**Malware Simulation & Analysis Report**

**Author:** Purna Kishore Kondapaneni  
**Date:** May 15, 2025  
**Project Title:** Binary Executable Simulation for Malware Analysis

**1. Objective**

The goal of this project is to simulate a malware executable that performs malicious actions such as downloading a payload from the internet, creating and deleting local files, and modifying the Windows registry. The executable is then analyzed through static and dynamic techniques using various malware analysis tools, followed by reverse engineering and crafting of YARA rules for detection.

**2. Malware Creation**

**2.1 C Code Description**

The malware executable performs the following operations:

* Downloads a remote executable using HTTP.
* Creates and deletes two files in the local system.
* Writes a custom registry key-value pair.

**2.2 C Code Sample (malware.c)**

Here, Used C code, using Windows APIs, where it first downloads payload.exe from http://127.0.0.1/. Then, it creates test\_file1.txt with specific content and test\_file2.dat with binary data. Next, it creates a registry key Software\Evil app and sets a Config value to "malicious data". Finally, it attempts to delete both created files. In essence, it's a simple program demonstrating file download, creation, deletion, and registry modification on a Windows system.

**2.3 Compilation on FLARE VM**

gcc malware.c -o malware.exe -lwininet

1. Save the code as malware.c.
2. Open Command Prompt as an administrator.
3. Compile: gcc malware.c -o malware.exe -lwininet
4. Verify malware.exe is created.

**2.4 Local Server Setup (Python) / Setting up the Simple HTTP Server (Simplified)**

Preparing a Payload:

mkdir C:\\EvilServer

copy C:\\Windows\\System32\\calc.exe C:\\EvilServer\\payload.exe

Starting a Server:

* + In the same Command Prompt, navigate to C:\EvilServer:
  + cd C:\EvilServer
  + Run: python -m http.server 80

A screenshot of a computer screen

AI-generated content may be incorrect.

chose calc.exe as a benign payload for testing. Creating the C:\EvilServer directory and copying the executable.

A screenshot of a computer

AI-generated content may be incorrect.

When I clicked on the link, it was downloaded and able to open a calculator

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**3. Static Analysis**

**3.1 Tools Used**

* Strings Utility
* PEStudio
* Dependency Walker
* Registry Editor
* Detect It Easy
* CFF Explorer

**3.2 Observations**

* **Strings:** URL, file paths, registry keys are visible as we defined in the C code.

A screenshot of a computer

AI-generated content may be incorrect.

* **PEStudio:** No digital signature, no packing.

A screenshot of a computer

AI-generated content may be incorrect.

* **Dependency Walker:** Identifies WinAPI dependencies (Wininet.dll, Kernel32.dll). Dependency Walker is an application that builds a hierarchical tree diagram of other programs and all dependent modules. For each module found, it lists all the functions that are exported by that module, and which of those functions are actually being called by other modules.

A screen shot of a computer

AI-generated content may be incorrect.

* **Detect It Easy:** Recognized as PE64 executable.

A screenshot of a computer program

AI-generated content may be incorrect.

* **CFF Explorer, HEX Editor:** Valid PE structure.

A screenshot of a computer

AI-generated content may be incorrect.

* **Registry Editor:** Custom key HKCU\\Software\\EvilApp with value Config = malicious\_data.
* A screenshot of a computer

  AI-generated content may be incorrect.

**4. Dynamic Analysis**

**4.1 Tools Used**

* Procmon
* Regshot
* virustotal.com

**4.2 File and Registry Activity (Procmon)**

* File create/delete actions captured.
* Registry modification logged under HKCU\\Software\\EvilApp
* calc.exe downloaded and stored as evil\_payload.exe

A screenshot of a computer

AI-generated content may be incorrect.

**4.3 Regshot Comparison**

* New registry key added: HKCU\\Software\\EvilApp
* No additional persistent changes

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**4.4 Virus Total Scan**

* Simulated sample recognized by several AV engines as "Malicious/Trojan"

A screenshot of a computer

AI-generated content may be incorrect.

**5. Reverse Engineering**

**5.1 Ghidra Analysis (Disassembling Malware)**

* Decompiled code matches behavior described in source
* Key functions: RegCreateKey, RegSetValueEx

A screenshot of a computer

AI-generated content may be incorrect.

Details of the imported file are shown here

A screenshot of a computer

AI-generated content may be incorrect.

This Screenshot shows the Ghidra windows

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

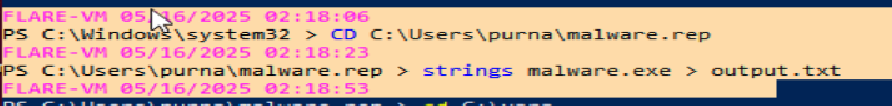
AI-generated content may be incorrect.

This Picture is showing the function graph in the listing window.

**5.2. YARA Rules (Signature based Detections):**



1st step: Downloaded the strings from .exe file (strings malware.exe > output.txt)



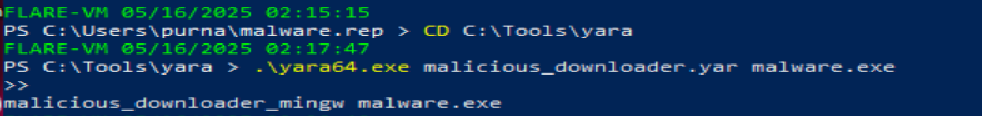
A screenshot of a computer program

AI-generated content may be incorrect.

2nd step: With the above strings downloaded, able to create the Yara rules and saved as malicious\_downloader.yar and extracted the latest yara-win64 zip file.

A screenshot of a computer

AI-generated content may be incorrect.



As the Yara Rule matches, we got the output of the name of the rule and file name (As yara64.exe starts the Yara scanning engine, it loads all rules from malicious\_downloader.yar., and it analyzes malware.exe using those rules).

**Final Notes:**

| * **Component** | * **Observed Behaviors** |
| --- | --- |
| * Downloader | * Successful download via WinINet |
| * File system | * File creation and deletion |
| * Registry | * New key/values in HKCU |
| * Static detection | * No obfuscation; flagged as suspicious |
| * Dynamic detection | * File, registry, and HTTP activity confirmed |
| * Reverse engineering | * All malicious logic visible in decompiled code |
| * YARA rule | * Matches based on unique strings and APIs |

**6. Conclusion**

This simulated malware was effective in demonstrating the behavior of typical downloader malware, incorporating real-world tactics like file manipulation, registry editing, and payload retrieval. The exercise enabled hands-on analysis using static and dynamic tools, and helped in crafting effective detection techniques using YARA rules. This forms a foundational understanding for identifying and analyzing real-world threats.

The simulated malware is very basic, lacking complexity, obfuscation, and evasion techniques of real threats. The analysis was likely shallow, focusing on immediate effects without deep behavioral or network inspection. The reliance on localhost and a controlled VM environment also simplifies the scenario compared to real-world infections. Finally, the YARA rule is likely basic and might not be robust against variations.