

Malware Simulation & Analysis Report

Author: Purna Kishore Kondapaneni

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

```
on Select Administrator: Command Prompt - python -m http.server 80
C:\malware>mkdir C:\EvilServer
A subdirectory or file C:\EvilServer already exists.

FLARE-VM Tue 05/13/2025 14:20:22.62
C:\malware>copy C:\Windows\System32\calc.exe C:\EvilServer\payload.exe
1 file(s) copied.

FLARE-VM Tue 05/13/2025 14:21:02.53
C:\malware>cd C:\EvilServer

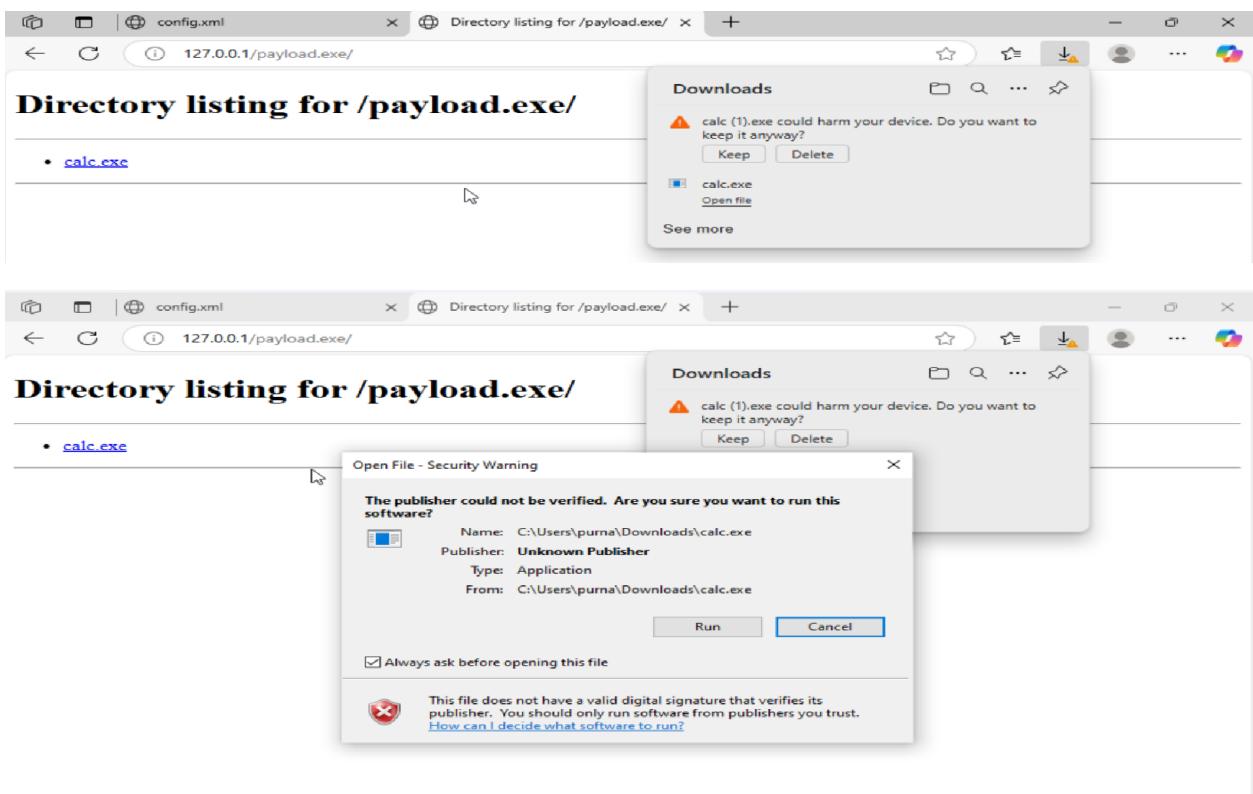
FLARE-VM Tue 05/13/2025 14:21:26.76
C:\EvilServer>python -m http.server 80
Serving HTTP on :: port 80 (http://[::]:80) ...
::ffff:127.0.0.1 - - [13/May/2025 14:35:05] "GET / HTTP/1.1" 200 -
::ffff:127.0.0.1 - - [13/May/2025 14:35:05] code 404, message File not found
::ffff:127.0.0.1 - - [13/May/2025 14:35:05] "GET /favicon.ico HTTP/1.1" 404 -
::ffff:127.0.0.1 - - [13/May/2025 14:35:09] "GET /payload.exe/ HTTP/1.1" 200 -
::ffff:127.0.0.1 - - [13/May/2025 14:35:16] "GET /payload.exe/calc.exe HTTP/1.1" 200 -
::ffff:127.0.0.1 - - [13/May/2025 14:35:16] "GET /payload.exe/calc.exe HTTP/1.1" 200 -

Exception occurred during processing of request from ('::ffff:127.0.0.1', 50593, 0, 0)
Traceback (most recent call last):
  File "C:\Python310\lib\socketserver.py", line 683, in process_request_thread
    self.finish_request(request, client_address)
  File "C:\Python310\lib\http\server.py", line 1304, in finish_request
    self.RequestHandlerClass(request, client_address, self,
  File "C:\Python310\lib\http\server.py", line 668, in __init__
    super().__init__(*args, **kwargs)
  File "C:\Python310\lib\socketserver.py", line 747, in __init__
    self.handle()
  File "C:\Python310\lib\http\server.py", line 433, in handle
    self.handle_one_request()
  File "C:\Python310\lib\http\server.py", line 421, in handle_one_request
    method()
  File "C:\Python310\lib\http\server.py", line 675, in do_GET
    self.copyfile(f, self.wfile)
  File "C:\Python310\lib\http\server.py", line 875, in copyfile
    shutil.copyfileobj(source, outputfile)
  File "C:\Python310\lib\shutil.py", line 198, in copyfileobj
    fdst.write(buf)
  File "C:\Python310\lib\socketserver.py", line 826, in write
    self._sock.sendall(b)
ConnectionAbortedError: [WinError 10053] An established connection was aborted by the software in your host machine
```

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



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



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.

```
C:\EvilServer>CD C:\  
FLARE-VM Tue 05/13/2025 14:57:45.48  
C:\>cd malware  
FLARE-VM Tue 05/13/2025 14:57:55.18  
C:\malware>strings malware.exe  
!This program cannot be run in DOS mode.  
.text  
.data  
.rdata  
@.pdata  
@.xdata  
@.bss  
.idata  
.CRT  
.tls  
.reloc  
B/19  
B/31  
B/45  
B/57  
B/70  
B/81  
8MZu  
HcP<H  
ATUWVSH  
[^\n]A\  
[^\n]A\  
D$8H  
T$ A  
)t$@  
)|$PD  
)D$~  
D$0I  
|$(H
```

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

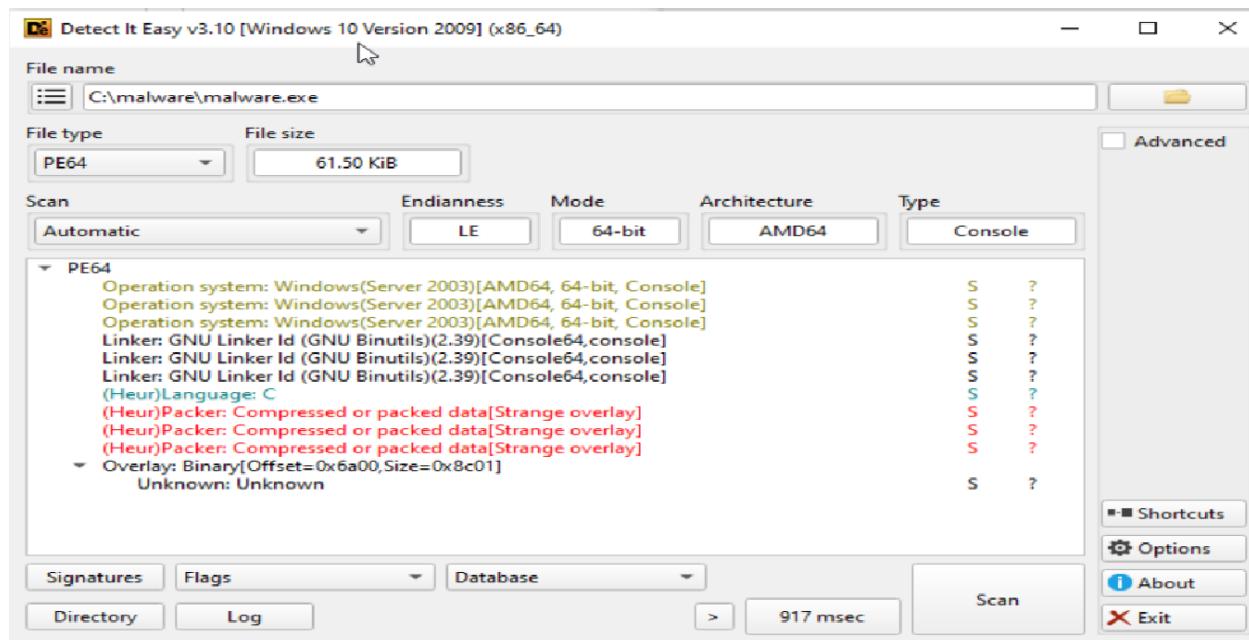
The screenshot shows the PEStudio interface with the file 'malware.exe' open. On the left, a tree view shows various sections like indicators, file headers, optional headers, and imports. On the right, a table lists the imported libraries and their details:

library (11)	flag (1)	type	imports (63)	description
ADVAPI32.dll	-	Implicit	2	Advanced Windows 32 Base API
KERNEL32.dll	-	Implicit	11	Windows NT BASE API Client
api-ms-win-crt-e...	-	Implicit	2	Windows ApiSet Stub Library
api-ms-win-crt-h...	-	Implicit	4	Windows ApiSet Stub Library
api-ms-win-crt-...	-	Implicit	1	Windows ApiSet Stub Library
api-ms-win-crt-p...	-	Implicit	2	Windows ApiSet Stub Library
api-ms-win-crt-r...	-	Implicit	17	Windows ApiSet Stub Library
api-ms-win-crt-s...	-	Implicit	9	Windows ApiSet Stub Library
api-ms-win-crt-t...	-	Implicit	3	Windows ApiSet Stub Library
api-ms-win-crt-fo...	-	Implicit	4	Windows ApiSet Stub Library
WININET.dll	x	Implicit	7	Internet Extensions for Win32 Library

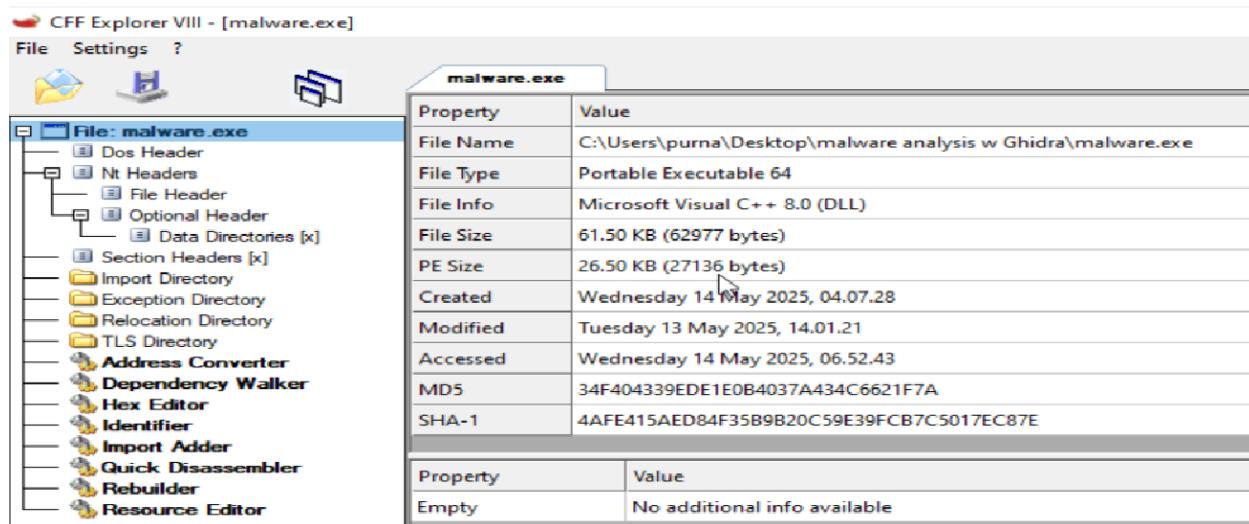
- **Dependency Walker:** Identifies WinAPI dependencies (Wininet.dll, Kernel32.dll). Dependency Walker is an application that builds a hierarchical tree diagram of other programs and all dependent modules. For each module found, it lists all the functions that are exported by that module, and which of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

```
FLARE-VM Wed 05/14/2025 2:27:45.27
C:\malware>malware.exe;
Downloading file from http://127.0.0.1/payload.exe to C:\Users\Public\evil_payload.exe...
HttpSendRequestA failed: 12007
File download failed.
Creating local files...
Created: C:\Users\Public\test_file1.txt
Created: C:\Users\Public\test_file2.dat
Registry key created: HKEY_CURRENT_USER\Software\EvilApp
Registry value set: malicious_data = LQVS\0
Deleting local files...
Deleted: C:\Users\Public\test_file1.txt
Deleted: C:\Users\Public\test_file2.dat
```

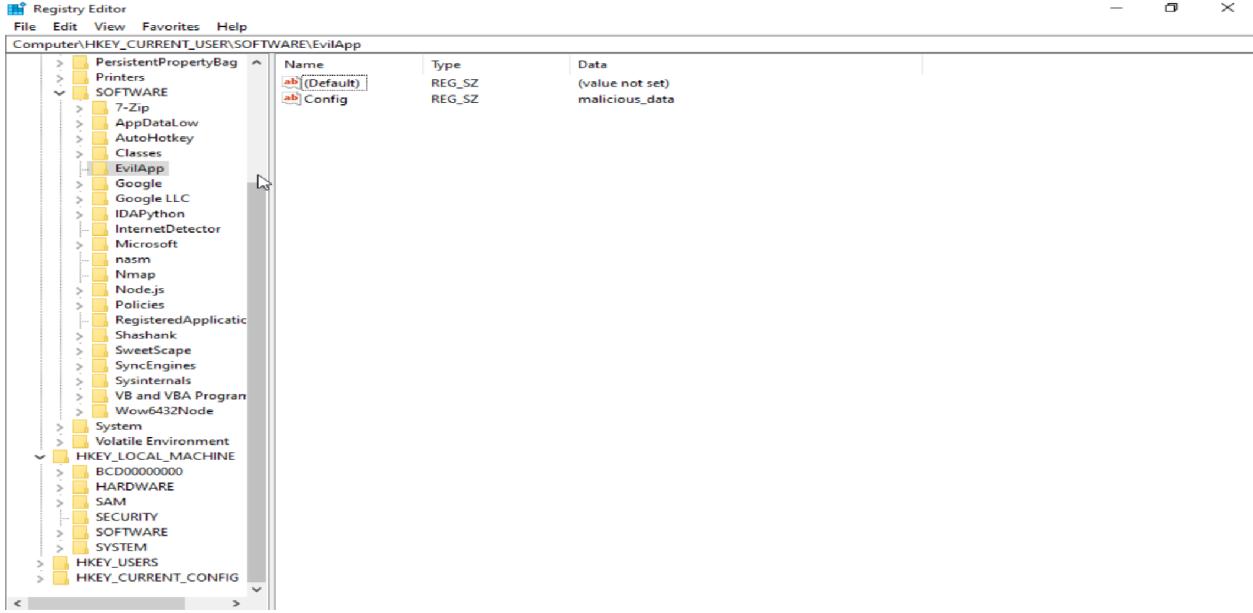
- **Detect It Easy:** Recognized as PE64 executable.



- CFF Explorer, HEX Editor: Valid PE structure.



- **Registry Editor:** Custom key HKCU\Software\EvilApp with value Config = malicious_data.



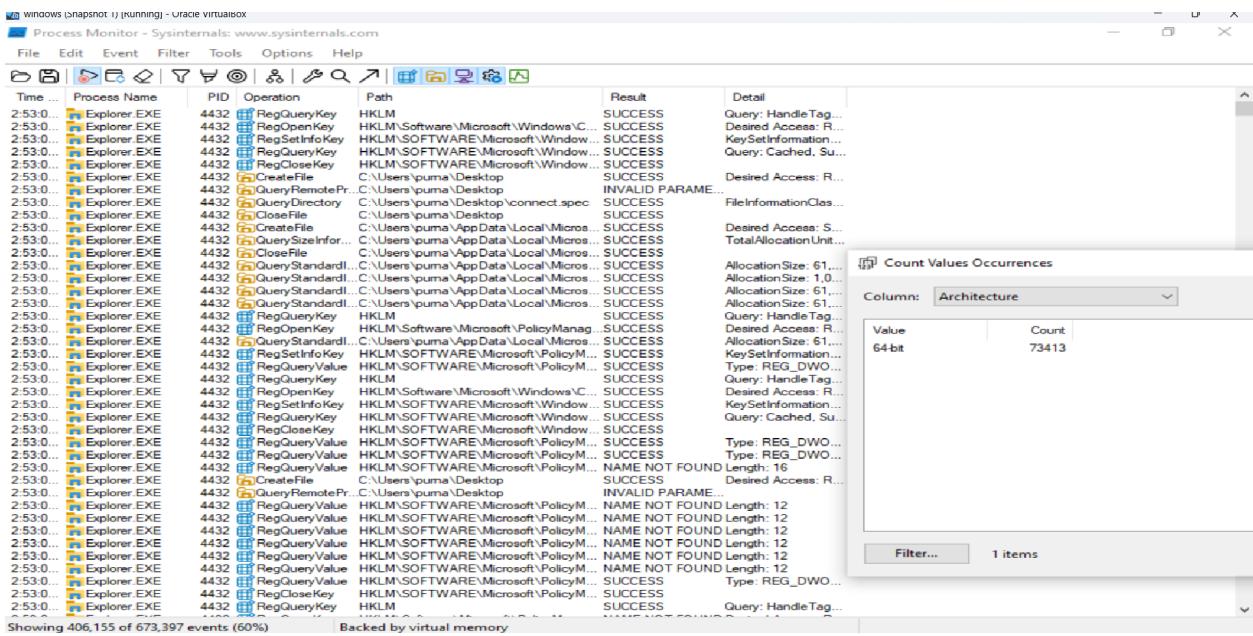
4. Dynamic Analysis

4.1 Tools Used

- Procmon
- Regshot
- virustotal.com

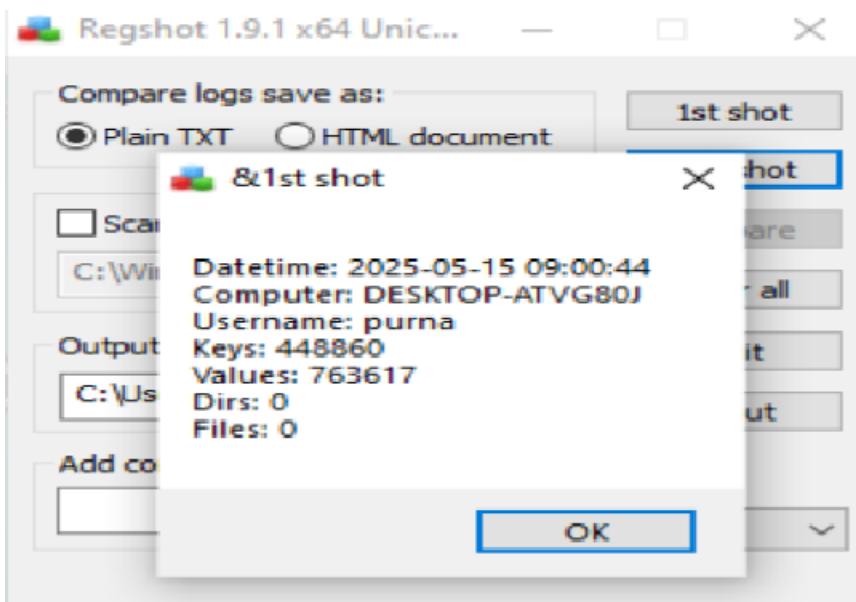
4.2 File and Registry Activity (Procmon)

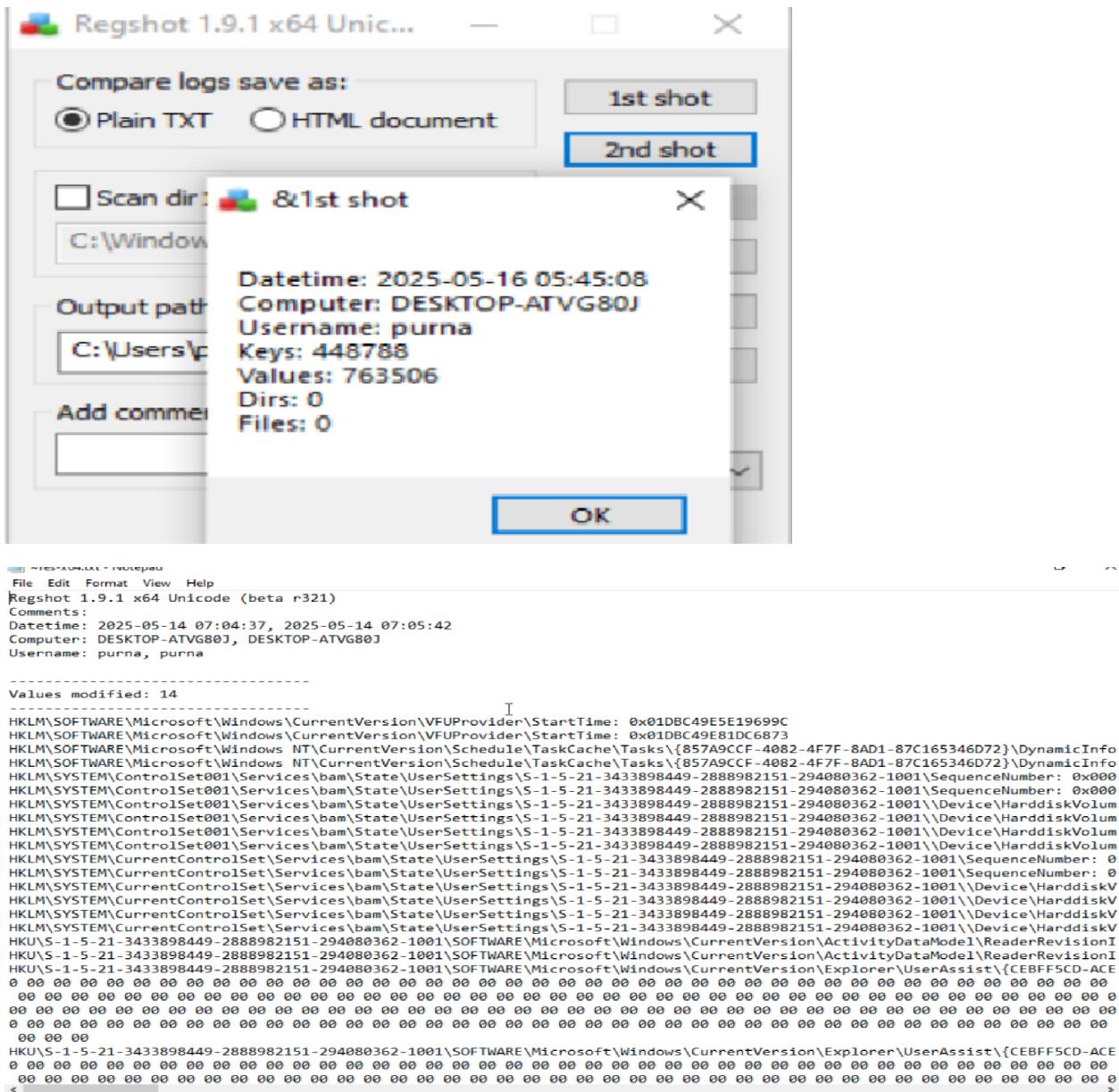
- File create/delete actions captured.
- Registry modification logged under HKCU\Software\EvilApp
- calc.exe downloaded and stored as evil_payload.exe



4.3 Regshot Comparison

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





4.4 Virus Total Scan

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

The screenshot shows the VirusTotal analysis interface for the file `eb11372e631ee5f6741b860fb0e3e28a2771abc309307014a504c98f90d4104d`. The main summary panel indicates 10/72 security vendors flagged it as malicious. The file is identified as `malware_sim.exe` and has a size of 57.62 KB. The analysis was performed a moment ago. The file type is EXE. Below the summary, there are tabs for DETECTION, DETAILS, RELATIONS, BEHAVIOR, and COMMUNITY. A banner encourages joining the community for additional insights. The SECURITY VENDORS' ANALYSIS section lists various vendor results, including Bkav Pro, Cynet, Google, Ikarus, SecureAge, Acronis (Static ML), and Alibaba. A button labeled "Activate Windows" is visible on the right.

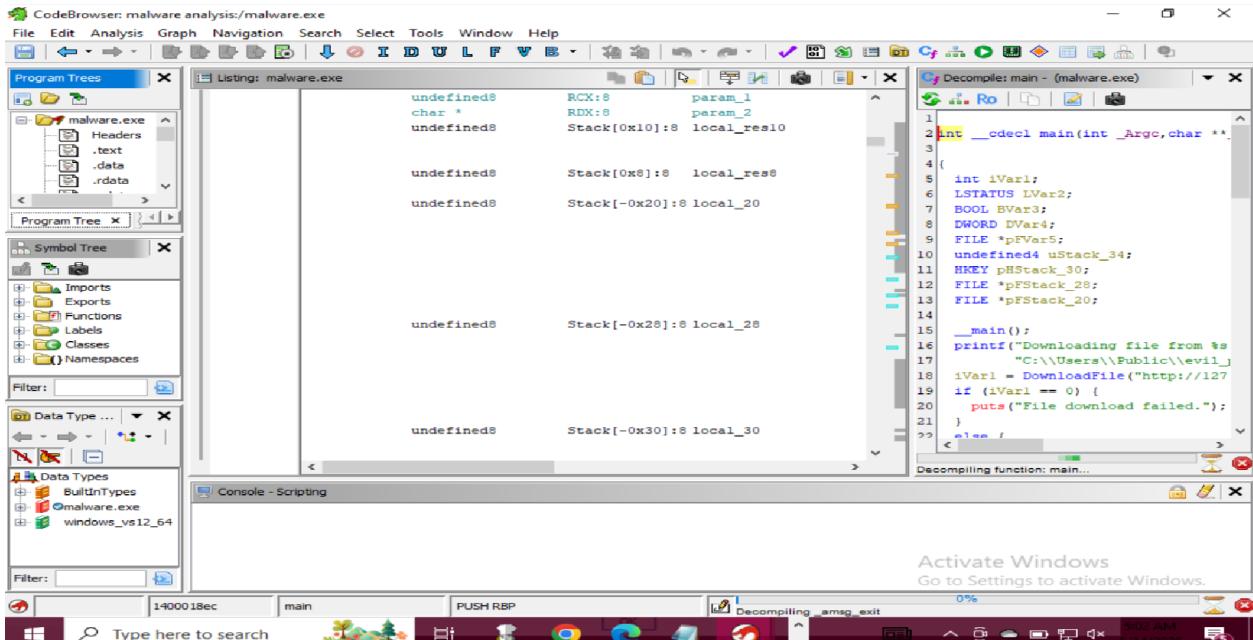
5. Reverse Engineering

5.1 Ghidra Analysis (Disassembling Malware)

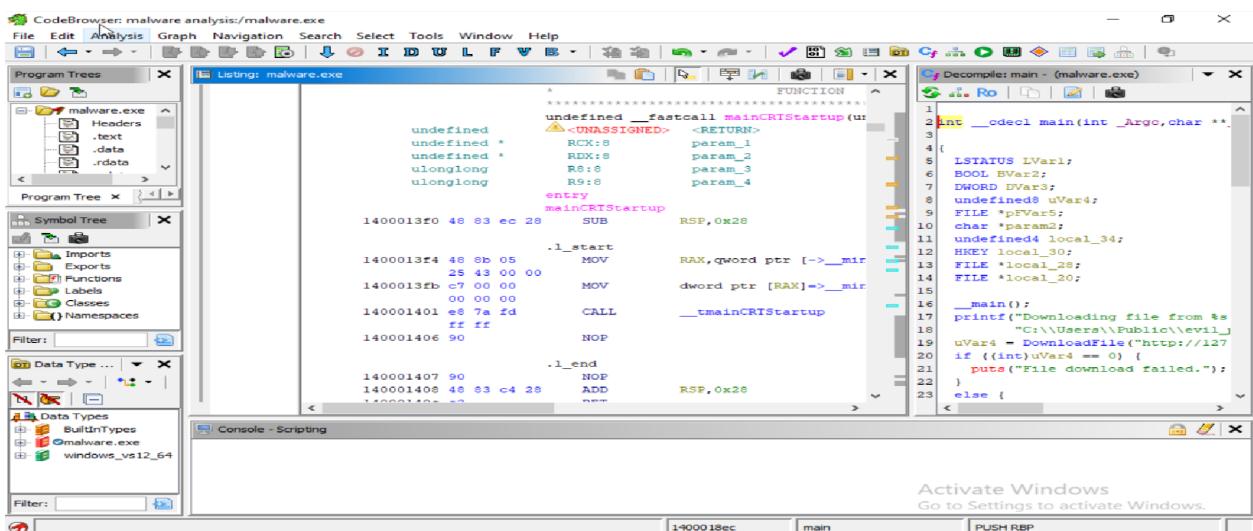
- Decompiled code matches behavior described in source
- Key functions: RegCreateKey, RegSetValueEx

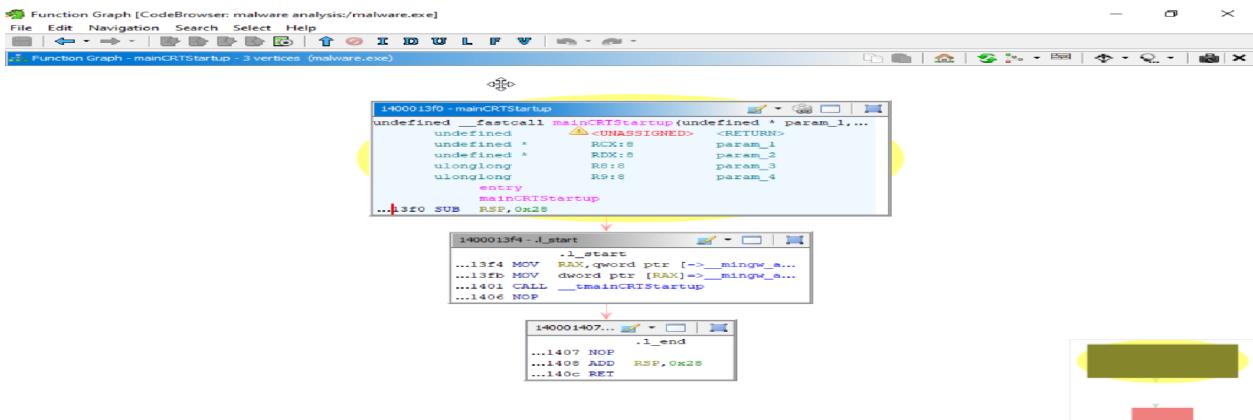
The screenshot shows the Ghidra software interface for the project `malware`. The left sidebar displays the tool chest and active project structure. The central pane shows the project configuration details, including the project file name (`malware.exe`), last modified date (May 14, 2025), and various processor and memory settings. The bottom pane shows additional information about imported symbols, such as API entries for CRT runtime and string functions. A watermark at the bottom right encourages activating Windows.

Details of the imported file are shown here



This Screenshot shows the Ghidra windows





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

5.2. YARA Rules (Signature based Detections):

```
FLARE-VM Thu 05/15/2025 3:56:08.72
C:\malware>yara ..\YaraRules\MaliciousDownloader.yar malware.exe
```

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

```
FLARE-VM 05/16/2025 02:18:06
PS C:\Windows\system32 > CD C:\Users\purna\malware.rep
FLARE-VM 05/16/2025 02:18:23
PS C:\Users\purna\malware.rep > strings malware.exe > output.txt
FLARE-VM 05/16/2025 02:18:53
```

```
*output.txt - Notepad
File Edit Format View Help
malicious_data
Registry value set: %s = %s
Failed to set registry value.
Failed to create/open registry key.
Deleting local files...
Deleted: %s
Failed to delete: %s (Error: %lu)
Argument domain error (DOMAIN)
Argument singularity (SIGN)
Overflow range error (OVERFLOW)
Partial loss of significance (PLOSS)
Total loss of significance (TLOSS)
The result is too small to be represented (UNDERFLOW)
Unknown error
_matherr(): %s in %s(%g, %g) (retval=%g)
Mingw-w64 runtime failure:
Address %p has no image-section
VirtualQuery failed for %d bytes at address %p
VirtualProtect failed with code 0x%x
Unknown pseudo relocation protocol version %d.
Unknown pseudo relocation bit size %d.
%d bit pseudo relocation at %p out of range, targeting %p, yielding the value %p.
runtime error %d
GCC: (x86_64-posix-seh-rev0, Built by MinGW-Builds project) 13.2.0
RegCloseKey
RegCreateKeyExA
RegSetValueExA
DeleteCriticalSection
DeleteFileA
EnterCriticalSection
GetLastError
InitializeCriticalSection
LeaveCriticalSection
SetUnhandledExceptionFilter
Sleep
TlsGetValue
```

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.

This PC > Local Disk (C:) > Tools > yara

Name	Date modified	Type	Size
malicious_downloader.yar	5/16/2025 2:12 AM	YAR File	2 KB
malware.exe	5/13/2025 2:01 PM	Application	62 KB
yara64.exe	9/10/2024 5:29 AM	Application	2,362 KB
yarac64.exe	9/10/2024 5:29 AM	Application	2,309 KB
yara-v4.5.2-2326-win64 (1).zip	5/15/2025 1:51 AM	ZIP File	2,180 KB

```
FLARE-VM 05/16/2025 02:15:15
PS C:\Users\purna\malware.rep > CD C:\Tools\yara
FLARE-VM 05/16/2025 02:17:47
PS C:\Tools\yara > .\yara64.exe malicious_downloader.yar malware.exe
>>
malicious_downloader_mingw malware.exe
```

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:

- | | |
|--|---|
| <ul style="list-style-type: none"> ✓ Component ✓ Downloader ✓ File system ✓ Registry ✓ Static detection ✓ Dynamic detection ✓ Reverse engineering | <ul style="list-style-type: none"> ✓ Observed Behaviors ✓ Successful download via WinINet ✓ File creation and deletion ✓ New key/values in HKCU ✓ No obfuscation; flagged as suspicious ✓ File, registry, and HTTP activity confirmed ✓ All malicious logic visible in decompiled code |
|--|---|

✓ Component	✓ Observed Behaviors
✓ 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.
