RIT | Rochester Institute of Technology of Dubai

CSEC.202.601 – Reverse Engineering Fundamentals

Homework 01: Basic Static Analysis

"Us vs. the CAT"

By: Ayush Shivane Gowda, Nikita Astionov, Syed Shayan Ali, Youssef Elgayar

Instructor: Dr. Emad Abu Khousa

Date of Submission: 21/02/2024

Table of Contents

1. Comprehensive Report

Introduction:

In this assignment, you've had the opportunity to dive into the world of basic static analysis. Static analysis involves dissecting and understanding an executable file without executing it. It's a crucial skill in the field of cybersecurity and low-level programming.

Please take a moment to reflect on your journey in this assignment:

- Basic Static Analysis: Explain what basic static analysis is and why it's important in the realm of cybersecurity and low-level programming. Share the key techniques you've used to analyze the provided executable.

  Basic static analysis is a key technique that is used in the cybersecurity industry. Basic static analysis is a technique in which software source code is analyzed without actually running the code. This allows for the examination of various malicious software, helping identify what type of risk it may have within it. Apart from the identification and migration of malicious software, static analysis enables us to detect bugs early in the development of a certain program. This address of issues at the beginning stage of software development allows for more cost-efficient development, as the earlier the bug is detected the less damage it would produce as the development goes on. All this implementation of static analysis can be scaled through the usage of certain automation tools, making this cybersecurity technique a perfect and efficient way to mitigate and prevent security risks and vulnerabilities in the realm of cybersecurity and low-level programming tasks. (Gillis, 2020) We have implemented various tools that have assisted us in the analysis of the provided executable, some of these tools included IDA, which was used to disassemble the malware with the purpose of analyzing its binary code. Other prominent tools we implemented were Resource Hacker and Pstudio which allowed us to further analyze the malware to find any integrated resources and to further confirm the findings from previous tools that have been used. Also, virus tool was used to identify overall information about the malware. All the tools were crucial in assisting us in the identification and overall analysis of the malware. [1]

- Skill Improvement: Describe the progress you've made in improving your skills related to basic static analysis throughout this assignment. Highlight any challenges you encountered and how you overcame them.

This assignment has enabled us to expand our knowledge of basic static analysis and at the same time test the skills that have been already developed by us. Specifically, the implementation of various tools for static analysis such as pstudio and Virustotal have widened our knowledge of how to implement tools that are new to us to a real-life problem, teaching us how to adapt to various scenarios in which different mitigation and identification techniques are used. Although this assignment seemed as simple task at first, as we progressed, we faced various challenges that slowed down our progress. One of those challenges was the problem with the installation which we faced through the process of starting up resource hacker on Kali Linux. We have overcome this problem by using online resources such as ChatGPT that have provided us with the linux commands that have assisted in successfully running and setting up these applications.

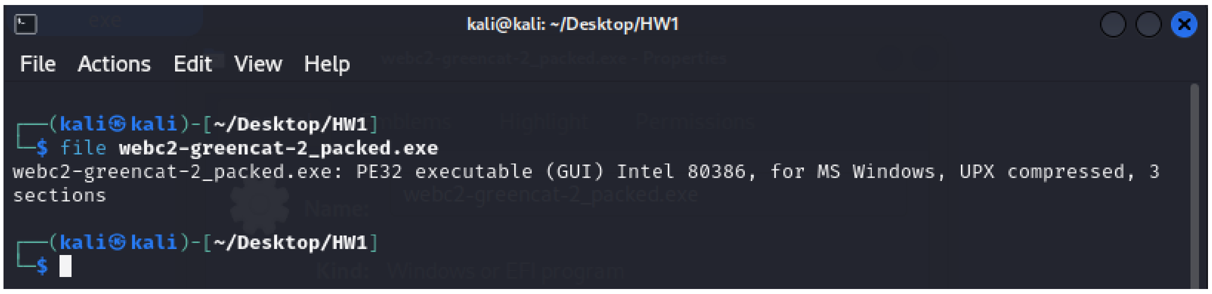
- Course Learning Outcomes (CLOs): Identify which Course Learning Outcomes (CLOs) are targeted by this homework. Specifically, discuss how the skills you've developed in static analysis align with the CLOs outlined for this course.

  This homework is targeted by a specific Course Learning Outcome (CLO), in particular, it is the 3rd course learning outcome. The 3rd CLO is "Using debuggers to trace code execution and analyze the behaviour of unknown executables". This CLO was clearly implemented in this homework assignment as we were given an unknown malware in .exe format, we have then used various debugging software that enabled us to perform static analysis on the malware. These skills that we have used in the assignment of static analysis are aligned with the 3rd CLO as we have gained knowledge in using various debugging software's to analyse the suspicious program.

2. Basic Static Analysis tasks

a) Indication of Packed or Obfuscated File

1- **Identify File Type:**
The provided malware binary is an executable file for MS Windows.



```
┌──(kali㉿kali)-[~/Desktop/HW1]
└─$ file webc2-greencat-2_packed.exe
webc2-greencat-2_packed.exe: PE32 executable (GUI) Intel 80386, for MS Windows, UPX compressed, 3
sections

┌──(kali㉿kali)-[~/Desktop/HW1]
└─$ 
```

Figure 1 – File type of the malware binary

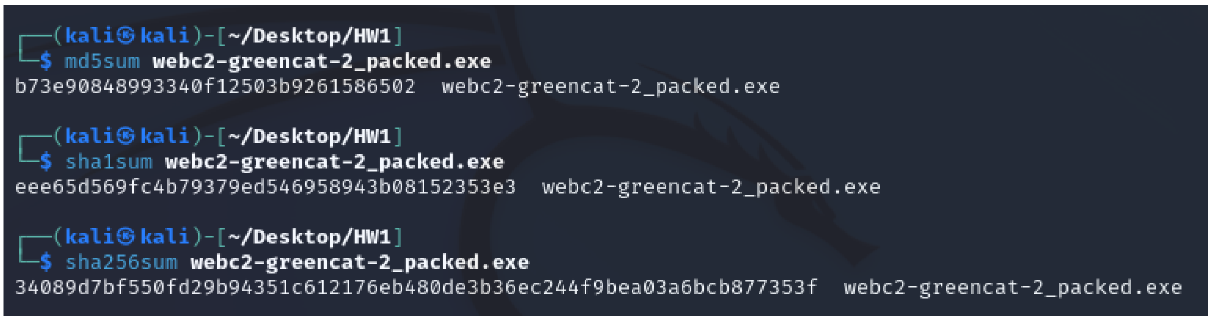
2- **Determine File Fingerprint:**

The file fingerprints for the malware binary are:

MD5: b73e90848993340f12503b9261586502

SHA-1: eee65d569fc4b79379ed546958943b08152353e3

SHA-256: 34089d7bf550fd29b94351c612176eb480de3b36ec244f9bea03a6bcb877353f

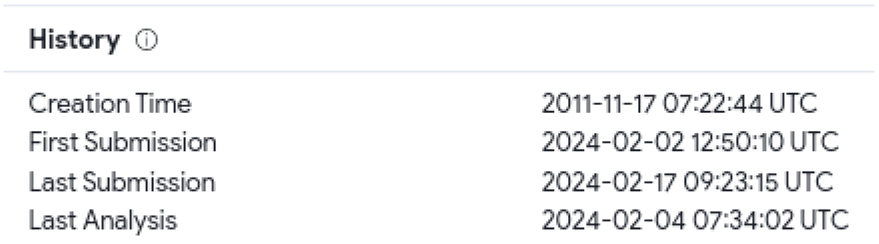


```
┌──(kali㉿kali)-[~/Desktop/HW1]
└─$ md5sum webc2-greencat-2_packed.exe
b73e90848993340f12503b9261586502  webc2-greencat-2_packed.exe

┌──(kali㉿kali)-[~/Desktop/HW1]
└─$ sha1sum webc2-greencat-2_packed.exe
eee65d569fc4b79379ed546958943b08152353e3  webc2-greencat-2_packed.exe

┌──(kali㉿kali)-[~/Desktop/HW1]
└─$ sha256sum webc2-greencat-2_packed.exe
34089d7bf550fd29b94351c612176eb480de3b36ec244f9bea03a6bcb877353f  webc2-greencat-2_packed.exe
```

Figure 2 – File Fingerprints for the binary

3- **VirusTotal Analysis:**

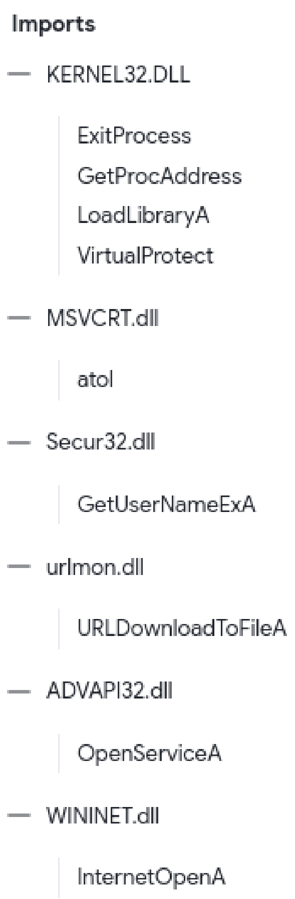
a. The malware binary was first created on 17th November 2011 at 07:22:44 UTC.
b. The first appearance of this packed version was on 2nd February 2024 at 12:50:10 UTC.
c. The malware binary has a total of 6 dynamically linked libraries and a total of 9 imported Windows functions.



History

Creation Time	2011-11-17 07:22:44 UTC
First Submission	2024-02-02 12:50:10 UTC
Last Submission	2024-02-17 09:23:15 UTC
Last Analysis	2024-02-04 07:34:02 UTC

Figure 3 – Information on the history of the file



Imports

— KERNEL32.DLL

ExitProcess
GetProcAddress
LoadLibraryA
VirtualProtect

— MSVCRT.dll

atol

— Secur32.dll

GetUserNameExA

— urlmon.dll

URLDownloadToFileA

— ADVAPI32.dll

OpenServiceA

— WININET.dll

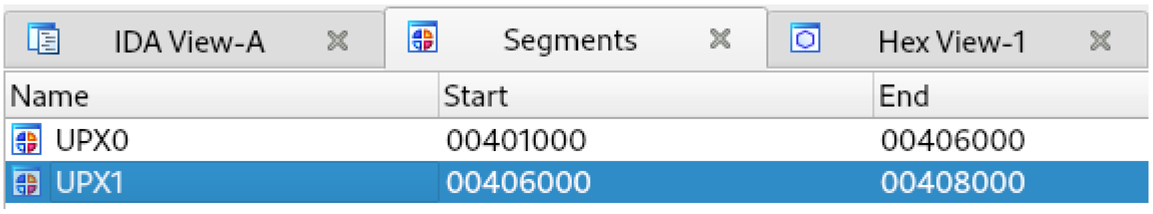
InternetOpenA

Figure 4 – Dynamically linked libraries and imported Windows functions.

4- **IDA Free Analysis:**

a. There are a total of 2 segments in the malware binary.
b. The segments are UPX0 and UPX1.
c. The starting and ending memory addresses for both segments are:

Segment	Starting address	Ending address
UPX0	00401000	00406000
UPX1	00406000	00408000

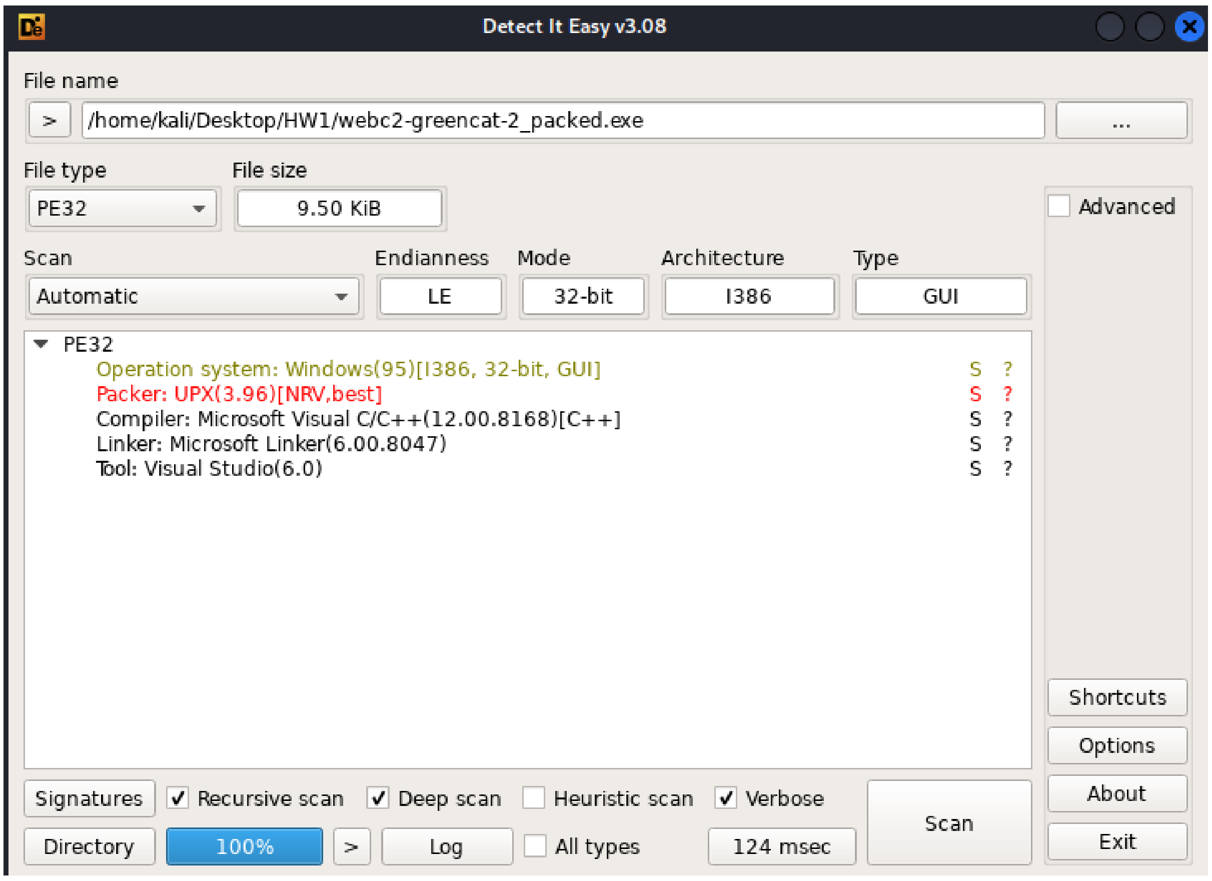


Name	Start	End
UPX0	00401000	00406000
UPX1	00406000	00408000

Figure 5 – Segments in the malware binary seen in IDA.

5- **Identify the Packing Tool:**

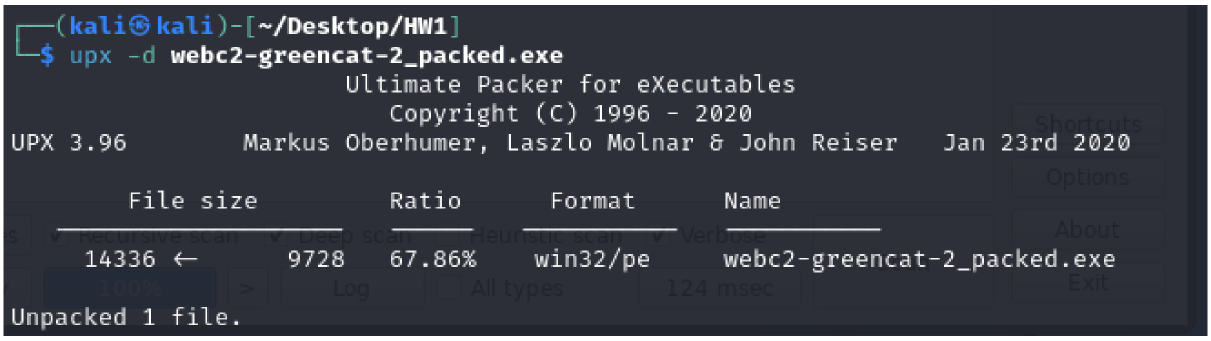
The malware binary was packed using the UPX (Ultimate Packer for eXecutables) tool.



```
PE32
    Operation system: Windows(95)[I386, 32-bit, GUI]      S  ?
    Packer: UPX(3.96)[NRV,best]                            S  ?
    Compiler: Microsoft Visual C/C++(12.00.8168)[C++]      S  ?
    Linker: Microsoft Linker(6.00.8047)                    S  ?
    Tool: Visual Studio(6.0)                               S  ?
```

Figure 6 – Detect It Easy checking for Packing/Obfuscation

6- **Unpack the Binary:**
The binary is unpacked using the command “upx -d webc2-greencat-2_packed.exe”.

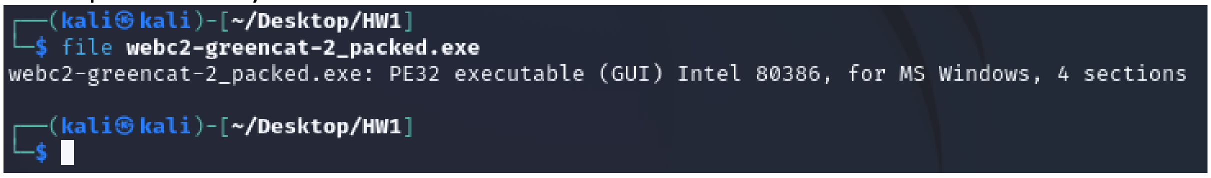


```
┌──(kali㉿kali)-[~/Desktop/HW1]
└─$ upx -d webc2-greencat-2_packed.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     14336 <-      9728   67.86%    win32/pe     webc2-greencat-2_packed.exe

Unpacked 1 file.
```

Figure 7 – Unpacking the binary.

b) Hashing and analysis using Online Antivirus Portals

1- **Identify File Type:**
The unpacked binary is an executable file for MS Windows.



```
┌──(kali㉿kali)-[~/Desktop/HW1]
└─$ file webc2-greencat-2_packed.exe
webc2-greencat-2_packed.exe: PE32 executable (GUI) Intel 80386, for MS Windows, 4 sections

┌──(kali㉿kali)-[~/Desktop/HW1]
└─$ 
```

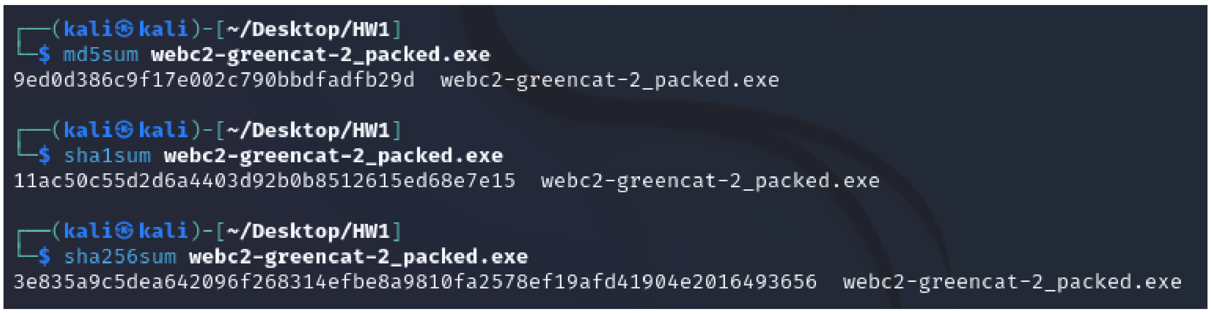
Figure 8 – File type of the unpacked binary

2- **Determine File Fingerprint:**
The file fingerprints for the unpacked binary are:

MD5: 9ed0d386c9f17e002c790bbdfadfb29d

SHA-1: 11ac50c55d2d6a4403d92b0b8512615ed68e7e15

SHA-256: 3e835a9c5dea642096f268314efbe8a9810fa2578ef19afd41904e2016493656



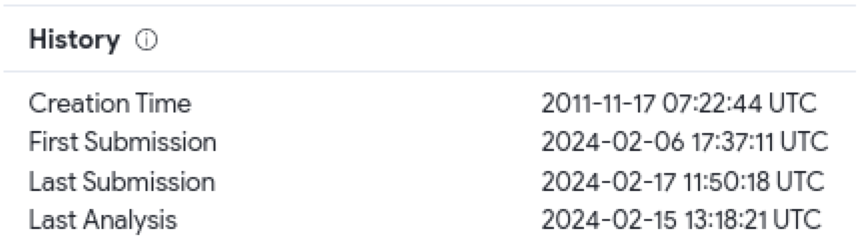
```
┌──(kali㉿kali)-[~/Desktop/HW1]
└─$ md5sum webc2-greencat-2_packed.exe
9ed0d386c9f17e002c790bbdfadfb29d  webc2-greencat-2_packed.exe

┌──(kali㉿kali)-[~/Desktop/HW1]
└─$ sha1sum webc2-greencat-2_packed.exe
11ac50c55d2d6a4403d92b0b8512615ed68e7e15  webc2-greencat-2_packed.exe

┌──(kali㉿kali)-[~/Desktop/HW1]
└─$ sha256sum webc2-greencat-2_packed.exe
3e835a9c5dea642096f268314efbe8a9810fa2578ef19afd41904e2016493656  webc2-greencat-2_packed.exe
```

Figure 9 – File Fingerprints for the unpacked binary

3- **VirusTotal Analysis:**

a. The unpacked binary was first created on 17th November 2011 at 07:22:44 UTC.
a. The debut of this unpacked binary was on 6th February 2024 at 17:37:11 UTC.

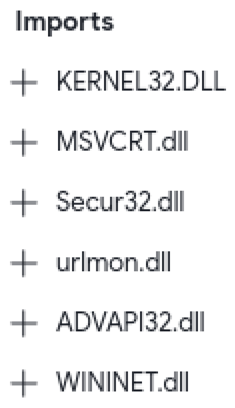
b. The unpacked binary has a total of 6 dynamically linked libraries and a total of 77 imported Windows functions. The unpacked binary reveals additional functionalities or hidden libraries that were not revealed when the binary was packed.



History ⓘ

Creation Time	2011-11-17 07:22:44 UTC
First Submission	2024-02-06 17:37:11 UTC
Last Submission	2024-02-17 11:50:18 UTC
Last Analysis	2024-02-15 13:18:21 UTC

Figure 10 – Information on the history of the file



Imports

+ KERNEL32.DLL
+ MSVCRT.dll
+ Secur32.dll
+ urlmon.dll
+ ADVAPI32.dll
+ WININET.dll

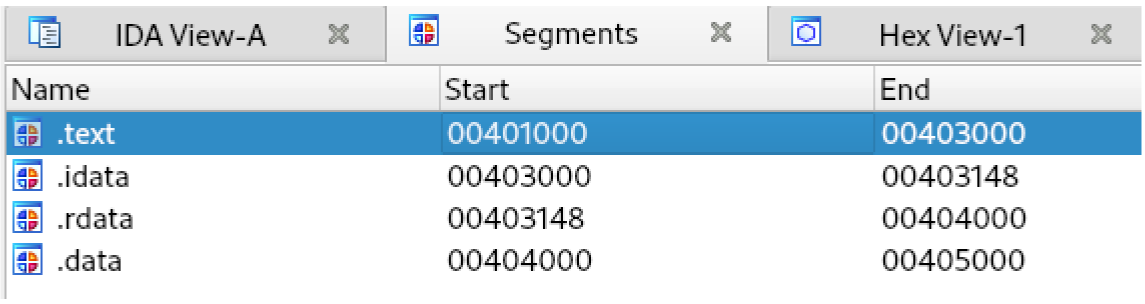
Figure 11 – Dynamically linked libraries and imported Windows functions.

4- **IDA Free Analysis:**

a. There are a total of 4 segments in the unpacked binary.
b. The segments are: ".text", ".idata", ".rdata", ".data".
c. The starting and ending memory addresses for the segments are:

Segments	Starting Address	Ending Address
.text	00401000	00403000
.idata	00403000	00403148
.rdata	00403148	00404000
.data	00404000	00405000

d. After unpacking the binary file, the names of the segments are revealed. There are also 2 additional segments after unpacking. The starting and ending memory addresses of the unpacked binary and packed binary are also different.



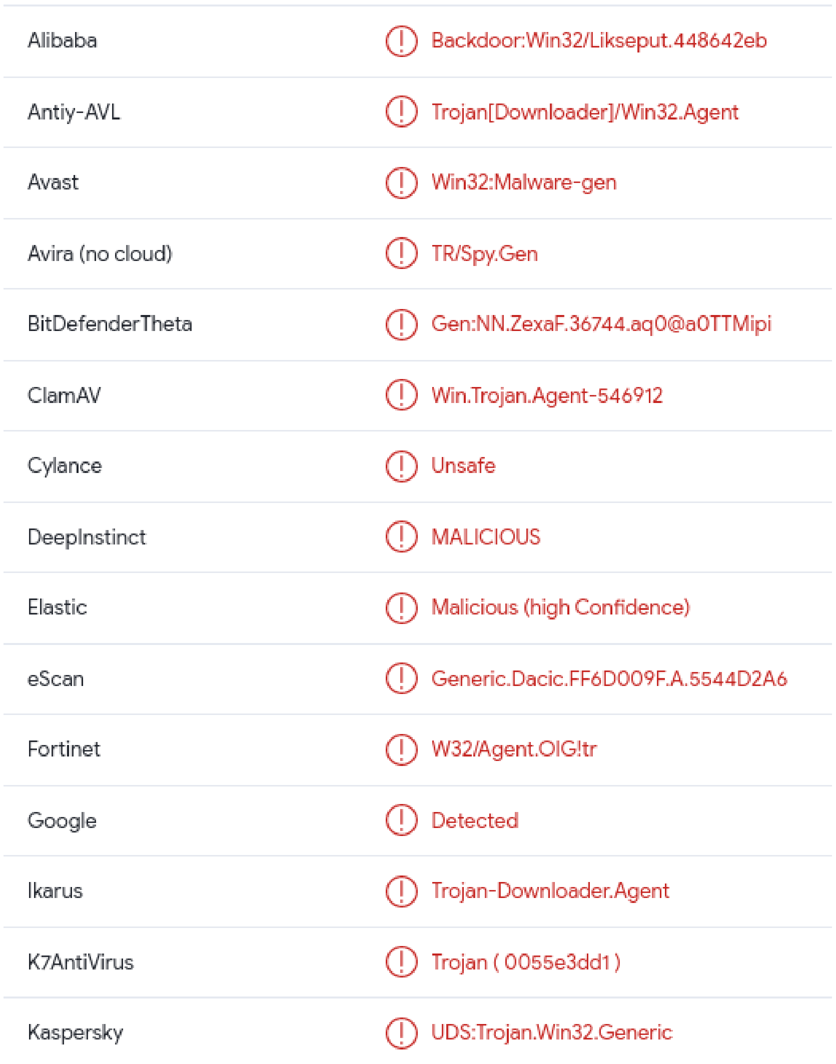
Name	Start	End
.text	00401000	00403000
.idata	00403000	00403148
.rdata	00403148	00404000
.data	00404000	00405000

Figure 12 – Segments in the unpacked binary seen in IDA.

5- Use VirusTotal to check if the executable is a known malware.

a. Out of 71 security vendors, a total of 60 security vendors flagged the "GreenCat" malware as malicious according to VirusTotal analysis.

b. The "GreenCat" malware is identified as a Windows-based backdoor malware by the security vendor Alibaba and the malware is also identified as a Trojan horse malware designed for 32-bit Windows systems by the security vendor Kaspersky.



Alibaba	Backdoor:Win32/Likseput.448642eb
Antiy-AVL	Trojan[Downloader]/Win32.Agent
Avast	Win32:Malware-gen
Avira (no cloud)	TR/Spy.Gen
BitDefenderTheta	Gen:NN.ZexaF.36744.aq0@a0TTMipi
ClamAV	Win.Trojan.Agent-546912
Cylance	Unsafe
DeepInstinct	MALICIOUS
Elastic	Malicious (high Confidence)
eScan	Generic.Dacic.FF6D009F.A.5544D2A6
Fortinet	W32/Agent.OIG!tr
Google	Detected
Ikarus	Trojan-Downloader.Agent
K7AntiVirus	Trojan (0055e3dd1)
Kaspersky	UDS:Trojan.Win32.Generic

Figure 13 – Security vendors identified by different vendors.

c) Imported Libraries and APIs

1- Compile a comprehensive list of all dynamically linked libraries (DLLs) and the APIs under each one.

Table of imported DLL and its APIs:

#	Imported DLL	APIs
1	KERNEL32.DLL	CloseHandle CreateFileA CreatePipe CreateProcessA CreateThread CreateToolhelp32Snapshot ExpandEnvironmentStringsA GetComputerNameA GetCurrentProcess GetDriveTypeA GetExitCodeProcess GetFileAttributesA GetFileSize GetLastError GetLogicalDrives GetModuleHandleA GetStartupInfoA GetSystemDirectoryA GetVolumeInformationA GetWindowsDirectoryA lstrcatA OpenProcess PeekNamedPipe Process32First Process32Next ReadFile SetCurrentDirectoryA Sleep TerminateProcess WaitForSingleObject WriteFile
2	MSVCRT.dll	??2@YAPAXI@Z ??3@YAXPAX@Z __CxxFrameHandler __getmainargs __p__commode __p__fmode __set_app_type __setusermatherr _acmdln _adjust_fdiv _controlfp

		_except_handler3 _exit _initterm _strcmpi _XcptFilter atoi atol exit memset sprintf sscanf strcat strcpy strlen strrchr strstr
3	Secur32.dll	GetUserNameExA
4	urlmon.dll	URLDownloadToFileA
5	ADVAPI32.dll	CloseServiceHandle ControlService CreateProcessAsUserA EnumServicesStatusExA OpenProcessToken OpenSCManagerA OpenServiceA StartServiceA
6	WININET.dll	HttpAddRequestHeadersA HttpOpenRequestA HttpSendRequestA InternetCloseHandle InternetConnectA InternetOpenA InternetQueryOptionA InternetReadFile InternetSetOptionA

2- Explain the significance of the **URLDownloadToFileA** API in the context of malware activity.

The "**URLDownloadToFileA**" API is a function that is primarily used for downloading files from the internet and saving them in a local file system. The significance of this API in the context of malware activity is its ability to be able to facilitate the download of malicious files and to execute additional malicious payloads or components by the malware. The malware uses the "**URLDownloadToFileA**" API to periodically check for updates of new versions of the malware from a remote server. Malware can avoid detection from antivirus software and intrusion detection systems by minimizing footprints and only downloading files when needed. [2]

d) Treasure Hunting and String Analysis of Executable

1- Process and Service Manipulation Analysis:
The strings that suggest the malware has capabilities to alter or control system processes and services are:

"TerminateProcess", **"CreateProcessA"**, **"OpenProcess"**, **"CreateProcessAsUserA"**, **"OpenProcessToken"**, **"CloseServiceHandle"**, **"EnumServicesStatusExA"**, **"OpenSCManagerA"**, **"ControlService"**, **"OpenServiceA"**, **"StartServiceA"**. [3],[4]

2- Networking Functionality:
The strings that show the malware's ability to connect to the internet are:

"HttpAddRequestHeadersA", **"HttpOpenRequestA"**, **"InternetConnectA"**, **"InternetSetOptionA"**, **"InternetOpenA"**, **"InternetQueryOptionA"**, **"HttpSendRequestA"**, **"InternetReadFile"**, **"InternetCloseHandle"**, **"URLDownloadToFileA"**. [5]

3- Command and Control Detection:
The strings that resemble commands for controlling an infected computer are:

"shell", **"start"**, **"whoami"**, **"pidrun"**, **"geturl"**, **"\\tasks"**, **"Computer:"**, **"list </p|/s|/d>"**, **"kill </p|/s> <pid|ServiceName>"**, **"getf/putf FileName <N>"**, **"exit"**, **"OK!"**, **"start </p|/s> <filename|ServiceName>"**, **"GetUrl URL FileName"**.

4- Error Handling and Debugging Messages:
The strings suggesting error handling and debugging features are:

"GetLastError", **"__CxxFrameHandler"**, **"_XcptFilter"**, **"_acmdln"**, **"__getmainargs"**, **"_initterm"**, **"__setusermatherr"**, **"_adjust_fdiv"**, **"__p__commode"**, **"__p__fmode"**, **"__set_app_type"**, **"_except_handler3"**.

5- File Manipulation:
The strings that hint at the malware creating, reading, or writing files are:

"CreatePipe", **"WriteFile"**, **"ReadFile"**, **"PeekNamedPipe"**, **"GetFileAttributesA"**, **"CreateFileA"**, **"URLDownloadToFileA"**.

The potential actions behind these actions are the purpose of file execution, data theft, establishing persistence in the system, data exfiltration, configurations, inter-process communication, and file manipulation.

6- System and User Information Reconnaissance:
The strings used by the malware to collect information about the system, or its users are:

“**GetComputerNameA**”, “**GetWindowsDirectoryA**”, “**GetSystemDirectoryA**”, “**ExpandEnvironmentStringsA**”, “**GetVolumeInformationA**”, “**GetDriveTypeA**”, “**GetLogicalDrives**”, “**GetUserNameExA**”.

7- Hardcoded Paths and Commands:
The strings that would enable the malware to access the command prompt directly are:

“**shell**”, “**cmd.exe**”, “**exit**”.

8- DLL Interaction Analysis:
The Dynamic Link Libraries (DLLs) identified from the malware strings are:

“**KERNEL32.dll**”, “**MSVCRT.dll**”, “**WININET.dll**”, “**ADVAPI32.dll**”, “**urlmon.dll**”, “**Secur32.dll**”.

The “**KERNEL32.dll**” provides core functions for memory management, process and thread creation, and various system services. The “**MSVCRT.dll**” provides standard C library functions. The “**WININET.dll**” provides functions for internet-related operations. The “**ADVAPI32.dll**” provides functions related to advanced system services and security. The “**urlmon.dll**” provides functions for URL monikers and asynchronous pluggable protocols. The “**Secur32.dll**” provides security functions related to authentication and secure communication.

9- Custom User-Agent Strings:

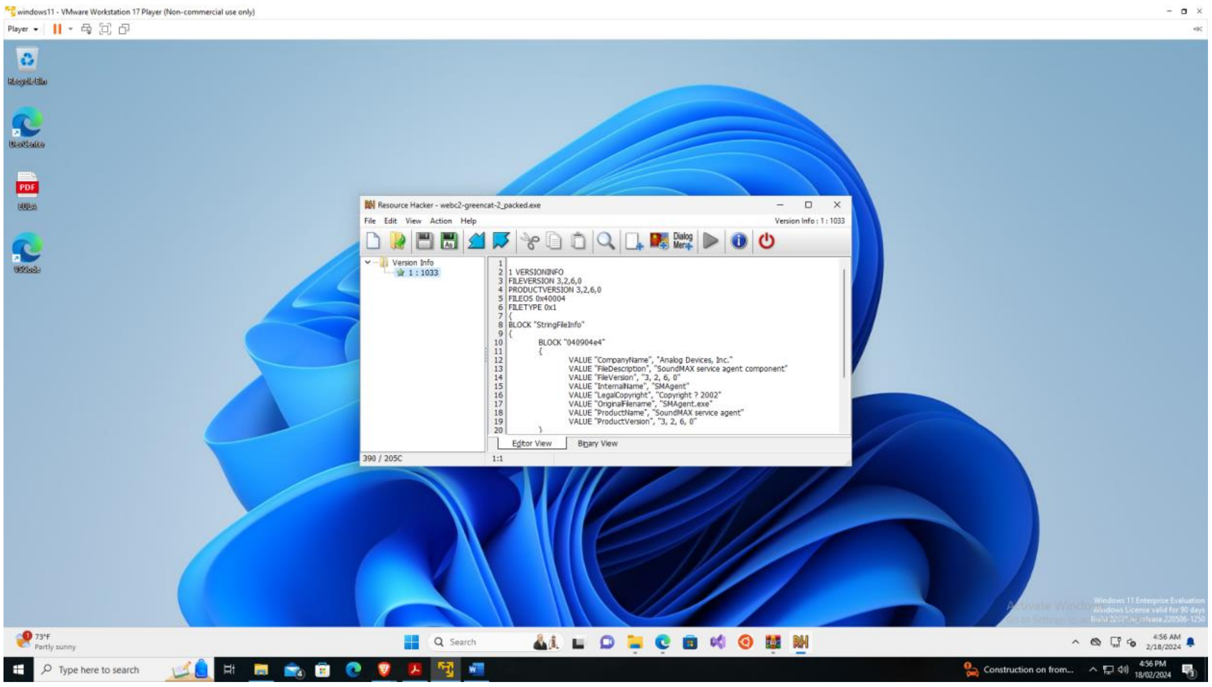
The custom User-Agent strings used by the malware to mimic web browsers are:
“**Mozilla/5.0**”, “**Mozilla/4.0**”.

10- Internet Communications:

In the malware strings, there were found to be no direct web address, URLs, or IP addresses indicating specific communication targets.

e) Discover Resources using Resource Hacker

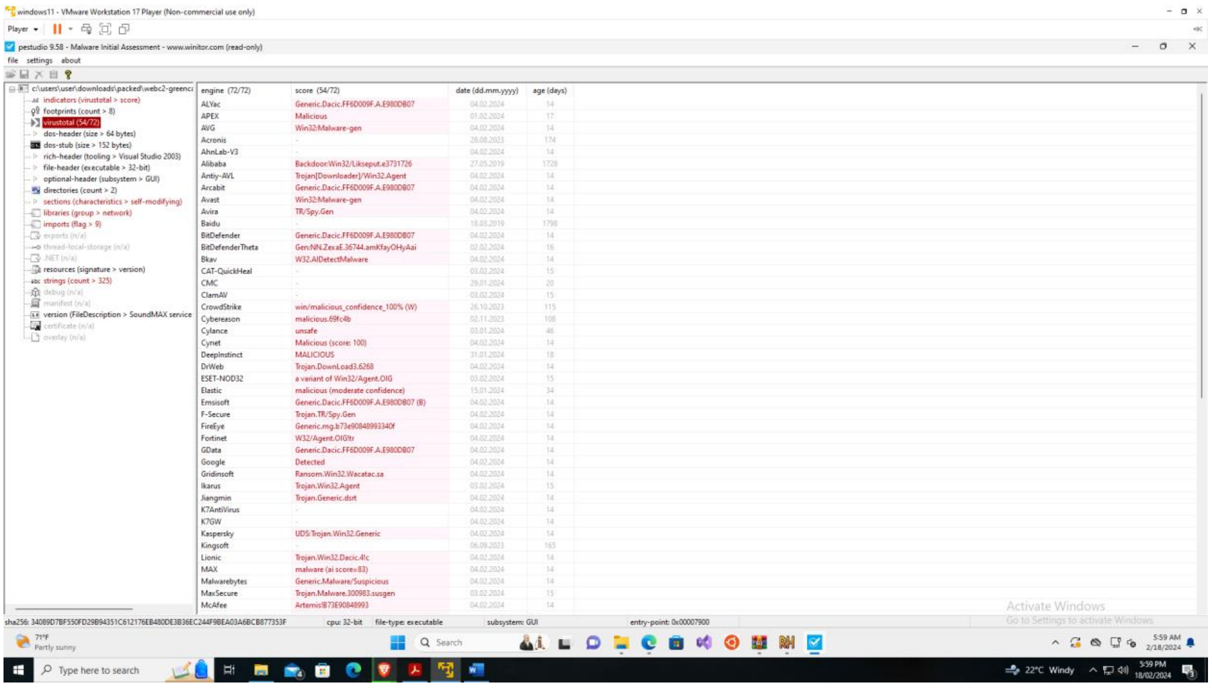
No suspicious integrated resources within the provided executable file. Only resource being used is the Version info.


windows11 - VMware Workstation 17 Player (Non-commercial use only)
Resource Hacker - webc2-greencat-2_packed.exe
File
Edit
View
Action
Help
Version Info : 1 : 1033
Version Info
1 : 1033
Editor View
Binary View
390 / 205C
1:1

```

1 VERSIONINFO
FILEVERSION 3,2,6,0
PRODUCTVERSION 3,2,6,0
FILEOS 0x40004
FILETYPE 0x1
{
BLOCK "StringFileInfo"
{
	BLOCK "040904e4"
	{
		VALUE "CompanyName", "Analog Devices, Inc."
		VALUE "FileDescription", "SoundMAX service agent component"
		VALUE "FileVersion", "3, 2, 6, 0"
		VALUE "InternalName", "SMAgent"
		VALUE "LegalCopyright", "Copyright ? 2002"
		VALUE "OriginalFilename", "SMAgent.exe"
		VALUE "ProductName", "SoundMAX service agent"
		VALUE "ProductVersion", "3, 2, 6, 0"
	}
```

f) Use Pestudio as a comprehensive tool

A total of 54/72 security vendors flagged the malware, an amount less than what we found previously during our investigation (which was 60).

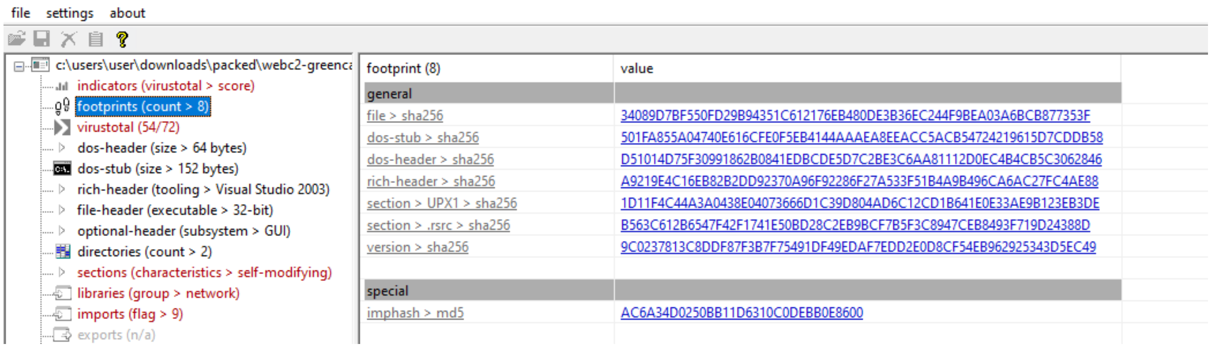

windows11 - VMware Workstation 17 Player (Non-commercial use only)
pestudio 9.58 - Malware Initial Assessment - www.winitor.com (read-only)
file
settings
about

c:\users\user\downloads\packed\webc2-greenc
- indicators (virustotal > score)
- footprints (count > 8)
- virustotal (54/72)
- dos-header (size > 64 bytes)
- dos-stub (size > 152 bytes)
- rich-header (tooling > Visual Studio 2003)
- file-header (executable > 32-bit)
- optional-header (subsystem > GUI)
- directories (count > 2)
- sections (characteristics > self-modifying)
- libraries (group > network)
- imports (flag > 9)
- exports (n/a)
- thread-local-storage (n/a)
- .NET (n/a)
- resources (signature > version)
- strings (count > 325)
- debug (n/a)
- manifest (n/a)
- version (FileDescription > SoundMAX service
- certificate (n/a)
- overlay (n/a)

engine (72/72)	score (54/72)	date (dd.mm.yyyy)	age (days)
ALYac	Generic.Dacic.FF6D009F.A.E980DB07	04.02.2024	14
APEX	Malicious	01.02.2024	17
AVG	Win32:Malware-gen	04.02.2024	14
Acronis	-	28.08.2023	174
AhnLab-V3	-	04.02.2024	14
Alibaba	Backdoor:Win32/Likseput.e3731726	27.05.2019	1728
Antiy-AVL	Trojan[Downloader]/Win32.Agent	04.02.2024	14
Arcabit	Generic.Dacic.FF6D009F.A.E980DB07	04.02.2024	14
Avast	Win32:Malware-gen	04.02.2024	14
Avira	TR/Spy.Gen	04.02.2024	14
Baidu	-	18.03.2019	1798
BitDefender	Generic.Dacic.FF6D009F.A.E980DB07	04.02.2024	14
BitDefenderTheta	Gen:NN.ZexaE.36744.amKfayOHyAai	02.02.2024	16
Bkav	W32.AIDetectMalware	04.02.2024	14
CAT-QuickHeal	-	03.02.2024	15
CMC	-	29.01.2024	20
ClamAV	-	03.02.2024	15
CrowdStrike	win/malicious_confidence_100% (W)	26.10.2023	115
Cybereason	malicious.69fc4b	02.11.2023	108
Cylance	unsafe	03.01.2024	46
Cynet	Malicious (score: 100)	04.02.2024	14
DeepInstinct	MALICIOUS	31.01.2024	18
DrWeb	Trojan.DownLoad3.6268	04.02.2024	14
ESET-NOD32	a variant of Win32/Agent.OIG	03.02.2024	15
Elastic	malicious (moderate confidence)	15.01.2024	34
Emsisoft	Generic.Dacic.FF6D009F.A.E980DB07 (B)	04.02.2024	14
F-Secure	Trojan.TR/Spy.Gen	04.02.2024	14
FireEye	Generic.mg.b73e90848993340f	04.02.2024	14
Fortinet	W32/Agent.OIG!tr	04.02.2024	14
GData	Generic.Dacic.FF6D009F.A.E980DB07	04.02.2024	14
Google	Detected	04.02.2024	14
Gridinsoft	Ransom.Win32.Wacatac.sa	04.02.2024	14
Ikarus	Trojan.Win32.Agent	03.02.2024	15
Jiangmin	Trojan.Generic.dsrt	04.02.2024	14
K7AntiVirus	-	04.02.2024	14
K7GW	-	04.02.2024	14
Kaspersky	UDS:Trojan.Win32.Generic	04.02.2024	14
Kingsoft	-	06.09.2023	165
Lionic	Trojan.Win32.Dacic.4!c	04.02.2024	14
MAX	malware (ai score=83)	04.02.2024	14
Malwarebytes	Generic.Malware/Suspicious	04.02.2024	14
MaxSecure	Trojan.Malware.300983.susgen	03.02.2024	15
McAfee	Artemis!B73E90848993	04.02.2024	14

sha256: 34089D7BF550FD29B94351C612176EB480DE3B36EC244F9BEA03A6BCB877353F   cpu: 32-bit   file-type: executable   subsystem: GUI   entry-point: 0x00007900

The sha256 hash of the file matches with what we found previously.

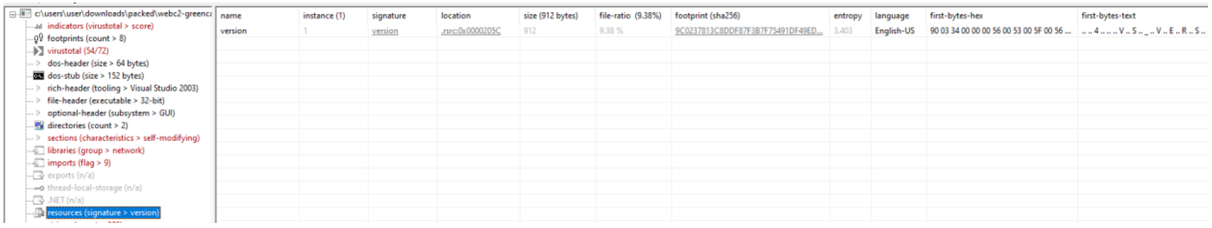


file settings about

- c:\users\user\downloads\packed\webc2-greenca
  - indicators (virustotal > score)
  - footprints (count > 8)
  - virustotal (54/72)
  - dos-header (size > 64 bytes)
  - dos-stub (size > 152 bytes)
  - rich-header (tooling > Visual Studio 2003)
  - file-header (executable > 32-bit)
  - optional-header (subsystem > GUI)
  - directories (count > 2)
  - sections (characteristics > self-modifying)
  - libraries (group > network)
  - imports (flag > 9)
  - exports (n/a)

footprint (8)	value
general	
file > sha256	34089D7BF550FD29B94351C612176EB480DE3B36EC244F9BEA03A6BCB877353F
dos-stub > sha256	501FA855A04740E616CFE0F5EB4144AAAEA8EEACC5ACB5472421961 5D7CDDB58
dos-header > sha256	D51014D75F30991862B0841EDBCDE5D7C2BE3C6AA81112D0EC4B4CB5C3062846
rich-header > sha256	A9219E4C16EB82B2DD92370A96F92286F27A533F51B4A9B496CA6AC27FC4AE88
section > UPX1 > sha256	1D11F4C44A3A0438E04073666D1C39D804AD6C12CD1B641E0E33AE9B123EB3DE
section > .rsrc > sha256	B563C612B6547F42F1741E50BD28C2EB9BCF7B5F3C8947CEB8493F719D24388D
version > sha256	9C0237813C8DDF87F3B7F75491DF49EDAF7EDD2E0D8CF54EB962925343D5EC49
special	
imphash > md5	AC6A34D0250BB11D6310C0DEBB0E8600

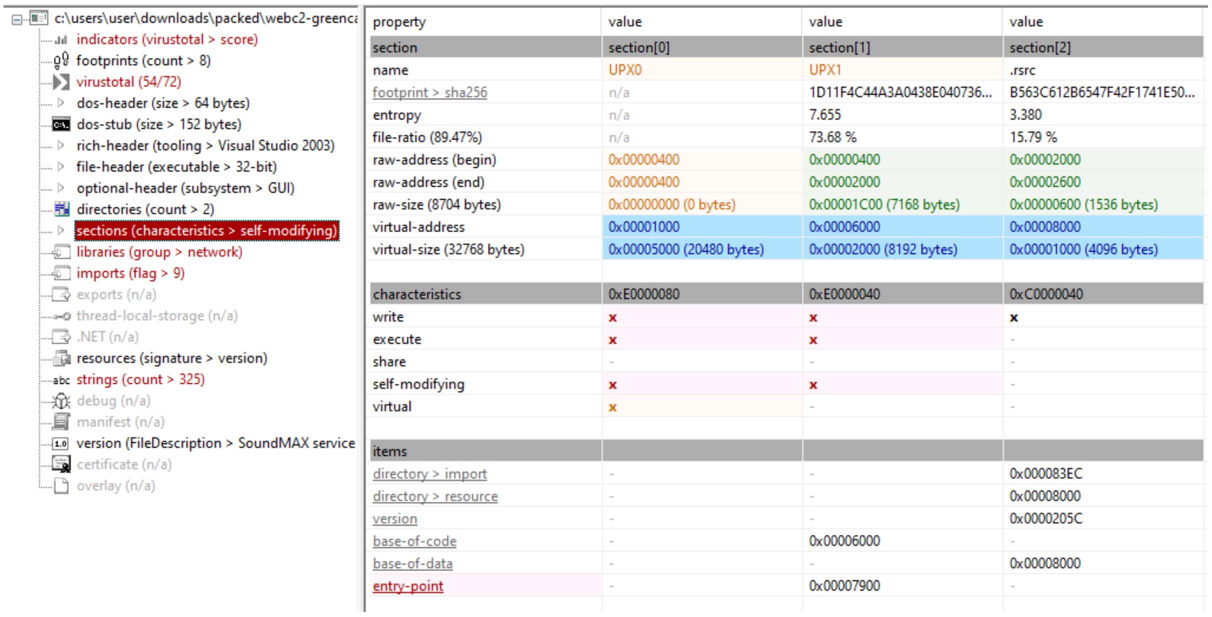
We see no resources being used besides the version which matches with our findings. The version resource itself also isn't flagged. The entropy levels are also low which means no indication of it being obfuscating.



- c:\users\user\downloads\packed\webc2-greenca
  - indicators (virustotal > score)
  - footprints (count > 8)
  - virustotal (54/72)
  - dos-header (size > 64 bytes)
  - dos-stub (size > 152 bytes)
  - rich-header (tooling > Visual Studio 2003)
  - file-header (executable > 32-bit)
  - optional-header (subsystem > GUI)
  - directories (count > 2)
  - sections (characteristics > self-modifying)
  - libraries (group > network)
  - imports (flag > 9)
  - exports (n/a)
  - thread-local-storage (n/a)
  - .NET (n/a)
  - resources (signature > version)

name	instance (1)	signature	location	size (912 bytes)	file-ratio (9.38%)	footprint (sha256)	entropy	language	first-bytes-hex	first-bytes-text
version	1	version	.rsrc:0x0000205C	912	9.38 %	9C0237813C8DDF87F3B7F75491DF49ED...	3.403	English-US	90 03 34 00 00 00 56 00 53 00 5F 00 56 ...	.. .. 4 .. .. .. V .. S .. _ .. V .. E .. R .. S ..

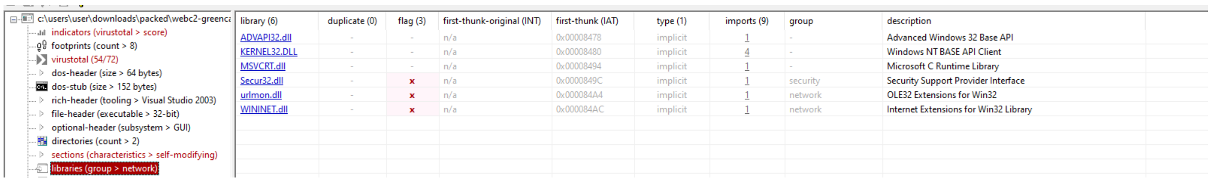
We find that Pestudio flags the characteristics of the files. UPX0 and UPX1 have write and execute permissions which are flagged. They both are also self-modifying files which is also flagged obviously.



- c:\users\user\downloads\packed\webc2-greenca
  - indicators (virustotal > score)
  - footprints (count > 8)
  - virustotal (54/72)
  - dos-header (size > 64 bytes)
  - dos-stub (size > 152 bytes)
  - rich-header (tooling > Visual Studio 2003)
  - file-header (executable > 32-bit)
  - optional-header (subsystem > GUI)
  - directories (count > 2)
  - sections (characteristics > self-modifying)
  - libraries (group > network)
  - imports (flag > 9)
  - exports (n/a)
  - thread-local-storage (n/a)
  - .NET (n/a)
  - resources (signature > version)
  - strings (count > 325)
  - debug (n/a)
  - manifest (n/a)
  - version (FileDescription > SoundMAX service
  - certificate (n/a)
  - overlay (n/a)

property	value	value	value
section	section[0]	section[1]	section[2]
name	UPX0	UPX1	.rsrc
footprint > sha256	n/a	1D11F4C44A3A0438E040736...	B563C612B6547F42F1741E50...
entropy	n/a	7.655	3.380
file-ratio (89.47%)	n/a	73.68 %	15.79 %
raw-address (begin)	0x00000400	0x00000400	0x00002000
raw-address (end)	0x00000400	0x00002000	0x00002600
raw-size (8704 bytes)	0x00000000 (0 bytes)	0x00001C00 (7168 bytes)	0x00000600 (1536 bytes)
virtual-address	0x00001000	0x00006000	0x00008000
virtual-size (32768 bytes)	0x00005000 (20480 bytes)	0x00002000 (8192 bytes)	0x00001000 (4096 bytes)
characteristics	0xE0000080	0xE0000040	0xC0000040
write	x	x	x
execute	x	x	-
share	-	-	-
self-modifying	x	x	-
virtual	x	-	-
items			
directory > import	-	-	0x000083EC
directory > resource	-	-	0x00008000
version	-	-	0x0000205C
base-of-code	-	0x00006000	-
base-of-data	-	-	0x00008000
entry-point	-	0x00007900	-

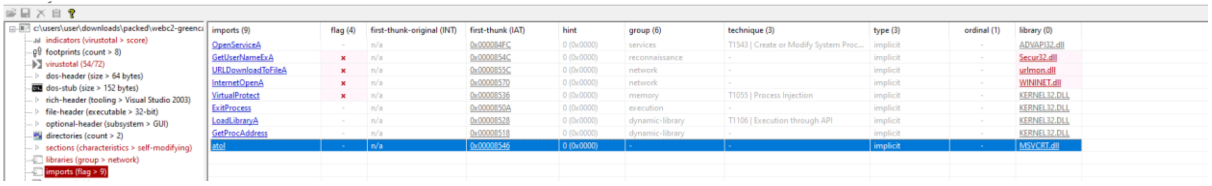
Out of the 6 libraries, 3 are flagged. They are WININET.dll, urlmonl.dll and Secur32.dll .



c:\users\user\downloads\packed\webc2-greenca
- indicators (virustotal > score)
- footprints (count > 8)
- virustotal (54/72)
- dos-header (size > 64 bytes)
- dos-stub (size > 152 bytes)
- rich-header (tooling > Visual Studio 2003)
- file-header (executable > 32-bit)
- optional-header (subsystem > GUI)
- directories (count > 2)
- sections (characteristics > self-modifying)
- libraries (group > network)

library (6)	duplicate (0)	flag (3)	first-thunk-original (INT)	first-thunk (IAT)	type (1)	imports (9)	group	description
ADVAPI32.dll	-	-	n/a	0x00008478	implicit	1	-	Advanced Windows 32 Base API
KERNEL32.DLL	-	-	n/a	0x00008480	implicit	4	-	Windows NT BASE API Client
MSVCRT.dll	-	-	n/a	0x00008494	implicit	1	-	Microsoft C Runtime Library
Secur32.dll	-	x	n/a	0x0000849C	implicit	1	security	Security Support Provider Interface
urlmon.dll	-	x	n/a	0x000084A4	implicit	1	network	OLE32 Extensions for Win32
WININET.dll	-	x	n/a	0x000084AC	implicit	1	network	Internet Extensions for Win32 Library

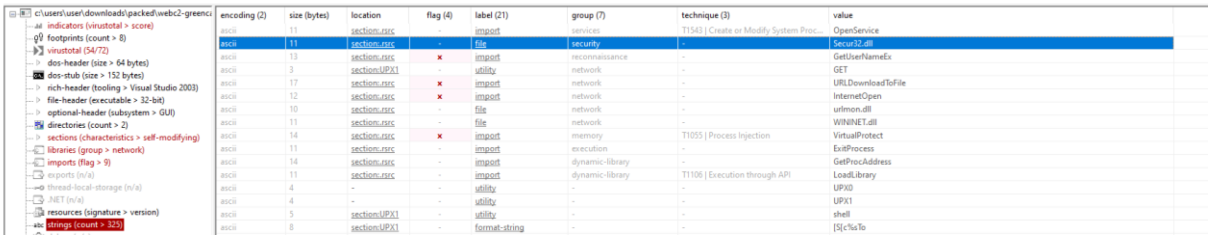
We now see the imports within the libraries where we find something interesting. Even though the library KERNEL32.dll was not flagged an import within it was. The import is called VirtualProtect and was most likely flagged because of the technique it uses which is a "Process injection". Cynet explains to us that "Process injection is a widespread defence evasion technique commonly employed within malware and fileless adversary attacks. It entails running custom code within the address space of another process. Process injection improves stealth, and some variant techniques also achieve persistence. [6]



c:\users\user\downloads\packed\webc2-greenca
- indicators (virustotal > score)
- footprints (count > 8)
- virustotal (54/72)
- dos-header (size > 64 bytes)
- dos-stub (size > 152 bytes)
- rich-header (tooling > Visual Studio 2003)
- file-header (executable > 32-bit)
- optional-header (subsystem > GUI)
- directories (count > 2)
- sections (characteristics > self-modifying)
- libraries (group > network)
- imports (flag > 9)

imports (9)	flag (4)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (6)	technique (3)	type (3)	ordinal (1)	library (0)
OpenServiceA	-	n/a	0x000084FC	0 (0x0000)	services	T1543 | Create or Modify System Proc...	implicit	-	ADVAPI32.dll
GetUserNameExA	x	n/a	0x0000854C	0 (0x0000)	reconnaissance	-	implicit	-	Secur32.dll
URLDownloadToFileA	x	n/a	0x0000855C	0 (0x0000)	network	-	implicit	-	urlmon.dll
InternetOpenA	x	n/a	0x00008570	0 (0x0000)	network	-	implicit	-	WININET.dll
VirtualProtect	x	n/a	0x00008536	0 (0x0000)	memory	T1055 | Process Injection	implicit	-	KERNEL32.DLL
ExitProcess	-	n/a	0x0000850A	0 (0x0000)	execution	-	implicit	-	KERNEL32.DLL
LoadLibraryA	-	n/a	0x00008528	0 (0x0000)	dynamic-library	T1106 | Execution through API	implicit	-	KERNEL32.DLL
GetProcAddress	-	n/a	0x00008518	0 (0x0000)	dynamic-library	-	implicit	-	KERNEL32.DLL
atol	-	n/a	0x00008546	0 (0x0000)	-	-	implicit	-	MSVCRT.dll

In the strings section we see 4 sections being flagged. One import conducts reconnaissance by getting the username. Two other imports are flagged because one can connect to the internet and the other can download through a URL. The other import shown flagged in strings section is because of the Process injection as discussed before



c:\users\user\downloads\packed\webc2-greenca
- indicators (virustotal > score)
- footprints (count > 8)
- virustotal (54/72)
- dos-header (size > 64 bytes)
- dos-stub (size > 152 bytes)
- rich-header (tooling > Visual Studio 2003)
- file-header (executable > 32-bit)
- optional-header (subsystem > GUI)
- directories (count > 2)
- sections (characteristics > self-modifying)
- libraries (group > network)
- imports (flag > 9)
- exports (n/a)
- thread-local-storage (n/a)
- .NET (n/a)
- resources (signature > version)
- strings (count > 325)

encoding (2)	size (bytes)	location	flag (4)	label (21)	group (7)	technique (3)	value
ascii	11	section:.rsrc	-	import	services	T1543 | Create or Modify System Proc...	OpenService
ascii	11	section:.rsrc	-	file	security	-	Secur32.dll
ascii	13	section:.rsrc	x	import	reconnaissance	-	GetUserNameEx
ascii	3	section:UPX1	-	utility	network	-	GET
ascii	17	section:.rsrc	x	import	network	-	URLDownloadToFile
ascii	12	section:.rsrc	x	import	network	-	InternetOpen
ascii	10	section:.rsrc	-	file	network	-	urlmon.dll
ascii	11	section:.rsrc	-	file	network	-	WININET.dll
ascii	14	section:.rsrc	x	import	memory	T1055 | Process Injection	VirtualProtect
ascii	11	section:.rsrc	-	import	execution	-	ExitProcess
ascii	14	section:.rsrc	-	import	dynamic-library	-	GetProcAddress
ascii	11	section:.rsrc	-	import	dynamic-library	T1106 | Execution through API	LoadLibrary
ascii	4	-	-	utility	-	-	UPX0
ascii	4	-	-	utility	-	-	UPX1
ascii	5	section:UPX1	-	utility	-	-	shell
ascii	8	section:UPX1	-	format-string	-	-	[S[c%sTo

3. Explore and discuss the following, using supporting evidence, screenshots, and logical reasoning:

1- **Is the Binary Malicious?** Provide a reasoned argument, supported by evidence from your analyses and the resources above. What characteristics or behaviours tip the scale towards malicious intent?

Yes, we believe the binary is malicious for several reasons. When we conducted an analysis using Virus Total, 60 cyber security vendors flagged the binary. Such a massive number of security vendors flagging the binary gives us confidence that the binary in our possession is not at all safe. The binary has characteristics that imply a malicious intent, such as being self-modifying and having execute permissions.

2- **Local or Remote Binary?** Determine the nature of the binary's operations. If it's designed to interact with remote resources, identify those resources and discuss their relevance to the malware's functionality.

Based on the libraries and the functions that have been discovered earlier in this assignment, we can tell it's likely designed for remote operations. As the binary is seen to heavily rely on functions like HttpOpenRequestA and URLDownloadToFileA to download stuff from the internet or just have these tasks performed via the internet, which strongly indicates that this malware relies on remote operation through the internet.

3- **Permanent Changes to the Hosting OS:** Investigate whether the executable is engineered to modify the system it infects permanently. Detail the specific local resources it interacts with and the nature of these interactions.

Based on the evaluations of the DLLs that have been imported and the Apis that are present, the malware has been engineered for the purpose of permanently modifying the host operating system. Through analysis, we can tell that the malware uses manipulation techniques on the system services and processes. “CreateProcessA”, “OpenProcess”, and “TerminateProcess” are some of the example functions that are implemented to manipulate the system processes. During the analysis we have also noticed functions like “URLDownloadToFileA” that indicate the relation with the internet connection of the malware, also functions like “shell”, “GetLastError” are used to permanently modify the host OS. “shell” function indicates that some execute commands and functions are used to execute processes, and the “GetLastError” function indicates the practice of debugging and error handling occurring. All these functions indicate the intent of this malware, which is to permanently modify the host operating system.

4- **Mechanisms of Change:** If the executable does introduce permanent changes, elucidate how it achieves this. The intricacies of its operations can shed light on its objectives and the threat level it poses.

   Based on our analysis of the malware we have previously concluded that this executable appears to be implanted for the purpose of permanent modification of the target system. The objectives of the malware can be analyzed through the functions that it incorporates. Functions like "EnumServicesStatusExA," and "ControlService" indicate the malware ability to control and modify crucial processes through these system manipulation techniques, further achieving resistance. Also functions like "URLDownloadToFileA" and “WriteFile” further indicate the malware’s ability to manipulate files. This potentially indicates that the malware can do various malicious tasks such as the implantation of payloads and configurations that are malicious to the target system. All these functions indicate that the malicious software is of a higher threat level, as it seems to be capable of data manipulation, and the system modification which we found to be permanent. Certain countermeasures would have to be taken to battle this malware.

4. In Dubai Courts Room - Expert Witness Simulation

Clarity and Layman's Terms:

I would say this malware is like a thief breaking into houses to steal data and abuse the systems of it, as well as putting risks in the security of the systems of the house to steal personal and financial information.

Identification of Malware Type:

We have identified this malware as a Trojan horse Because it calls itself “SOUNDMAX service agent component” which is a suspicious name.

And based on our analysis the suspicious URL could download things into the computer which will process injections and viruses into the device.

Once it has access to the device it can create threads and take control remotely, which will make it execute commands and access unauthorized data.

Independent Ransomware Attack Capability:

To know if the malware can initiate a ransomware attack without any external commands or interactions, we need to know it's libraries that are imported. And judging by the available APIs file encryption could be used with function CreateFileA to open the file and WriteFile to modify and change its content and CloseHandle to close after the encryption. But without any evidence of encryption, it’s hard to detect if the program is a ransomware. So, to make sure if the accusation is right, we would have to examine the program source code and analyze its behaviour.

Execution Method of Attack:

If the malware does not directly encrypt data it could still potentially be involved in a ransomware attack for example if the malware acts as a download file that carries encryption and sends a ransom note after encrypting the files, it would then be an opening for the hacker to potentially make it a gateway for him to apply more and more ransomware attacks.

Recommendation for Further Analysis:

Further analysis of this malware is important to understand its limits, applying both static analysis and dynamic analysis is important to know its vulnerabilities as well as the malwares behaviour in a controlled environment which could help us understand the malware and know its weaknesses.

5. References

[1] A. S. Gillis, “What is static analysis (static code analysis)?,” *WhatIs*, 31-Jul-2020. [Online]. Available: https://www.techtarget.com/whatis/definition/static-analysis-static-code-analysis.

[2] “URLDownloadToFile function,” *Microsoft.com*. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/ms775123(v=vs.85).

[3] “KERNEL32 Versions,” *Geoffchappell.com*. [Online]. Available: https://www.geoffchappell.com/studies/windows/win32/kernel32/history/index.htm?tx=51.

[4] “CloseServiceHandle function (winsvc.h),” *Microsoft.com*. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/winsvc/nf-winsvc-closeservicehandle.

[5] “Wininet.h header - Win32 apps,” *Microsoft.com*. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/wininet/.

[6] “Process injection techniques,” *Cynet*, 22-Apr-2021. [Online]. Available: https://www.cynet.com/attack-techniques-hands-on/process-injection-techniques/.

[7] Mandiant, “APT1,” *Mandiant.com*. [Online]. Available: https://www.mandiant.com/sites/default/files/2021-09/mandiant-apt1-report.pdf.

6. Bonus Exploration: Additional Tool Usage

We used several additional tools to confirm some of our findings. For example, when *ResourceHacker* didn't show us that any malicious resource was being used, we confirmed our finding by using PEview, a similar tool as *ResourceHacker*. Furthermore, we also used x64dbg a powerful debugging tool with a user-friendly GUI which helped us get a holistic perspective of the malware with its syntax highlighting and graphs.

X64dbg: https://x64dbg.com/

7. Bonus Attribution Investigation

After a very thorough investigation where we left no stone unturned, we were able to pinpoint the source of the malware. APT1, also known as People's Liberation Army Unit 61398, is Chinese hacking group that serves the ruling Chinese Communist Party. It is one of China's leading espionage units and is very dangerous with its state level backing. [7]