace with the dynamic landscape of malware tactics. By proactively incorporating the latest evasion techniques into the framework, it can maintain its efficacy in thwarting emerging threats. Lastly, the integration of machine learning algorithms represents a cutting-edge avenue for future development. By infusing the framework with the ability to adapt and learn from new attack patterns, it can significantly enhance its capability to identify and counteract novel threats more efficiently. Embracing these three key directions—platform diversification, ongoing research integration, and machine learning augmentation—will fortify the framework's resilience and efficacy in the face of an ever-changing cybersecurity landscape.

# References

[1] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, “A Survey on Malware Detection Using Data Mining Techniques,” ACM Computing Surveys, vol. 50, no. 3, pp. 1–40, Jun. 2017, doi: <https://doi.org/10.1145/3073559>.

‌[2] R. Tahir, “A Study on Malware and Malware Detection Techniques,” International Journal of Education and Management Engineering, vol. 8, no. 2, pp. 20–30, Mar. 2018, doi: <https://doi.org/10.5815/ijeme.2018.02.03>.

[3] D. Patten, “The Evolution to Fileless Malware 1 The Evolution to Fileless Malware.” Accessed: Mar. 23, 2022. [Online]. Available: <http://infosecwriters.com/Papers/DPatten_Fileless.pdf>

[4] E. Chaffey, D. Sgandurra, and R. Holloway, “Malware vs Anti-Malware Battle -Gotta Evade 'em All!,” 2020. Accessed: Apr. 29, 2023. [Online]. Available: <https://core.ac.uk/download/pdf/340199834.pdf>

‌[5]A. Kapellas, “A Thesis in Malware Development.” Accessed: Mar. 29, 2022. [Online]. Available: <https://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/11190/Kapellas_mte1604.pdf?sequence=2&isAllowed=y>

[6] Georgios Ioannou, “Malware Development for Red Teaming Using Metasploit Petros Katritzidakis,” 2018. Accessed: Apr. 29, 2023. [Online]. Available: <https://repository.ihu.edu.gr/xmlui/bitstream/handle/11544/29159/Dissertation_Thesis_Petros_Katritzidakis_Malware%20Development%20for%20Red%20Teaming%20Using%20Metasploit.pdf?sequence=1>

[7] X. Hu, “Large-Scale Malware Analysis, Detection, and Signature Generation.” Accessed: Apr. 30, 2023. [Online]. Available: <https://deepblue.lib.umich.edu/bitstream/handle/2027.42/89760/huxin_1.pdf%3Fsequence%3D1%26isAllowed%3Dy>

[8] D. Miller, R. Alford, A. Applebaum, H. Foster, C. Little, and B. Strom, “Automated Adversary Emulation: A Case for Planning and Acting with Unknowns.” Accessed: Apr. 30, 2023. [Online]. Available: <https://apps.dtic.mil/sti/pdfs/AD1108001.pdf>

[9] A. B. Ajmal, M. A. Shah, C. Maple, M. N. Asghar, and S. U. Islam, “Offensive Security: Towards Proactive Threat Hunting via Adversary Emulation,” IEEE Access, vol. 9, pp. 126023–126033, 2021, doi: <https://doi.org/10.1109/access.2021.3104260>.

[10] B. Strom, A. Applebaum, D. Miller, K. Nickels, A. Pennington, and C. Thomas, “MITRE ATT&CK®: Design and Philosophy.” Available: <https://www.mitre.org/sites/default/files/2021-11/prs-19-01075-28-mitre-attack-design-and-philosophy.pdf>

‌[11]M. Carvalho et al., “Command and Control Requirements for Moving-Target Defense,” IEEE Intelligent Systems, vol. 27, no. 3, pp. 79–85, May 2012, doi: <https://doi.org/10.1109/mis.2012.45>.

[12] B. Strom, A. Applebaum, D. Miller, K. Nickels, A. Pennington, and C. Thomas, “MITRE ATT&CK®: Design and Philosophy.” Available: <https://www.mitre.org/sites/default/files/2021-11/prs-19-01075-28-mitre-attack-design-and-philosophy.pdf>

[13] TylerMSFT, “System Calls,” learn.microsoft.com. <https://learn.microsoft.com/en-us/cpp/c-runtime-library/system-calls?view=msvc-170> (accessed Apr. 30, 2023).

[14] adegeo, “Dynamic Language Runtime Overview - .NET Framework,” learn.microsoft.com. <https://learn.microsoft.com/en-us/dotnet/framework/reflection-and-codedom/dynamic-language-runtime-overview>

[15] Cascade GitHub. Available online: <https://github.com/mitre/cascade-server>.

[16] Metta GitHub. Available online: <https://github.com/uber-common/metta>.

[17] APT Simulator GitHub. Available online: <https://github.com/NextronSystems/APTSimulator>.

[18] Applebaum, A.; Miller, D.; Strom, B.; Korban, C.; Wolf, R. Intelligent, automated red team emulation. In Proceedings of the 32nd Annual Conference on Computer [19] Security Applications, Los Angeles, CA, USA, 5–9 December 2016; pp. 363–373. [Google Scholar]

[20] Miller, D.; Alford, R.; Applebaum, A.; Foster, H.; Little, C.; Strom, B. Automated Adversary Emulation: A Case for Planning and Acting with Unknowns; MITRE: McLean, VA, USA, 2018. [Google Scholar]

[21] Kavak, H.; Padilla, J.J.; Vernon-Bido, D. A characterization of cybersecurity simulation scenarios. In Proceedings of the 2016 Spring Simulation Conference (CNS), Pasadena, CA, USA, 3–6 April 2016. [Google Scholar]

[22] Kuhl, M.E.; Sudit, M.; Kistner, J.; Costantini, K. Cyber attack modeling and simulation for network security analysis. In Proceedings of the 2007 Winter Simulation Conference, Washington, DC, USA, 9–12 December 2007; pp. 1180–1188. [Google Scholar]

[23] Applebaum, A.; Miller, D.; Strom, B.; Foster, H.; Thomas, C. Analysis of automated adversary emulation techniques. In Proceedings of the Summer Simulation Multi-Conference, Bellevue, WA, USA, 9–12 July 2017; pp. 1–12. [Google Scholar]

[24] Caldera. Available online: [**https://www.mitre.org/research/technology-transfer/open-source-software/caldera**](https://www.mitre.org/research/technology-transfer/open-source-software/caldera)