ADVERSARY EMULATION FRAMEWORK

Kiara Rivera, Genesis Resto, Carlos Roque

krivera786@email.uagm.edu, gresto19@email.uagm.edu, croque16@email.uagm.edu

Mentors: Prof. Alcides Alvear Suarez & Robel Campbell

INTRODUCTION

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 more effectively counteract them.[1].

Objectives

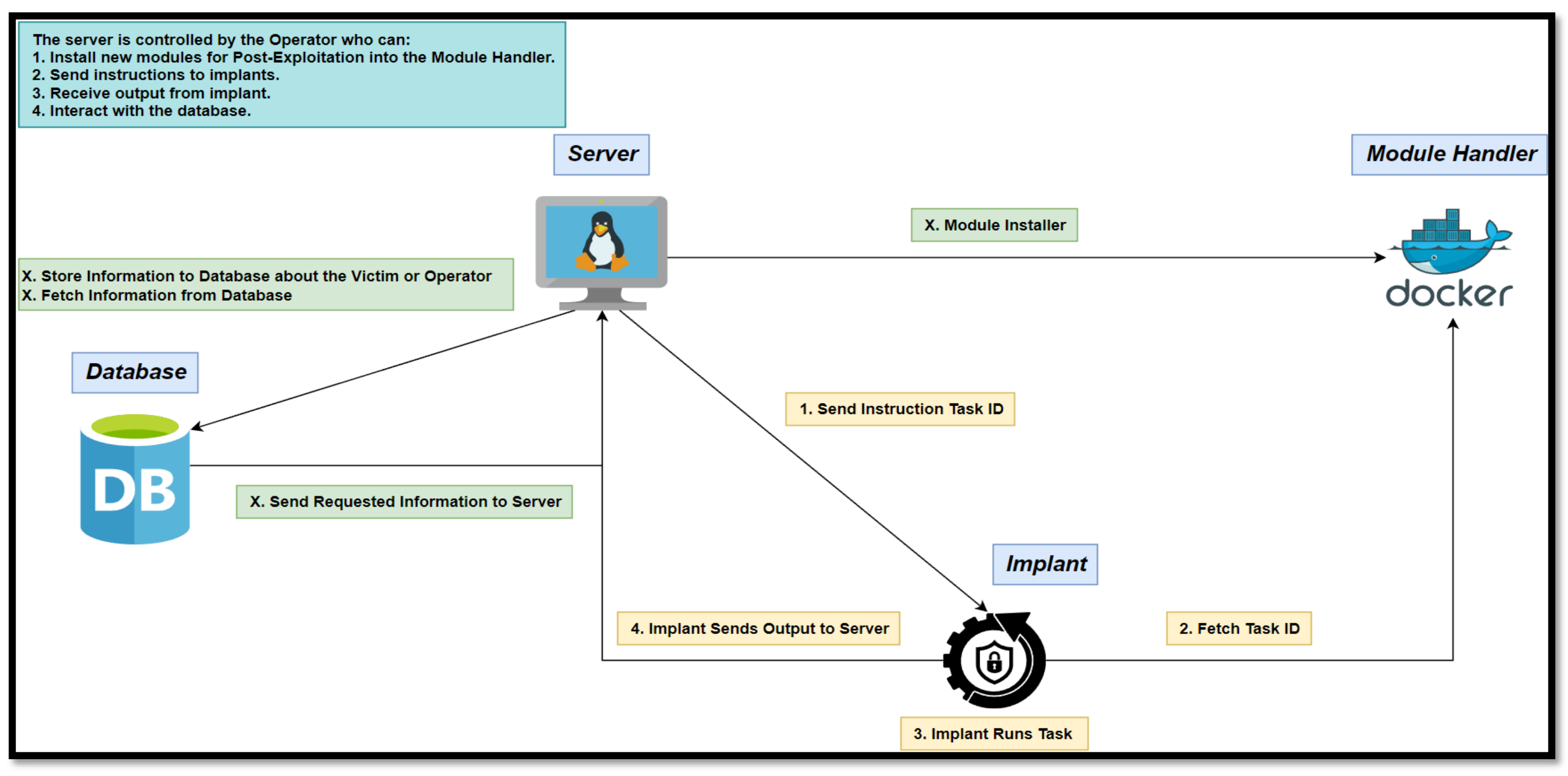
Develop a framework that offers a structured and strategic approach to managing complex networked systems and offers various options to expose these malware evasion techniques. This framework should employ the following to enhance its effectiveness and resilience:

- **Dynamic Evasion** – Variable Syscalls, ETW/AMSI Patching
- **In-memory Execution** – Download Cradles, Microcontroller
- **Encrypted Payloads** – AES-256, XOR

METHODOLOGY

Concept Diagram

The objective of the Concept Diagram is to illustrate how the entire system will be working together.



The server is controlled by the Operator who can:
1. Install new modules for Post-Exploitation into the Module Handler.
2. Send instructions to implants.
3. Receive output from implant.
4. Interact with the database.

Server
Module Handler
X. Module Installer
docker
X. Store Information to Database about the Victim or Operator
X. Fetch Information from Database
Database
DB
1. Send Instruction Task ID
X. Send Requested Information to Server
Implant
4. Implant Sends Output to Server
2. Fetch Task ID
3. Implant Runs Task

Figure 1. Concept Diagram

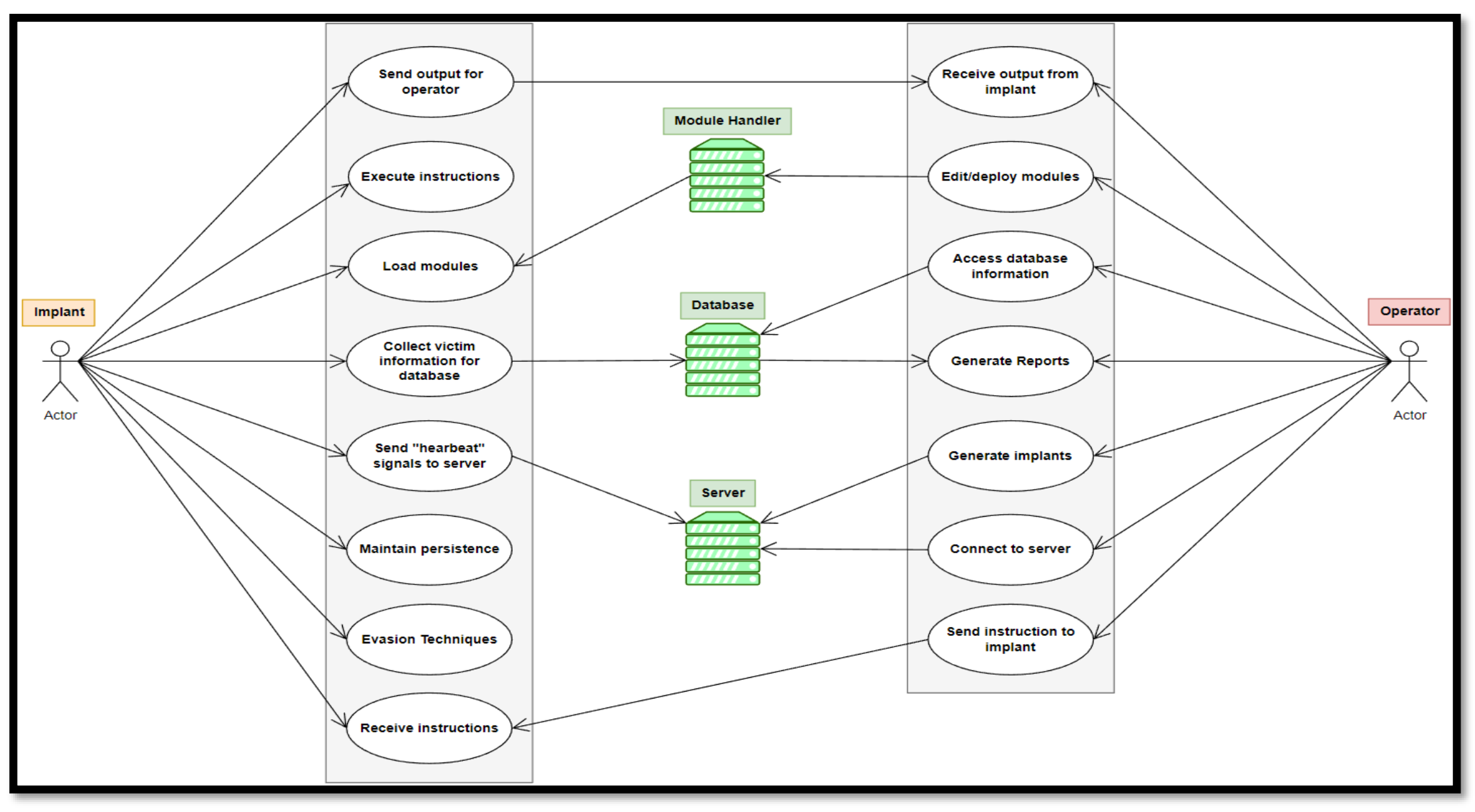


Figure 2. Use Case Diagram

Use Case Diagram

The *Use Case Diagram* is used to determine the functions of the Implant and Operator.

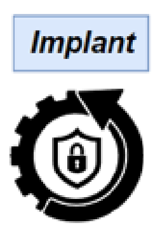


Figure 3. Implant

The implant employ various evasion tactics to avoid detection and collect victim information for database.

Figure 4. Database

The database helps storing and managing data related to victims, and operators securely. Also, enforce consistency and validation to avoid errors and corruption.

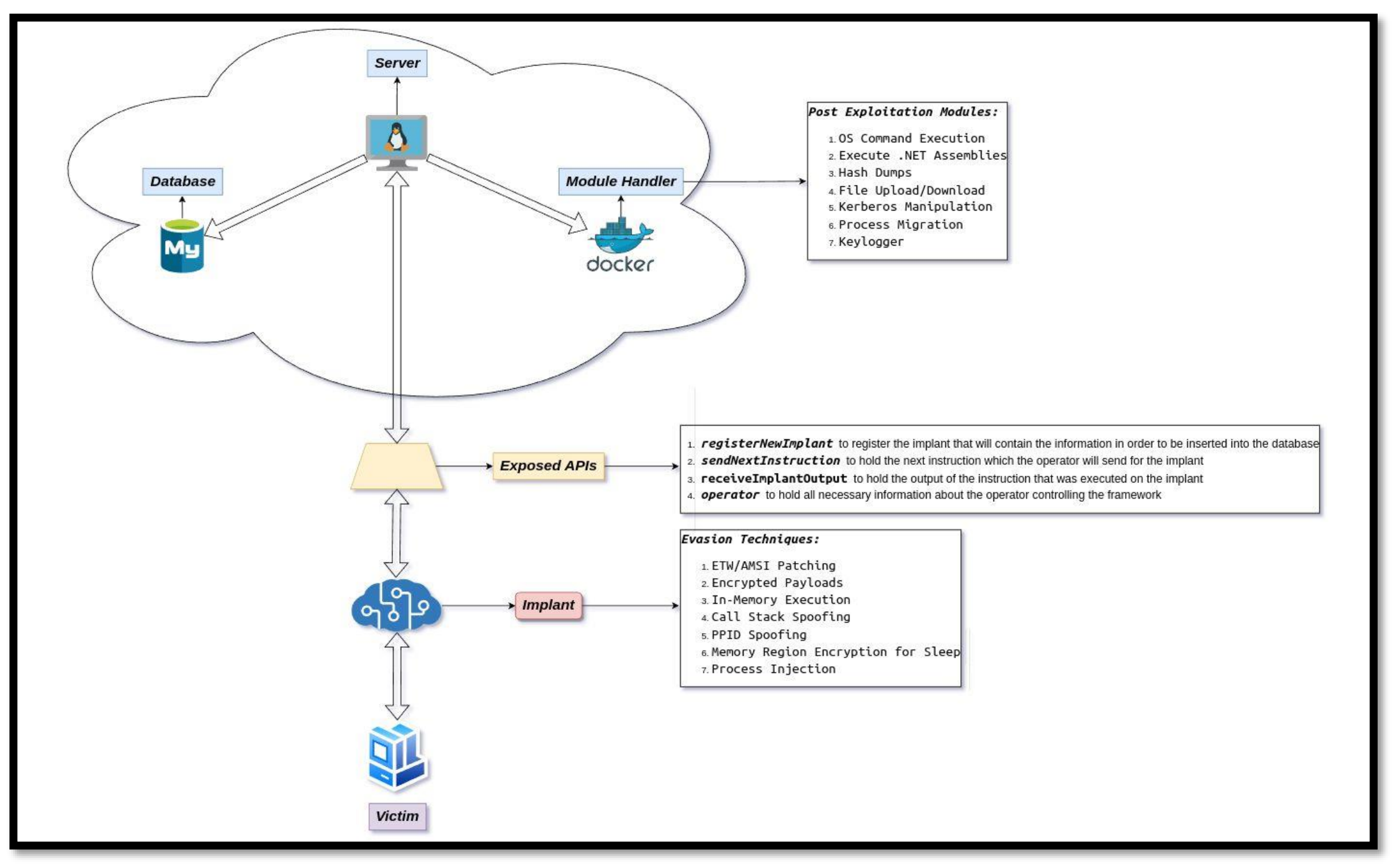
Figure 5. Server

The server can send instructions to the implant and receive the output, help storing victim information in a database, and you can customize your own implant.

RESULTS

Architecture Diagram

The purpose of the *Architecture Diagram* is to simulate how the command-and-control framework would look like for the operator with the functionalities it is expected to have.



Post Exploitation Modules:
1. OS Command Execution
2. Execute .NET Assemblies
3. Hash Dumps
4. File Upload/Download
5. Kerberos Manipulation
6. Process Migration
7. Keylogger

1. ***registerNewImplant*** to register the implant that will contain the information in order to be inserted into the database
2. ***sendNextInstruction*** to hold the next instruction which the operator will send for the implant
3. **receiveImplantOutput** to hold the output of the instruction that was executed on the implant
4. ***operator*** to hold all necessary information about the operator controlling the framework

Evasion Techniques:
1. ETW/AMSI Patching
2. Encrypted Payloads
3. In-Memory Execution
4. Call Stack Spoofing
5. PPID Spoofing
6. Memory Region Encryption for Sleep
7. Process Injection

Figure 6. Architecture Model

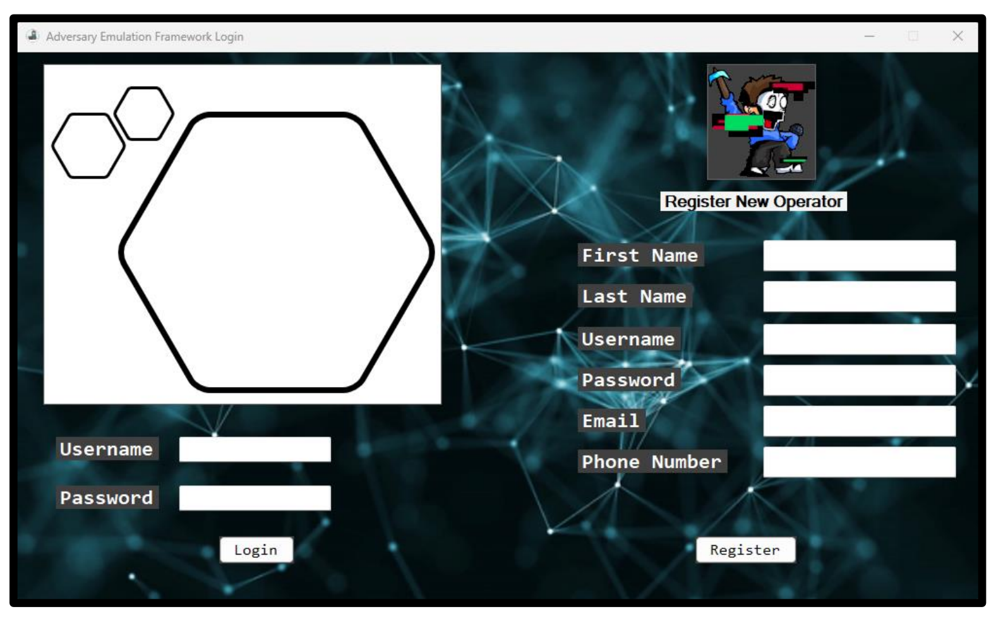
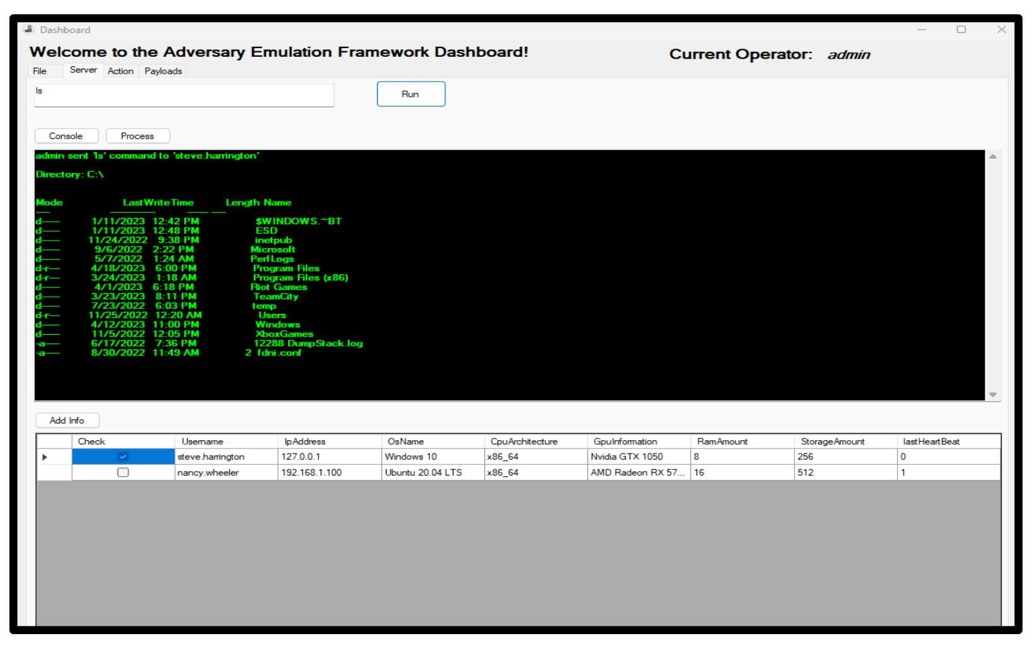


Figure 7. Login Screen



Dashboard

Welcome to the Adversary Emulation Framework Dashboard!   Current Operator: *admin*

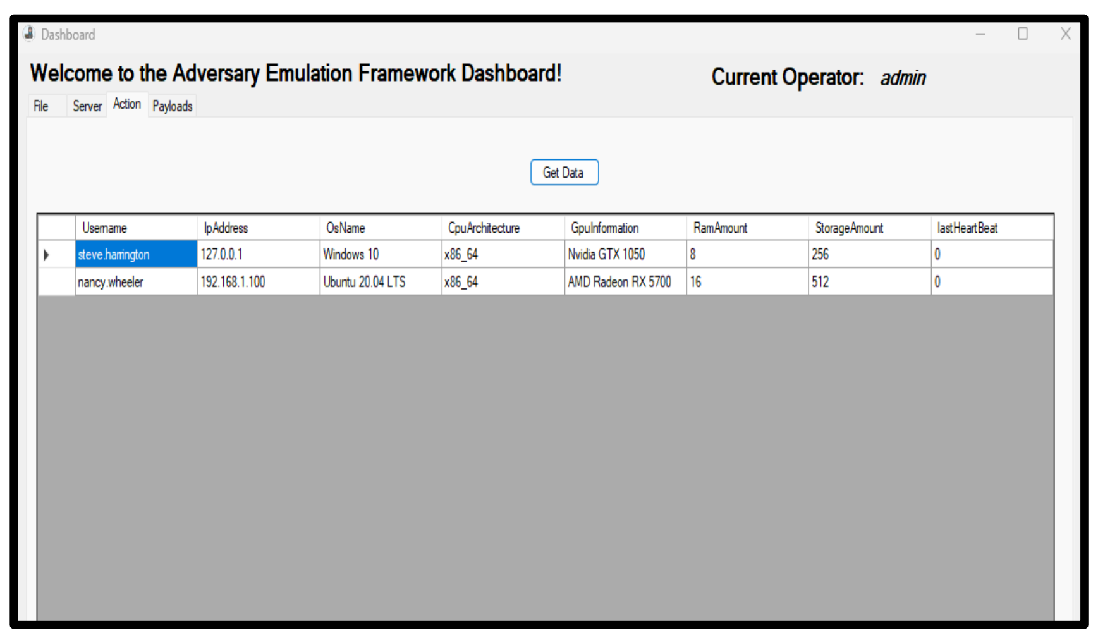
```
admin sent 'ls' command to 'steve.harrington'

Directory: C:\

Mode          LastWriteTime        Length Name
----          -------------        ------ ----
d-----     1/11/2023  12:42 PM            $WINDOWS.~BT
d-----     1/11/2023  12:48 PM            ESD
d-----    11/24/2022   9:38 PM            inetpub
d-----      9/6/2022   2:22 PM            Microsoft
d-----      5/7/2022   1:24 AM            PerfLogs
d-r---     4/18/2023   6:00 PM            Program Files
d-r---     3/24/2023   1:18 AM            Program Files (x86)
d-----      4/1/2023   6:18 PM            Riot Games
d-----     3/23/2023   8:11 PM            TeamCity
d-----     7/23/2022   6:03 PM            temp
d-r---    11/25/2022  12:20 AM            Users
d-----     4/12/2023  11:00 PM            Windows
d-----     11/5/2022  12:05 PM            XboxGames
-a----     6/17/2022   7:36 PM      12288 DumpStack.log
-a----     8/30/2022  11:49 AM          2 fdni.conf
```

	Check	Username	IpAddress	OsName	CpuArchitecture	GpuInformation	RamAmount	StorageAmount	lastHeartBeat
▸	☑	steve.harrington	127.0.0.1	Windows 10	x86_64	Nvidia GTX 1050	8	256	0
	☐	nancy.wheeler	192.168.1.100	Ubuntu 20.04 LTS	x86_64	AMD Radeon RX 57...	16	512	1

Figure 8. Dashboard for Operators



Dashboard

Welcome to the Adversary Emulation Framework Dashboard!   Current Operator: *admin*

	Username	IpAddress	OsName	CpuArchitecture	GpuInformation	RamAmount	StorageAmount	lastHeartBeat
▸	steve.harrington	127.0.0.1	Windows 10	x86_64	Nvidia GTX 1050	8	256	0
	nancy.wheeler	192.168.1.100	Ubuntu 20.04 LTS	x86_64	AMD Radeon RX 5700	16	512	0

Figure 9. Action Dashboard Tab

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 [2], we can analyze the internals of defense software to uncover new vectors for evasion.

REFERENCES

1. [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 Available at: https://doi.org/10.1145/3073559.
2. [2] Domars (2023) WinDbg Overview - windows drivers, WinDbg Overview - Windows drivers | Microsoft Learn. Microsoft. Available at: https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/windbg-overview (Accessed: January 30th, 2023).
3. [X] 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 Available at: https://doi.org/10.5815/ijeme.2018.02.03.
4. [X] E. Chaffey, D. Sgandurra, and R. Holloway, "Malware vs Anti-Malware Battle - Gotta Evade 'em All!," 2020. Accessed: March 10th, 2023. [Online]. Available at: https://core.ac.uk/download/pdf/340199834.pdf

UNIVERSIDAD
ANA G. MÉNDEZ
UAGM

CAPSTONE 1:
DESIGN CONCEPTS