Adversary Emulation Framework

CPEN 491 Computer Engineering Design Concepts

Mid-Term Report

Submitted by

Genesis Resto

Kiara Rivera

Carlos Roque

Date May 15, 2023

Electrical and Computer Engineering Program

Universidad Ana G. Méndez

Gurabo, PR 00778

EXECUTIVE SUMMARY

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.

Table of Contents

Acknowledgments 6

Copyright 6

1. Problem Definition 7

1.1. Users and Stakeholders 7

1.2. Problem Description 7

1.3. Project Objectives 7

1.4. Project Significance 7

1.5. Current State of the Art 7

2. Design Constraints and Requirements 8

2.1. Design Constraints 8

2.2. Performance Criteria 8

2.3. Engineering Standards 8

2.4. Other Requirements 8

3. Concept Generation 9

3.1. Basic Working Principles 9

3.2. Alternative Design Solutions 9

3.3. Evaluation of Design Alternatives 9

3.4. Final Design Concept 9

4. Design Description 10

4.1. Sub-system XXX 10

4.2. Sub-system YYY 10

5. System Integration 11

5.1. System Integration Plan 11

5.2. System-Level Assembly 11

6. Simulation and Testing 12

6.1. System Simulation 12

6.2. Test Plan 12

6.3. Performance Assessment 12

7. Prototype 13

7.1. Parts List 13

7.2. Prototype Demonstration 13

8. Project Management 14

8.1. Project Timeline 14

8.2. Budget Statement 14

8.3. Individual Contributions 14

9. Conclusions 15

9.1. Concluding Remarks 15

9.2. Recommendations for Future Work 15

10. Intellectual Property 16

11. References 17

12. Appendices 18

List of Figures

Figure 1: SEM image of a particulate composite. 22

List of Tables

Table 1: Predicted Student Grade Distribution 22

Acknowledgments

We would like to express our gratitude to the following individuals and organizations for their support in the completion of this Capstone Project:

* Dr. Alcides Alvear, we would like to express sincere gratitude to all those who supported and contributed to this project. For his invaluable guidance, patience, and feedback throughout the project.
* Robel Cambell, our project mentor, for his support and contributions to the project.

We also extend our thanks to all those who contributed to this project in various ways, but whose names may not be listed here.

Copyright

“We the team members,

Kiara Rivera Domenech Genesis Resto García Carlos Roque Fontánez

Member 1 Member 2 Member 3

hereby assign our copyright of this report and its corresponding Executive Summary to the Electrical and Computer Engineering (ECE) Program of Universidad Ana G. Méndez. We also hereby agree that the video of our Oral Presentations is the full property of the ECE Program.

Publication of this report does not constitute approval by Universidad Ana G. Méndez, the ECE Program or its faculty members of the findings or conclusions contained herein. It is published for the sole purpose of exchanging and stimulating ideas.

|  |
| --- |
| **Authors’ Ethics Statement:**  The work submitted in this project was solely prepared by KIARA RIVERA DOMENECH, GENESIS RESTO GARCIA CARLOS A. ROQUE FONTANEZ and it is original. Excerpts from others’ work have been clearly identified and listed in the list of references. All the engineering drawings, computer programs, formulations, and related files submitted on the accompanying materials are also original and prepared by  KIARA RIVERA DOMENCH, GENESIS RESTO GARCIA & CARLOS A. ROQUE FONTANEZ.  (Kiara Rivera Domenech, Genesis Resto García y Carlos Roque Fontánez) |

1. Problem Definition
   1. Users and Stakeholders

The user base of this system encompasses all individuals who can access the internet via any device. Additionally, it caters to those interested in understanding network security operations. A system is considered 'hacked' when an unauthorized entity has intruded upon it, potentially gaining access to confidential information, disrupting processes, or causing system damage. In such a scenario, both system users and stakeholders might be adversely affected due to potential security breaches and operational disruptions, potentially leading to the exposure of personal or sensitive information.

* 1. Problem Description

The internet, being an integral part of everyday life, opens an enormous wealth of information and resources. However, the same ubiquity also exposes individuals and organizations to varied forms of cyber-attacks. To combat these cyber threats, defensive systems have been devised to identify and thwart malicious activities.

These protective systems employ either behavioral or static analysis to constantly monitor and upgrade their defenses against emerging malware techniques. This includes recognizing signatures for payloads in Metasploit's MSFvenom and Cobalt Strike's beacons. Nonetheless, the evolving nature of malware attacks implies the perpetual risk of newer, more advanced techniques being introduced.

* 1. Project Objectives

One principal goal is to ensure the client's malware detection system does not intercept their simulated attack. This approach offers a realistic portrayal of the potential threats and vulnerabilities an actual attacker might exploit. To attain this goal, red teamers must persistently observe and modify their tactics and strategies to keep pace with the evolving landscape of malware and exploitation techniques. Additionally, they need to pinpoint vulnerabilities in an organization's security systems that actual attackers can exploit. Such insights can then be leveraged to design and execute effective countermeasures, bolstering the organization's overall security.

* 1. Project Significance

Protection of personal data: With the internet being an expansive and intricate network, personal data can be compromised unless appropriate measures are taken. Understanding safe internet navigation can help secure personal data, including credit card details, social security numbers, and other sensitive information, preventing unauthorized access or misuse.

Prevention of scams and fraud: The internet is a fertile ground for various scams and fraudulent activities. Internet navigation knowledge can assist individuals in avoiding such traps and safeguarding their financial assets.

Privacy upkeep: The internet serves as a robust platform for communication and self-expression but can also expose personal data to the public. Understanding how to navigate safely on the internet can help individuals maintain their privacy and control the information they share with others.

Defending against malware and other cyber threats: The internet is a frequent source of malware and other cyber threats. Safe internet navigation knowledge can help individuals protect their computer systems and networks from these threats, thus reducing the likelihood of falling prey to a cyber-attack.

* 1. Current State of the Art

***1.5.1 Code Execution Methods***

Currently, Command and Control frameworks perform their code execution using Injection Techniques. This is largely since implants can be boiled down to shellcode instructions to be inserted into memory. Although this opens paths for initial access into computer networks, the operator needs a very deep knowledge of the internals of operating systems and assembly instructions to create shellcode. Moreover, the advantage of this approach is that the implant code resides inside the computer memory. Hence evading on-disk detection mechanisms.

***1.5.2 Evasion Techniques***

As of 2023, the most prevalent evasion techniques that exists is Reflective DLL Injection. Using the Shellcode Reflective DLL Injection (sRDI) project [15], actors can convert compiled DLLs into position-independent shellcode for injection or loading mechanisms. A combination of this, coupled with obfuscation, shellcode encryption and detection services’ patching, the implant can pass through defenses and achieve execution.

***1.5.3 New Developments***

In our framework, we take a different approach to loading code into memory. First, we use download cradles to download scripts from the internet. This download cradle is for the Microcontroller to get the implant code, coupled with Dynamic Language Runtime [14] we can load the C# implant which is encrypted, into memory and execute it. Now, this approach can also be expanded into compiled bytes of .NET Assemblies, and this lets us use any Dynamic Programming Language that can accept the .NET Framework for compilation or inline execution such as plain-text IronPython code.

***1.5.3 Evasion Vectors***

Now, since we’ll also be implementing evasion just before the actual .NET code is loaded, we need to consider each detection mechanism step by step. Our approach was to encrypt the plain-text C# implant using AES-256 to avoid AMSI from scanning it and rendering it malicious. Next, patch ETW by setting up the correct bytes and writing them into the buffer of the loaded process where ETW exists and writing the bytes to immediately return the EtwEventWrite function. Next, we also patch AMSI by sending the correct bytes to AmsiScanBuffer just as we did with ETW and return a clean scan before the actual scan. When decrypting the implant code and loading into memory using a C# runspace, we check if there are commands on the server, if not (dormant state) we encrypted some of the heap sections in the implant buffer to evade memory scanners. When the server has a command for an implant, decrypt that heap buffer and execute the command. To which encrypts it again to lay dormant.

1. Design Constraints and Requirements
   1. Design Constraints

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. The vast majority 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.

* 1. 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.
  1. Engineering Standards

This study aims to validate its findings, propositions, and theories, adhering strictly to all the regulations applicable to engineering research papers, in accordance with IEEE citation guidelines. The evidence for the study, including work examples, will be presented in appendices, and all relevant references will be properly cited. Furthermore, the annex will comprise various resources and instruments essential for crafting the study and its substantiating materials, examples, and suggestions.

1. Concept Generation
   1. Basic Working Principles

The Basic working of principle of adversary Emulation Framework is a Command-and-control Framework that involves. Is based on a hierarchical structure that allows a central command entity to direct and control the activities of subordinate units or individuals. This structure is designed to enable rapid decision-making and coordination in complex and rapidly changing situations, such as military operations, emergency response situations, or crisis management scenarios. We work with Dynamic *Evasion – using Variable Syscalls, ETW/AMSI Patching, In-memory Execution – Download Cradles, Microcontroller and Encrypted Payloads – AES-256, XOR* as an alternative solution.

* 1. Alternative Design Solutions

We Consider at least two possible design solutions for solve our problem:

* A Command-and-Control Framework that include the following techniques:

**Dynamic evasion** is a defense technique where a system or network is capable of adapting and evading attacks in real-time, allowing for a faster and more effective response to constantly evolving threats.

**Variable syscalls** is a technique used to avoid detection and exploitation of vulnerabilities in operating systems by dynamically altering the available system calls in a program or system.

**Event Tracing for Windows (ETW) and AMSI Patching** are two security mechanisms built into Windows operating systems used to monitor and detect malicious activities. ETW is an event logging tool that allows developers and system administrators to capture and analyze detailed information about system activity in real-time. On the other hand, *AMS*I is an interface that allows anti-malware applications to scan the content of scripts, macros, and other files that can be used to carry out malicious attacks.

**In-memory execution** is another technique used by attackers to evade detection by traditional security systems, by executing malicious code directly in the system's memory.

**Download cradles** are a cybersecurity technique used by attackers to evade detection by traditional security solutions and download malware onto a compromised system. This technique involves the use of malicious scripts or commands to download malware onto the system via legitimate servers or compromised websites.

**A microcontroller** is a device that combines a microprocessor, memory, and input/output peripherals on a single chip.

* Artificial Intelligence can detect different types of malwares attacks using the following techniques:

**Detection of High Entropy:** High entropy in malware suggests that the code is obfuscated or encrypted, making it difficult to detect. By training AI algorithms on a large dataset of known malware samples, the AI model can learn patterns and characteristics associated with high entropy. This trained model can then analyze files or network traffic in real-time and flag instances with high entropy as potential malware.

**Region Manipulation Detection:** Region manipulation is when malware hides or modifies itself within specific parts of a file or memory. AI-based algorithms can be developed to analyze file structures, memory snapshots, or binary code and identify any irregularities or anomalies that indicate region manipulation. To do this, the AI model can be trained on normal file or memory structures to recognize deviations from the expected patterns.

**Detection of Code Injection:** Code injection occurs when malware inserts its malicious code into legitimate processes or applications. AI can be helpful in detecting code injection by learning the normal behavior patterns of processes and identifying any deviations caused by injected code. An AI algorithm, such as an anomaly detection technique, can learn what is typical for a process and flag any unusual code execution or system activity.

* 1. Evaluation of Design Alternatives

**Command and Control Framework**

**Advantages:**

* + Centralized Control: Command and control (C2) frameworks provide a way to centrally manage, and control compromised devices, making it easier for administrators to coordinate and streamline operations.
  + Flexibility: C2 frameworks allow administrators to issue commands, gather information, and deploy updates or patches to multiple compromised devices at once, offering flexibility in managing these systems.
  + Covert Communication: C2 frameworks use hidden communication channels, making it difficult for security systems to detect or block communication between compromised devices and the command server. This covert communication helps maintain persistence and evade detection.
  + Scalability: C2 frameworks can handle many compromised devices, making them suitable for managing extensive botnets or distributed systems.
  + Malware Evolution: C2 frameworks enable developers to quickly respond to countermeasures and enhance the capabilities of malware by providing a centralized platform for managing and updating the malware.

**Disadvantages:**

* Detection by Security Systems: Advanced security systems can detect C2 communications by monitoring network traffic and behavior. Intrusion detection systems and prevention systems can identify and mitigate compromised devices.
* Vulnerability to Takedowns: C2 frameworks are vulnerable to takedowns by law enforcement agencies, security researchers, or others. If the infrastructure supporting the C2 framework is discovered and disrupted, it can impact the operations of compromised devices.
* Single Point of Failure: C2 frameworks rely on a central command server, which can be a single point of failure. If the command server is compromised or taken down, it disrupts control and communication with compromised devices, making them ineffective.
* Legal and Ethical Implications: Using C2 frameworks for malicious purposes, such as launching cyberattacks or unauthorized surveillance, is illegal and unethical. Engaging in such activities can lead to legal consequences and damage a person's or organization's reputation.
* Security Risk: C2 frameworks introduce security risks as they involve compromising and controlling systems. Compromised devices can be used for further attacks, compromising sensitive data, and causing harm to individuals or organizations.

**Artificial Intelligence**

**Advantages:**

* Better Detection: AI algorithms can analyze a lot of data, find patterns, and detect unusual behavior more effectively than traditional methods.
* Adaptability to New Threats: AI models can learn from new threats and identify attacks that were not previously known.
* Real-Time Analysis: AI algorithms can quickly analyze files, network traffic, and system behavior to identify potential threats in real-time.
* Automation and Efficiency: AI can automate tasks like analyzing malware, detecting anomalies, and monitoring systems, which can save time and resources.

**Disadvantages:**

* Complexity: AI technologies can be complex to understand and implement, requiring specialized knowledge and skills.
* Data Limitations: AI models need large and diverse datasets to be trained effectively, which may be difficult to obtain, especially for new or uncommon threats.
* False Positives/Negatives: AI systems may incorrectly flag harmless activities as threats (false positives) or fail to detect actual threats (false negatives), which can affect the accuracy of the system.
* Resource Requirements: Implementing AI in cybersecurity may require significant computational power and resources, which can be challenging for organizations with limited budgets or infrastructure.
* Ethical Considerations: The use of AI raises ethical concerns related to privacy, bias, and the potential for unintended consequences. It's important to consider and address these ethical issues when implementing AI in cybersecurity.
  1. Final Design Concept

We choose the Command-and-Control Framework mainly for these reasons:

* Centralized management: A C2 framework provides a centralized platform to manage and control a set of compromised devices. This facilitates the coordination and execution of actions across all devices in an efficient manner.
* Flexibility: C2 frameworks offer flexibility in managing compromised systems. They allow administrators to issue commands, gather information and deploy updates or patches across the network of compromised devices simultaneously.
* Covert communication: C2 frameworks often use covert communication channels, making it difficult for security systems to detect or block. This covert communication helps maintain persistence and evade detection by traditional security solutions.
* Scalability: C2 frameworks are designed to be scalable and can handle many compromised devices. This makes them suitable for managing large botnets or distributed systems.
* Malware evolution: C2 frameworks can facilitate malware evolution and adaptation. By providing a centralized platform for managing and updating malware, developers can quickly respond to countermeasures and enhance malware capabilities.

1. Design Description
   1. Dynamic Evasion

Defense technique where a system or network is capable of adapting and evading attacks in real-time, allowing for a faster and more effective response to constantly evolving threats.

* 1. Variable Syscalls

Technique used to avoid detection and exploitation of vulnerabilities in operating systems by dynamically altering the available system calls in a program or system.

* 1. Event Tracing for Windows (ETW) and AMSI Patching

The ETW/AMSI Patching technique refers to the modification of these security mechanisms to evade detection by security systems.

* 1. In-memory Execution

Technique used by attackers to evade detection by traditional security systems, by executing malicious code directly in the system's memory.

* 1. Download Cradles

Technique used by attackers to evade detection by traditional security solutions and download malware onto a compromised system. This technique involves the use of malicious scripts or commands to download malware onto the system via legitimate servers or compromised websites.

* 1. Microcontroller

Device that combines a microprocessor, memory, and input/output peripherals on a single chip.

1. System Integration
   1. 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.
  1. System-Level Assembly

A screenshot of a computer screen

Description automatically generated with low confidence

Figure 1. Wireframe user login

A screenshot of a computer

Description automatically generated

Figure 2. Wireframe dashboard

1. Simulation and Testing
   1. Diagram

      Description automatically generatedSystem Simulation

Figure 1. Projected Completion Model

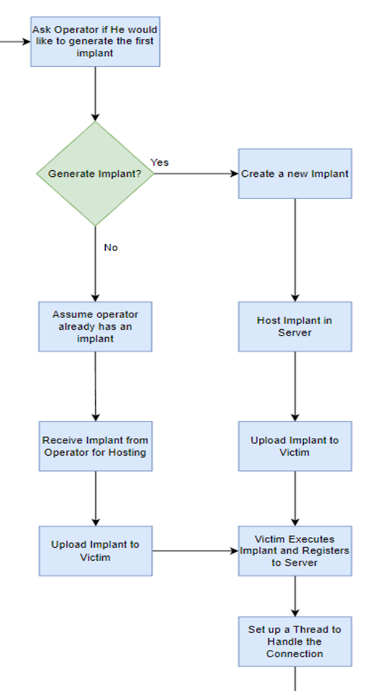
* 1. Test Plan

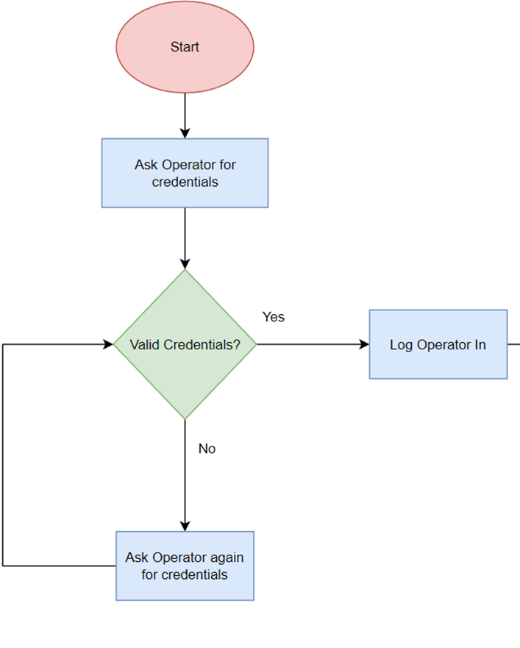
The system should be built to create a special tool called an implant. If the operator already has this tool, the system will take it in and manage it. If the operator doesn't have one, the system will make one, manage it, and get it ready to run. Then, it sets up a way for the tool and the server to 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. 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.

* 1. Performance Assessment

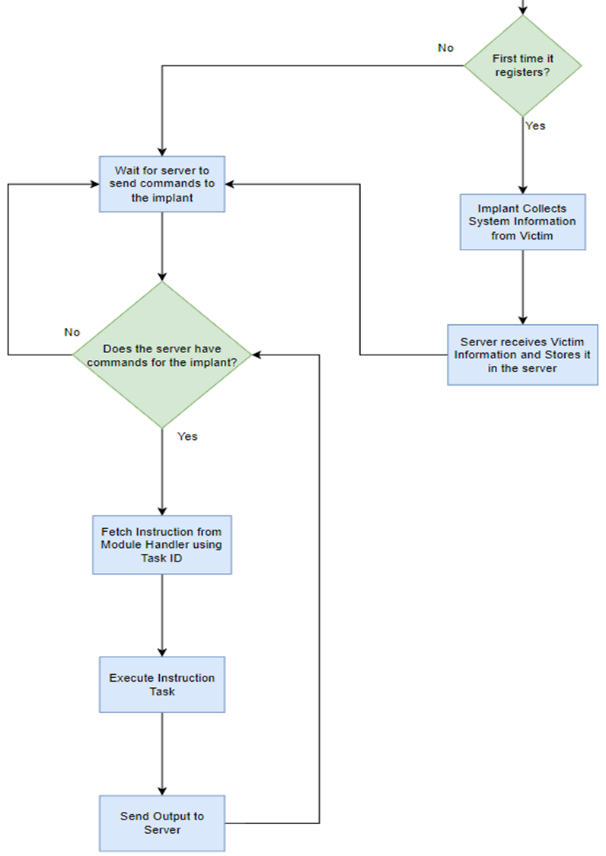
After an examination of each of the components of the command-and-control framework, it meets all the required performance criteria for the design.

1. Prototype
   1. Parts List

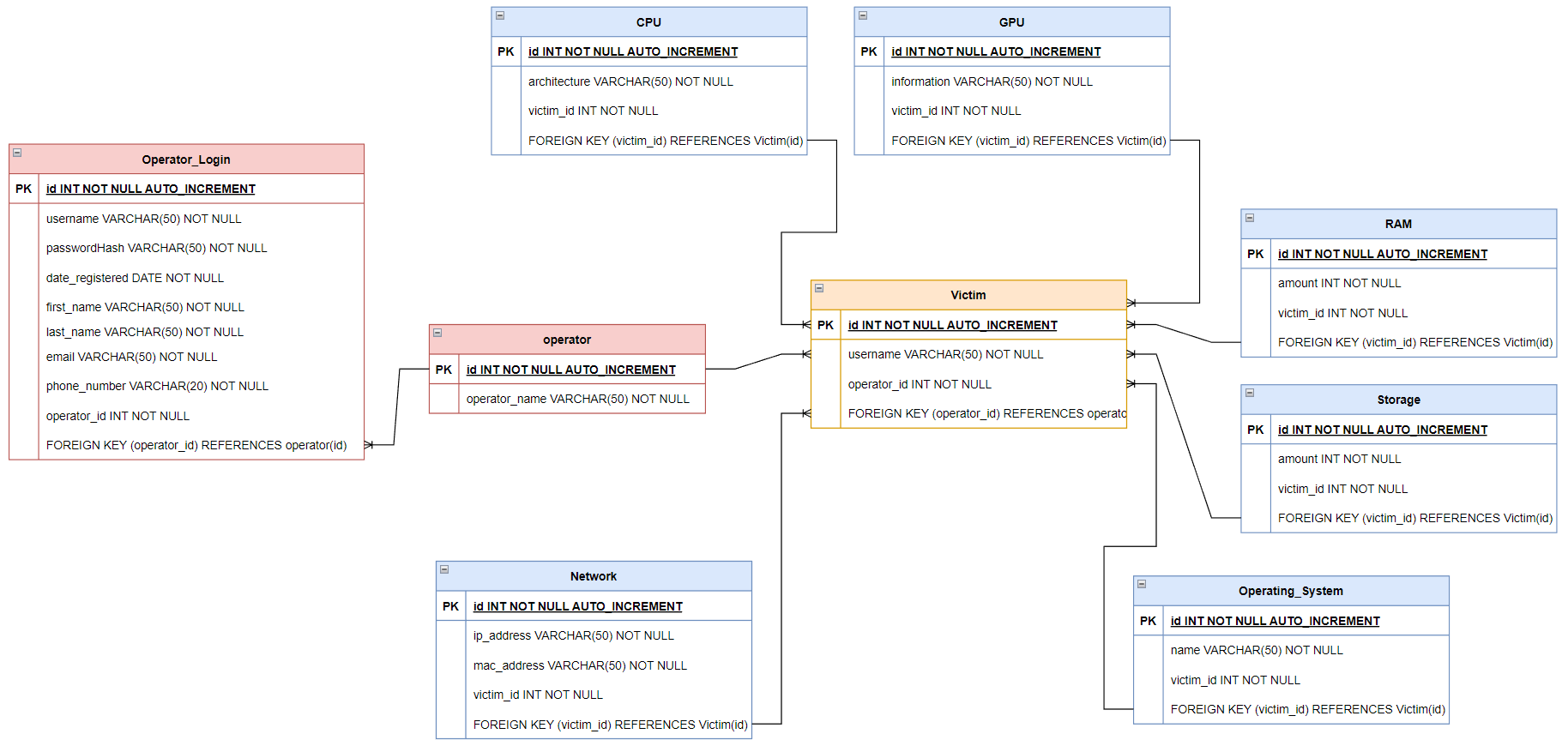




**Figure 3. Login Operator Flowchart Figure 4. Dashboard Operations**



**Figure 5. Implant Management**

**Figure 6. Database Diagram**

**A picture containing text, screenshot, line, diagram

Description automatically generated**

**Figure 7. Concept Diagram**

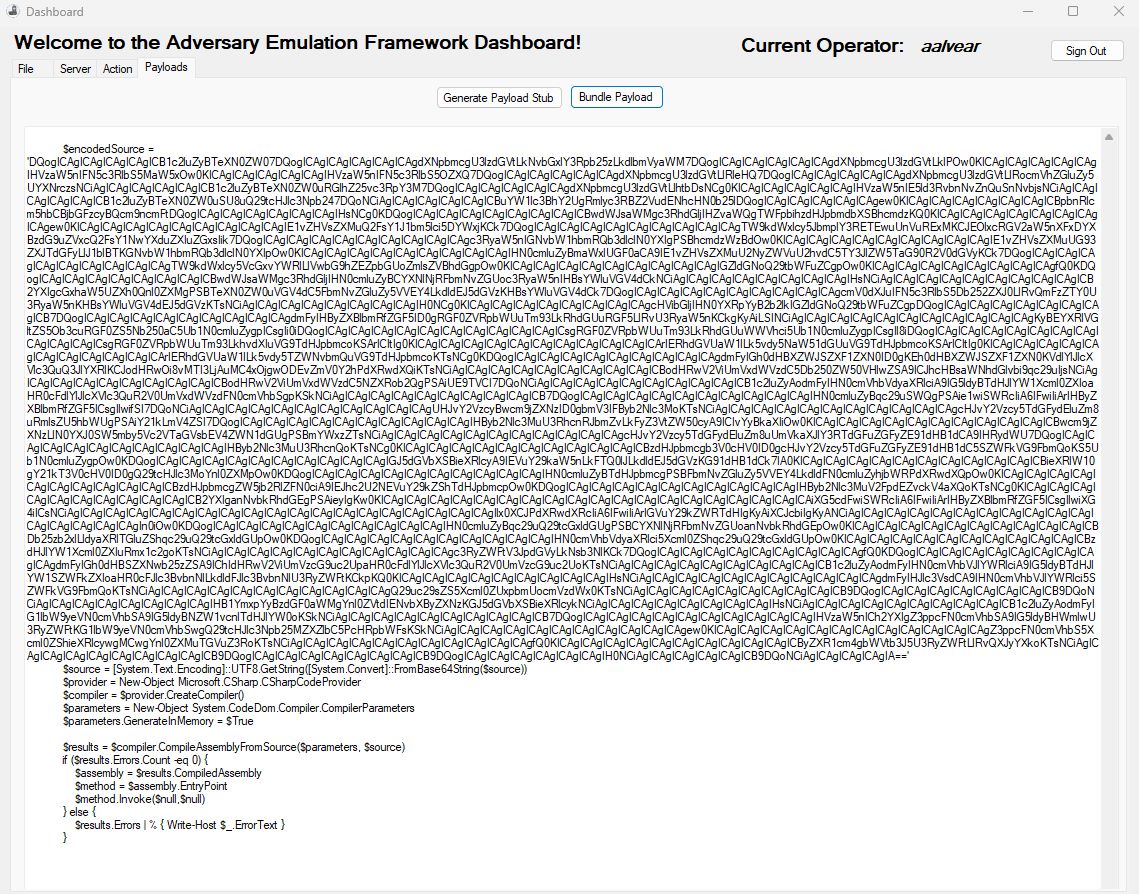
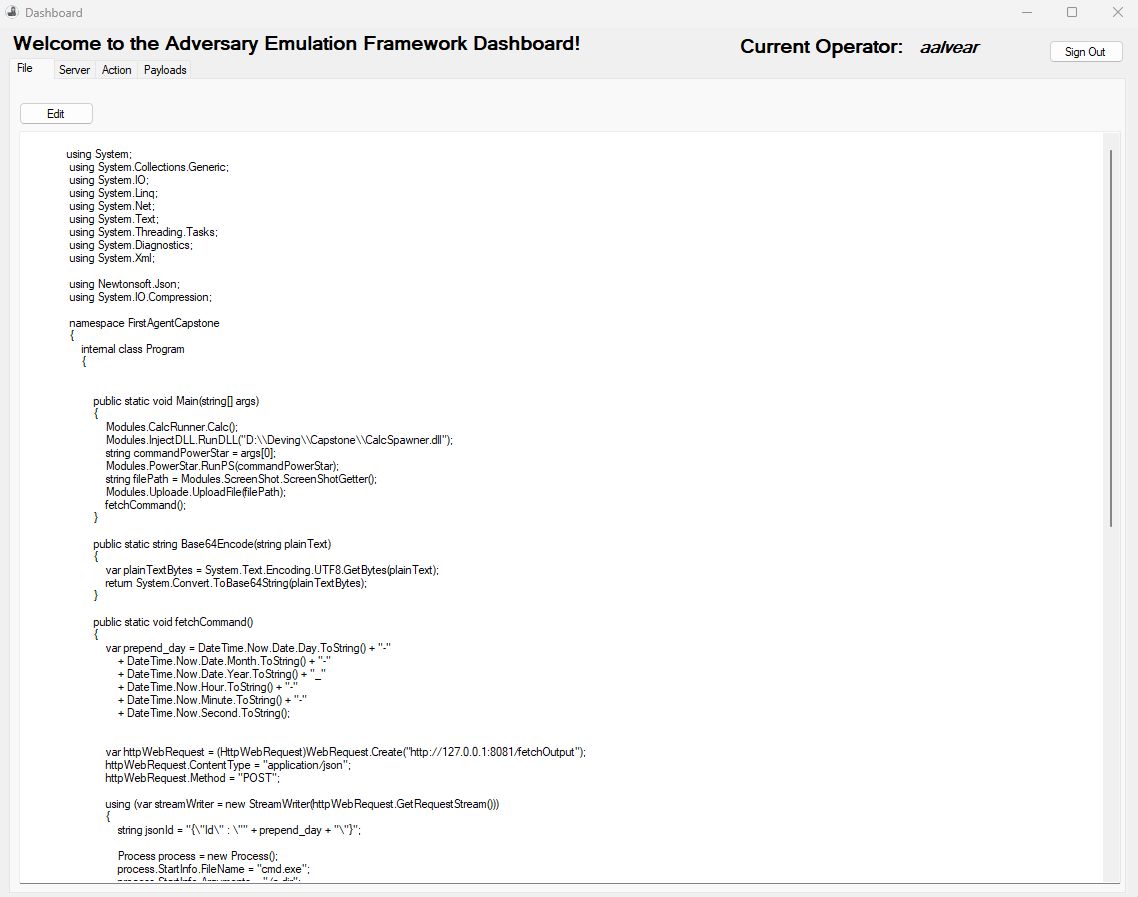
A picture containing text, diagram, line

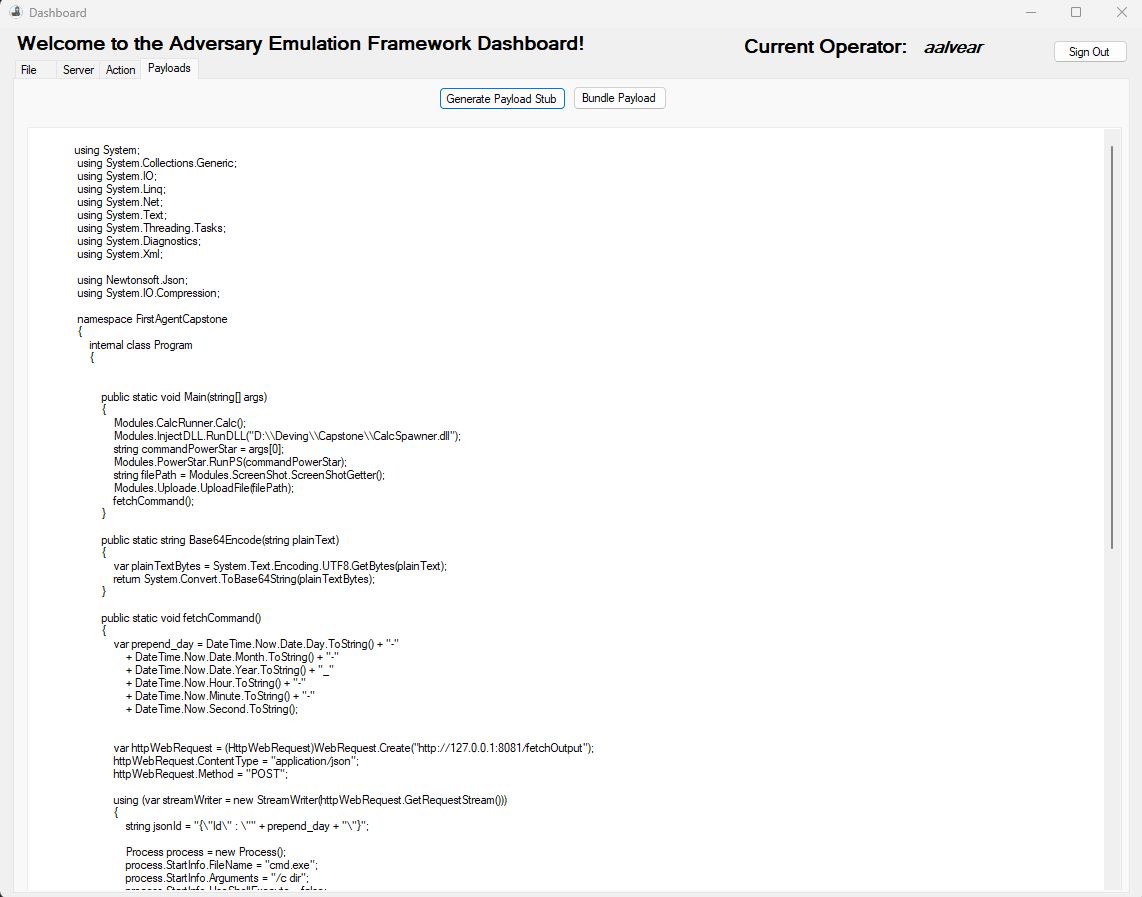
Description automatically generated

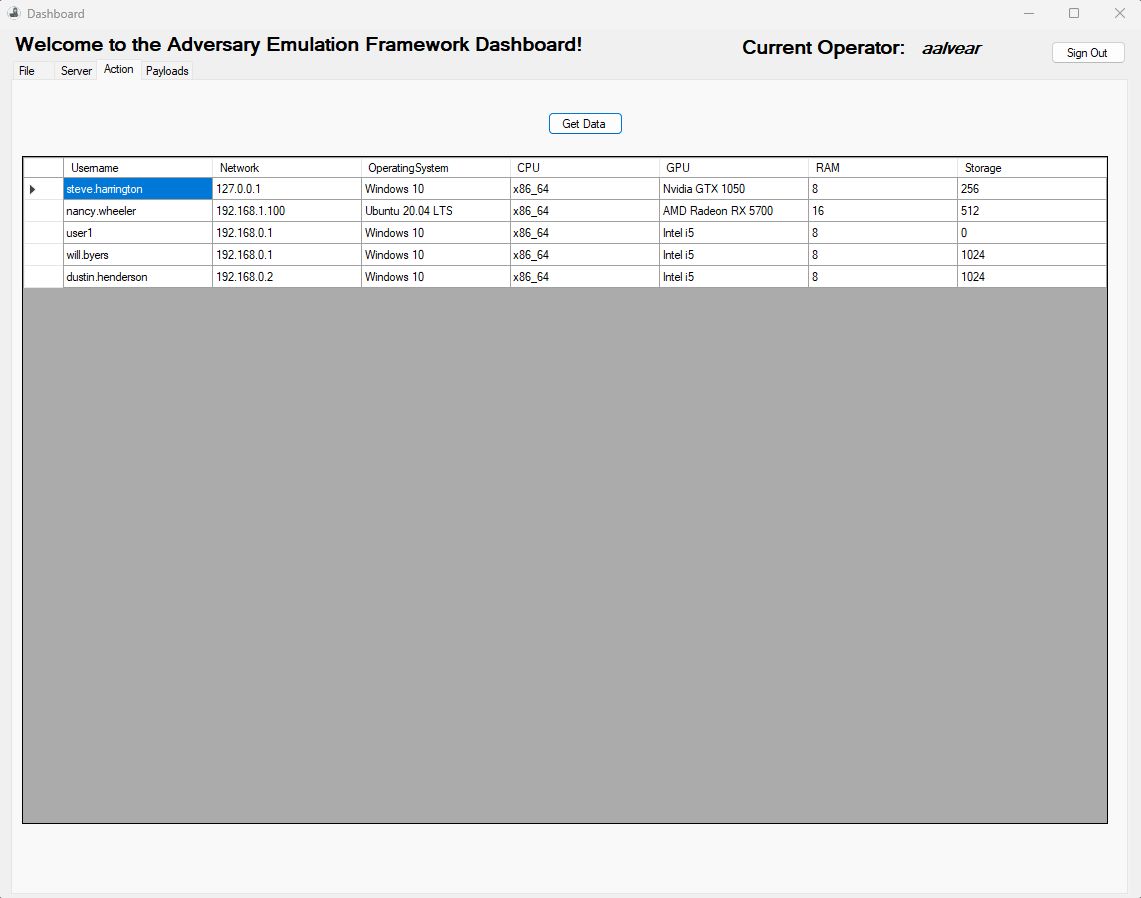
**Figure 8. Use Case Diagram**

A screenshot of a computer

Description automatically generated7.2. Prototype Demonstration

**Figure 9. Dashboard showing Payloads Bundle Tab**

**Figure 10. Dashboard showing Payloads Tab**

****

**Figure 11. Dashboard showing Data.**

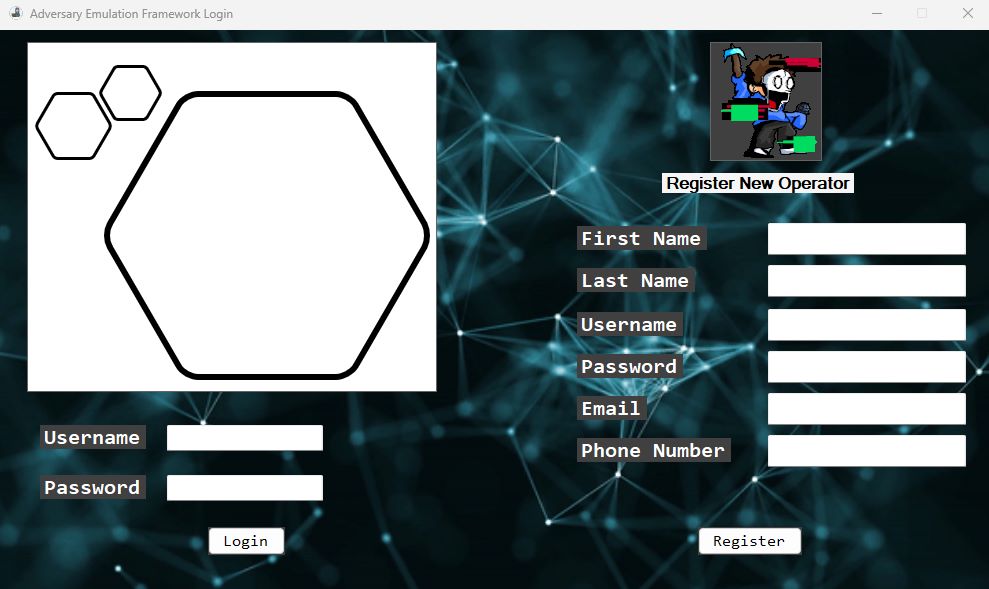


Figure 12. Operator login interface

1. Conclusions
   1. ***Concluding Remarks***

In conclusion, Adversary and Emulation Framework is an innovative approach to combating constantly involving nature of malware attacks. Its strategic approach to managing complex networked systems, combined with advanced techniques such as dynamic evasion, in-memory execution, and encrypted payloads, makes it a powerful tool in the fight against harmful programs. By using various evasion tactics and collecting victim information for a secure database, the framework enhances its effectiveness and resilience.

* 1. Recommendations for Future Work

In our upcoming efforts, we anticipate developing all features to operate as described below:

The system will be built to create a special tool called an implant. If the operator already has this tool, the system will take it in and manage it, otherwise, the system will make one, manage it, and get it ready to run. Then, it sets up a communication channel 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. The implant should cover evasion techniques that range from code obfuscation all the way through its life in memory to perform heap. encryption when it’s dormant, and many more. Using tools such as debuggers, we can analyze the internals of defense software to uncover new vectors for evasion.

1. 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>

(Accessed Apr. 30, 2023).

‌

[15] Monoxgas, “Monoxgas/SRDI: Shellcode implementation of reflective DLL injection. convert dlls to position independent shellcode,” GitHub, <https://github.com/monoxgas/sRDI> (accessed May 15, 2023).