BFOR - 418/618 - Malware Reverse Engineering
Basic Static Analysis

Katta Sreenivasulu, Tarun Sai

September 27, 2023

Dr. Prinkle Sharma
University at Albany, SUNY

Contents

1 Introduction

In this lab, we have performed static analysis using tools like VirusTotal.com, strings, understanding DLL imports, how hash signatures are used to detect, PE Viewer and Identifier tools.

2 Static Malware Analysis

Static malware analysis involves examining and analyzing malware without executing or running it. It mainly focuses on understanding the structure, behavior, and potential threats posed by a software by analyzing its code, file attributes, metadata, and other characteristics.

Static analysis does not involve running the malware in a live environment, which helps mitigate risks associated with executing the malicious software.

Pro's:

1. **Signature Generation:**
   Static analysis can help generate signatures or patterns that can be used for detecting and identifying similar malware in the future. These signatures can be used in antivirus or intrusion detection systems.

2. **Different aspects:**
   Analysis of various aspects of malware such as file headers, strings, imports, exports and more to understand its functionality better and deduce impact.

3. **Reverse Engineering:**
   Provides the opportunity to reverse engineer the code, to understand it's flow, logic and potential vulnerabilities which it is trying to exploit.

Con's:

1. **Polymorphism:**
   Static analysis fails to provide insights on polymorphic malware, where their core functionality remains unchanged, however their signature is totally different.

2. **Evasion Techniques:**
   If the malware are using the latest evasion techniques, thereby completely obfuscating the malware, then it is hard to to detect and analyze the malware.

3 Software Used

(i) **VirusTotal.com:**
VirusTotal aggregates many antivirus products and online scan engines called Contributors. The aggregated data allows a user to check for viruses that the user's own antivirus software may have missed.

(ii) **Windows Sandbox:**
Windows Sandbox is used to maintain a lightweight desktop environment to execute windows applications in isolation. It is mainly used in this lab to perform static analysis on samples, without actually damaging the original host operating system, network or device hardware in general.

(iii) **Flare-VM:**
FLARE VM is a freely available and open sourced Windows-based security distribution designed for reverse engineers and malware analysts. FLARE VM delivers a fully configured platform with a comprehensive collection of Windows security tools such as debuggers, disassemblers (Ghidra, IDA Free), decompilers, Hex Editors (HxD), static and dynamic analysis utilities (HashCalc, PE-ID, PE-Bear, Detect It Easy, Strings Utility etc.,), network analysis and manipulation, web assessment, exploitation, vulnerability assessment applications, and many others.

4 Methodology

4.1 Analyzing Malware Files using VirusTotal.com

4.1.1 Files: Lab01.01.exe and Lab01-01.dll

1. **Upload the files to http://www.VirusTotal.com/ and view the reports. Does either file match any existing antivirus signatures?**

   Yes, it is flagged as malicious among 55 security vendors and 1 sandbox among 71 and categorized as Trojan.

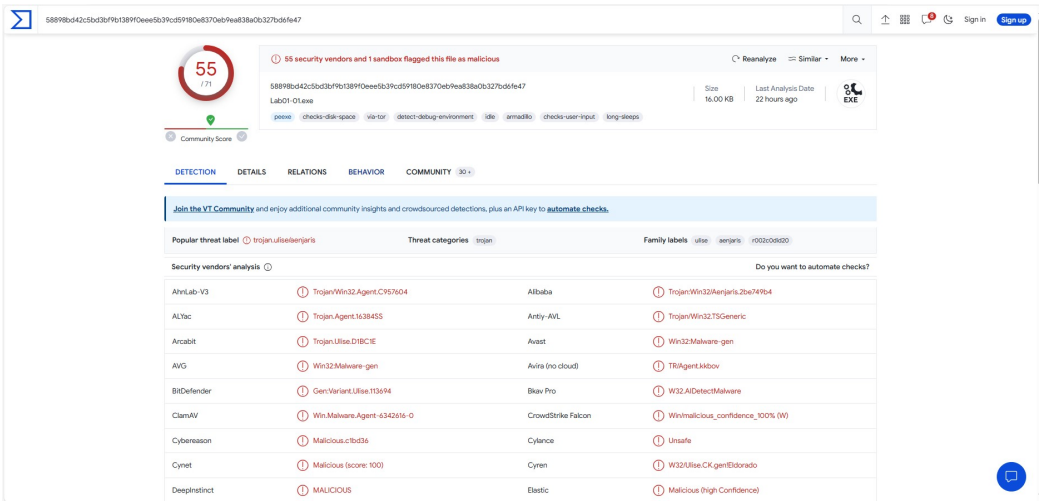

Figure 1: Lab01-01.exe VirusTotal

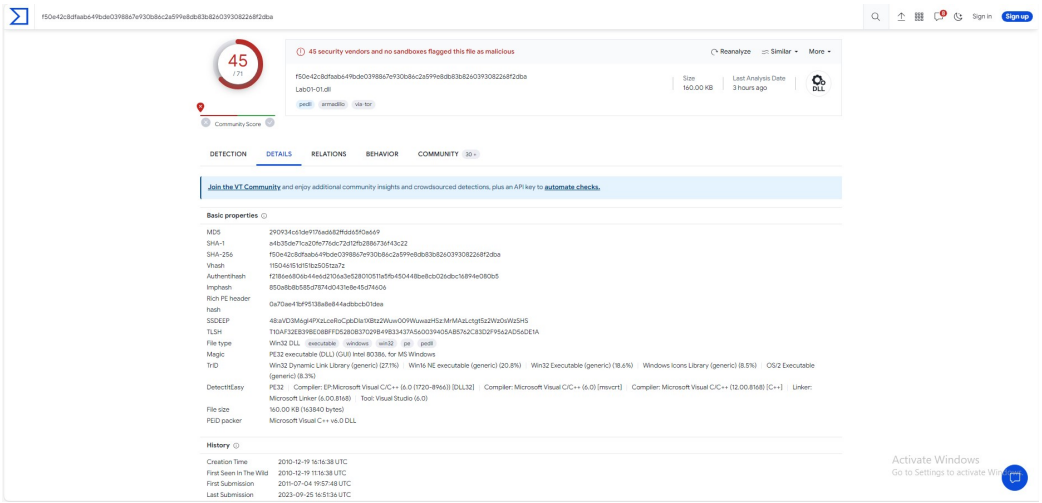

Figure 2: Lab01-01.dll VirusTotal

2. **When were these files compiled?**

   The file **Lab01-01.exe** has **Compilation Timestamp: 2010-12-19 16:16:19 UTC**
   The file **Lab01-01.dll** has **Compilation Timestamp: 2010-12-19 16:16:38 UTC**

3. **Are there any indications that either of these files is packed or obfuscated? If so, what are these indicators?**

   By observing the low number of imports from *Figure 3 and Figure 4*, it is understood that the files **Lab01-01.exe** and **Lab01-01.dll** are not packed or obfuscated.

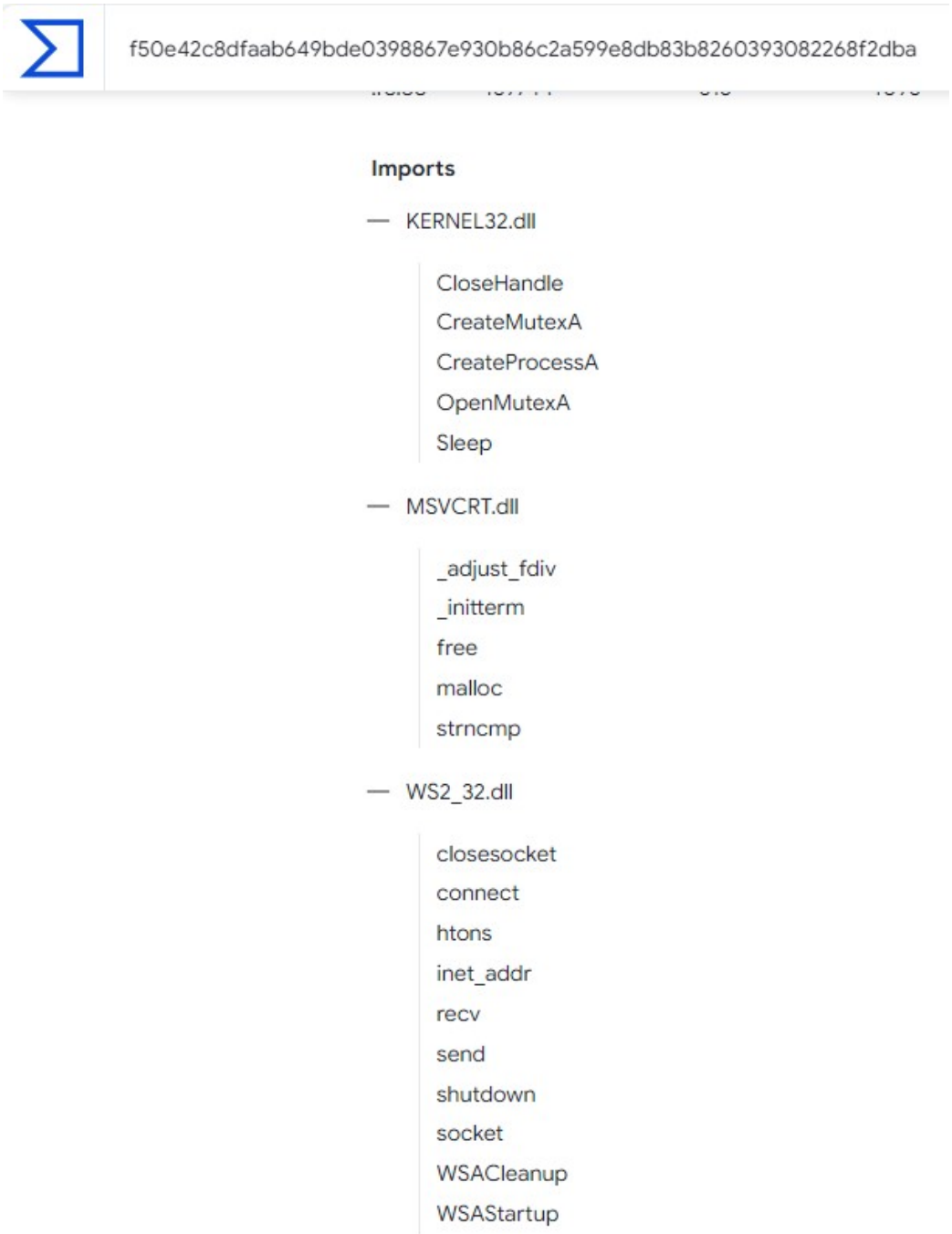

Figure 3: Lab01-01.dll imports

Figure 4: Lab01-01.exe imports

4. **Do any imports hint at what this malware does? If so, which imports are they?**

   The imports for **Lab01-01.exe** as shown in *Figure 4* has **FindNextFileA** and **FindFirstFileA** which are used to search and manipulate files like open and modifying its data and metadata.

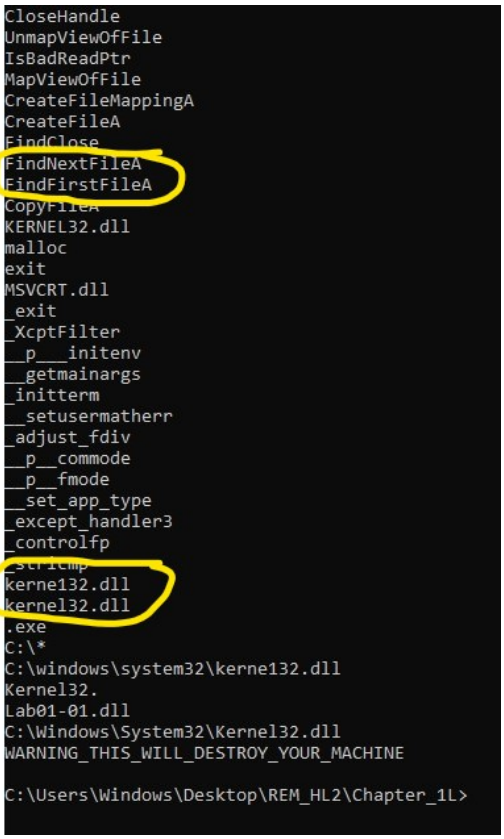

Figure 5: Strings output for Lab01-01.exe

5. **Are there any other files or host-based indicators that you could look for on infected systems?**

   As shown in the *Figure 5*, there is kerne**1**32.dll which is used to disguise itself as legitimate
   kernel32.dll. So the kerne**1**32.dll could have served as a host based indicator and needs to be analysed further.

6. **What network-based indicators could be used to find this malware on infected machines?**

   After observing *Figure 3*, it is understood that using import WS2_32[.]dll the executable has network functions like **connect, send, recv** to establish a connection, send and receive packets using a network respectively. It also has functions like **CreateProcessA** and **Sleep** which can be used to create a backdoor process.

7. **What would you guess is the purpose of these files?**

   This executable resembled similar to functionality of a backdoor, by trying to maintain a connection to manipulate host files and directories.

4.1.2 File: Lab01-04.exe

1. **Upload the Lab01-04ėxe file to http://www.VirusTotal.com/. Does it match any existing antivirus definitions?**

   Yes, it is matching with existing anti-virus definitions. From *Figure 6*, 59 out of 71 vendors and 2 sandboxes flag this executable file as malicious.

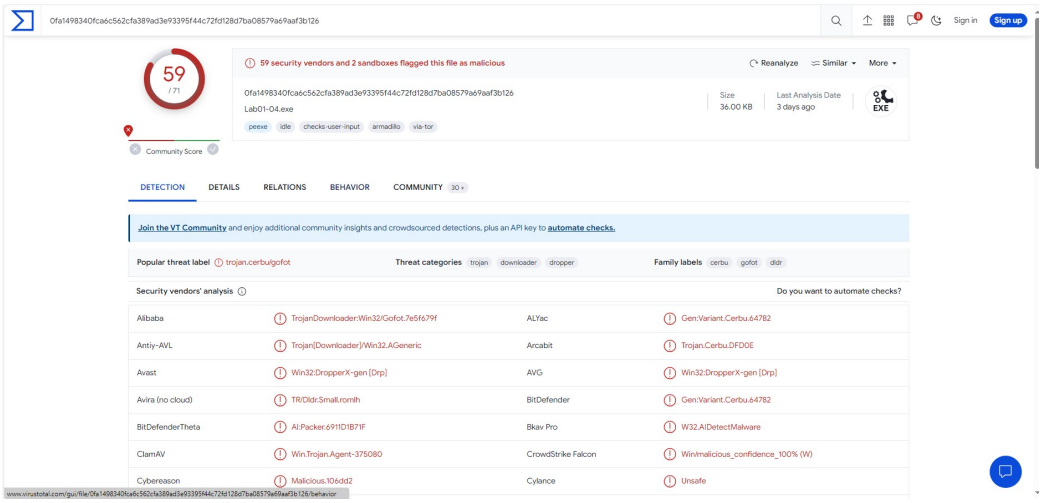

Figure 6: Lab01-04.exe VirusTotal

2. **Are there any indications that this file is packed or obfuscated? If so, what are these indicators? If the file is packed, unpack it if possible.**

   No, the file is not packed or obfuscated.

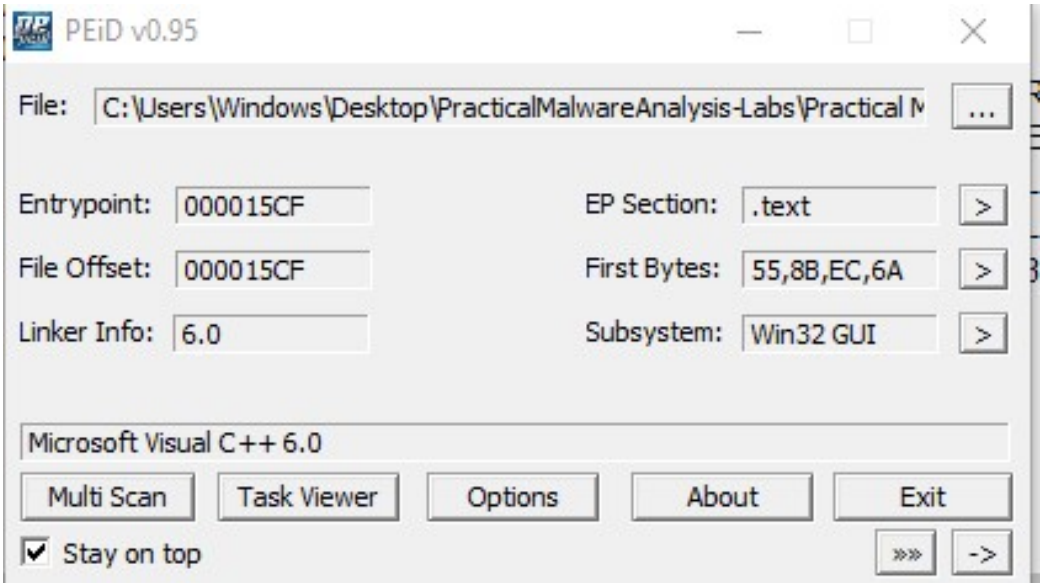

Figure 7: PEID output for Lab01-04.exe

3. **When was this program compiled?**

   The program was compiled on **30th August, 2019 Friday at 22:26:59 UTC**.

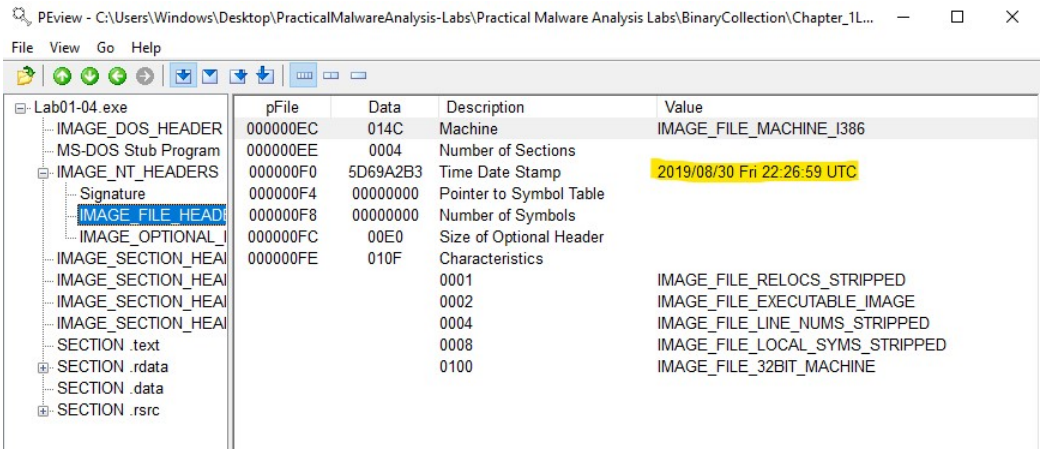

Figure 8: Lab01-04.exe Compilation Time

4. **Do any imports hint at this program's functionality? If so, which imports are they and what do they tell you?**

   Using ADVAPI32dll, the executable file has functions like **WinExec** and **WriteFile** has access to write files in disk and **LoadResource** could be used to read resource section of the file it is trying to write to.

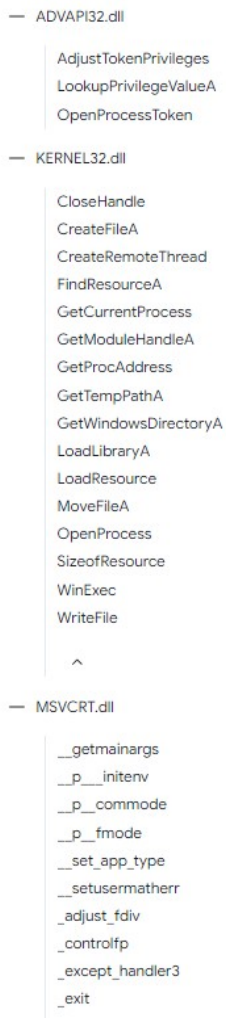

Figure 9: Lab01-04.exe Imports

5. **What host- or network-based indicators could be used to identify this malware on infected machines?**

   Host based indicators - wupdmgrd.exe in system32 indicate that it has total access to windows Upgrade Download Manager.

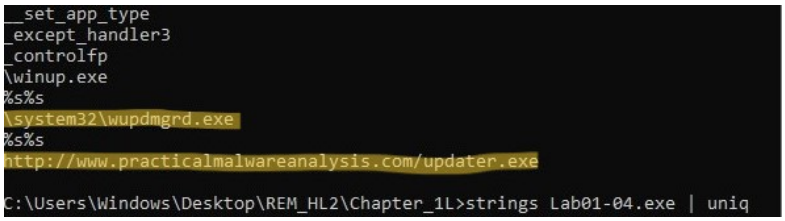

Figure 10: Network based indicators for Lab01-04.exe

6. **This file has one resource in the resource section. Use Resource Hacker to examine that resource, and then use it to extract the resource. What can you learn from the resource?**

   This resource is downloading another executable file, by using **URLDownloadToFileA** import from a remote site - http://practicalmalwareanalysis.com/updater.exe.

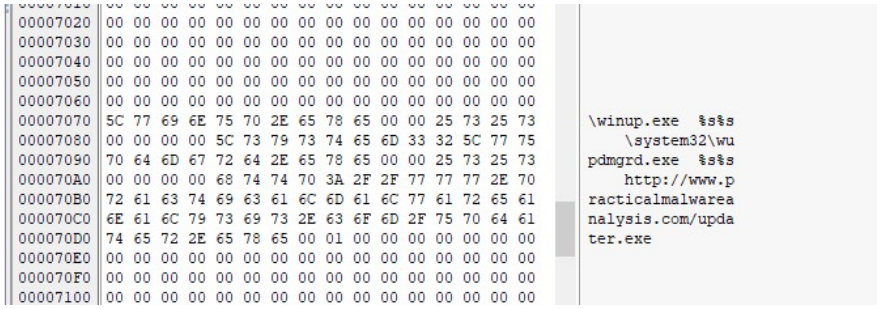

Figure 11: Resource Hacker for Lab01-04.exe

4.2 Part 2: Analyzing Malware Files in Tools using Virtual Machines

Attach result screenshots from all the tools (HashCalc, DIE, HxD, PE-bear, Strings, and CyberChef) that we have used in this part of the lab.

4.2.1 File: re_lotsastuff.exe

- **Identifying the hash of the file – md5 and sha256**

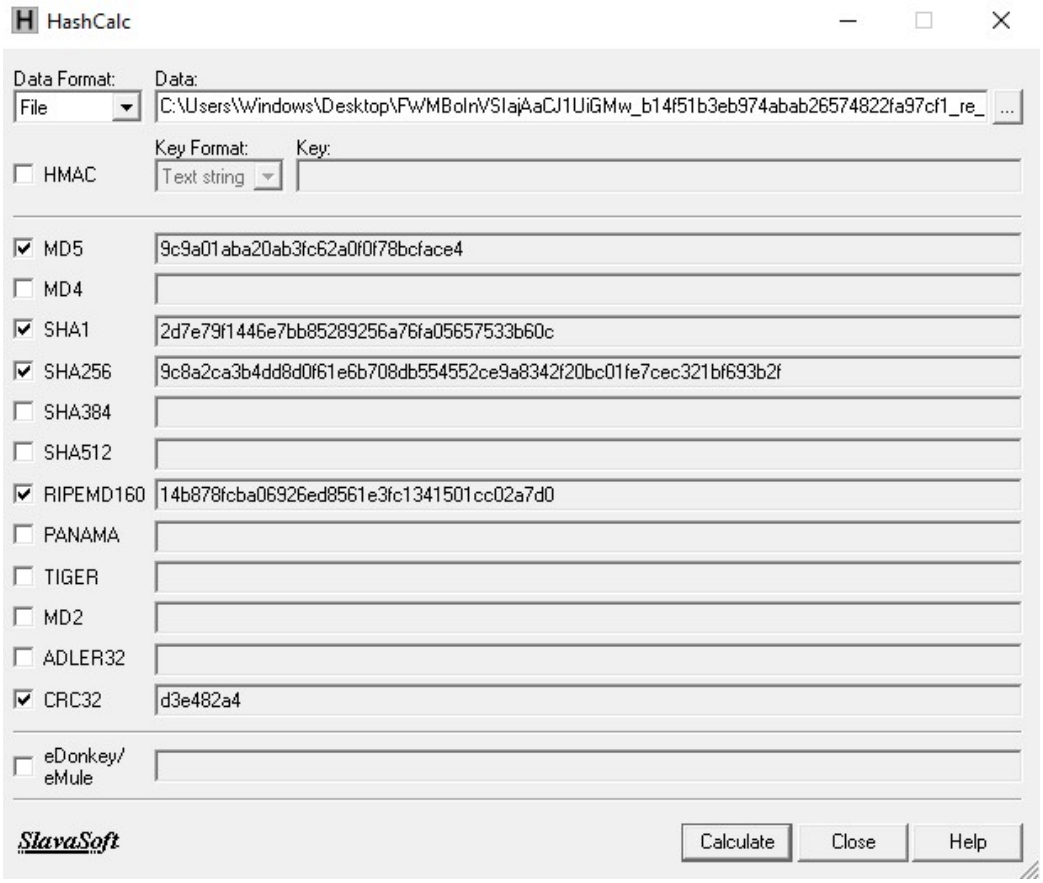

Figure 12: Hash Value of re_lotsastuff.exe

- **Identifying the type of the file - .exe, .dll, office document, script, etc.**

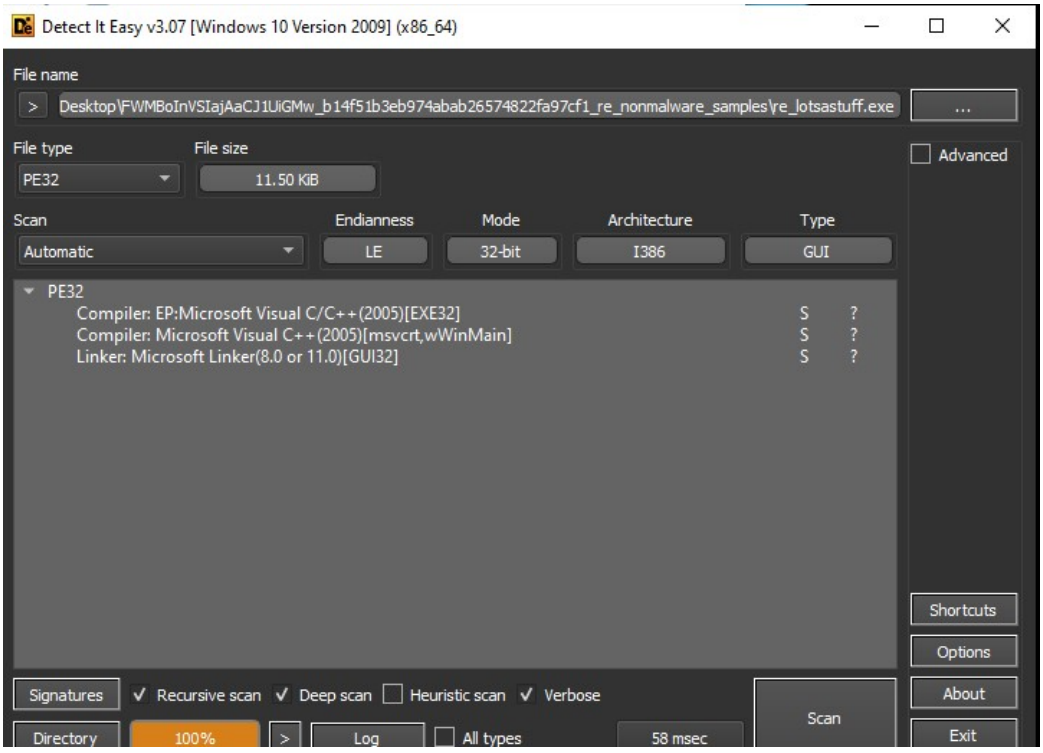

Figure 13: Detect It Easy for re_lotsastuff.exe

- **Identifying the file details – time stamp, compiler**

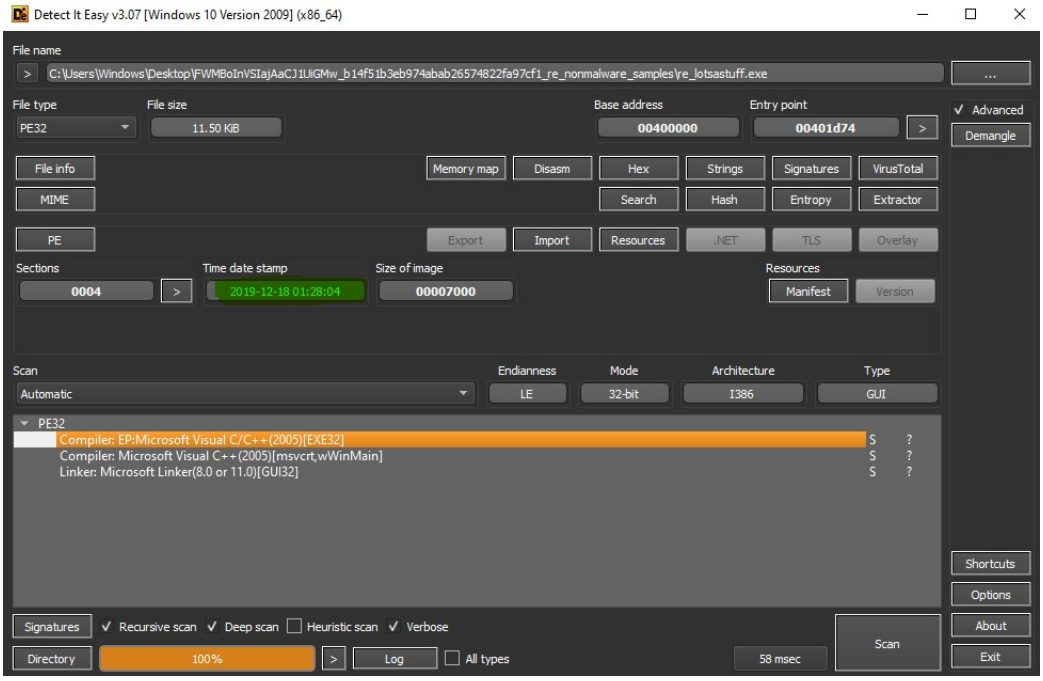

Figure 14: Compilation Timestamp for re_lotsastuff.exe

- **Identifying the text strings – URLs, filenames, registry**

  Using Strings in Windows command line, we can deduce the list of filenames, URL's, DLL Imports and some random junk data.

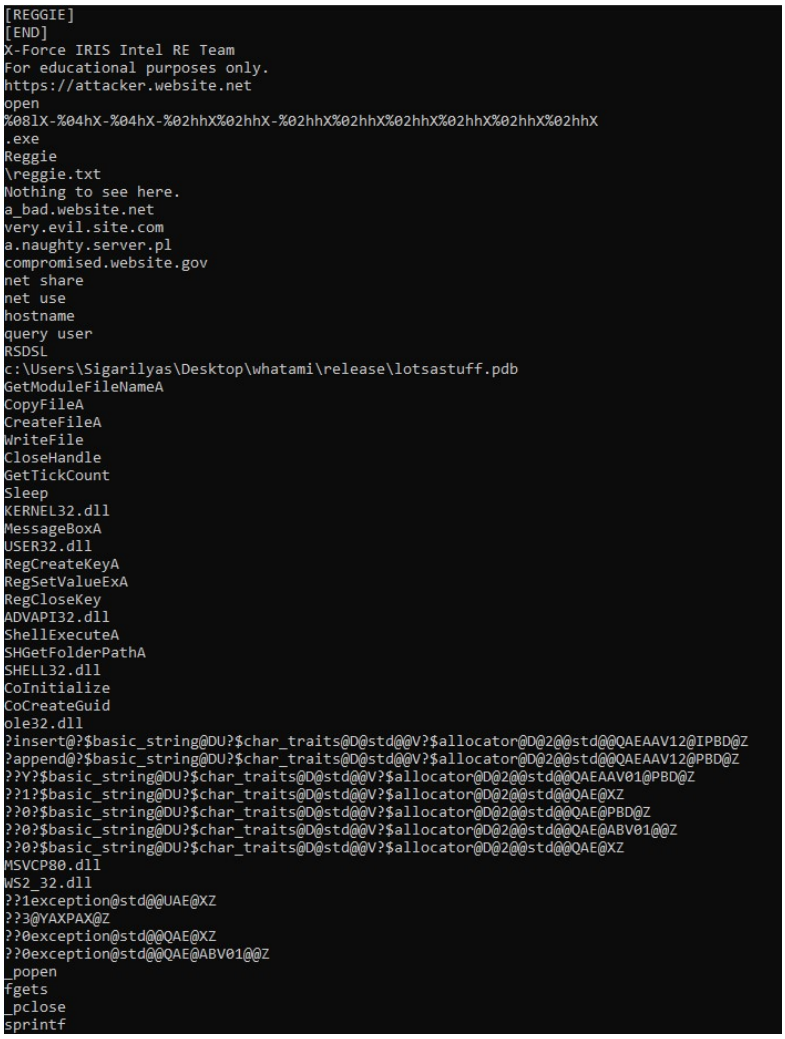

Figure 15: Strings output

- **Identifying the imported API calls**

  Using PE Bear, using the **Imports** tab, we can see all the DLL imports and functions the executable is importing.

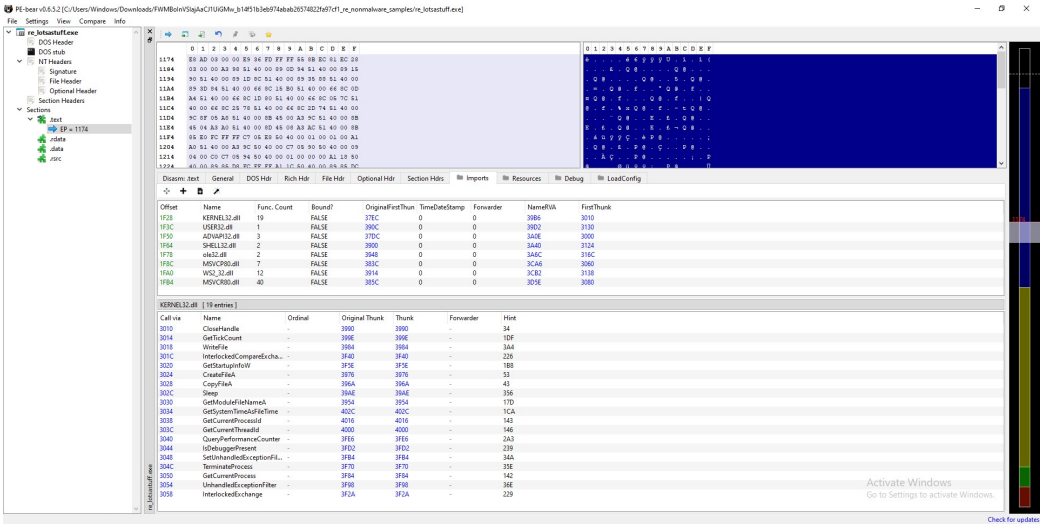

Figure 16: PE Bear Imports Tab

4.2.2 File: a.message.ps1

- **Identifying the hash of the file – md5 and sha256**

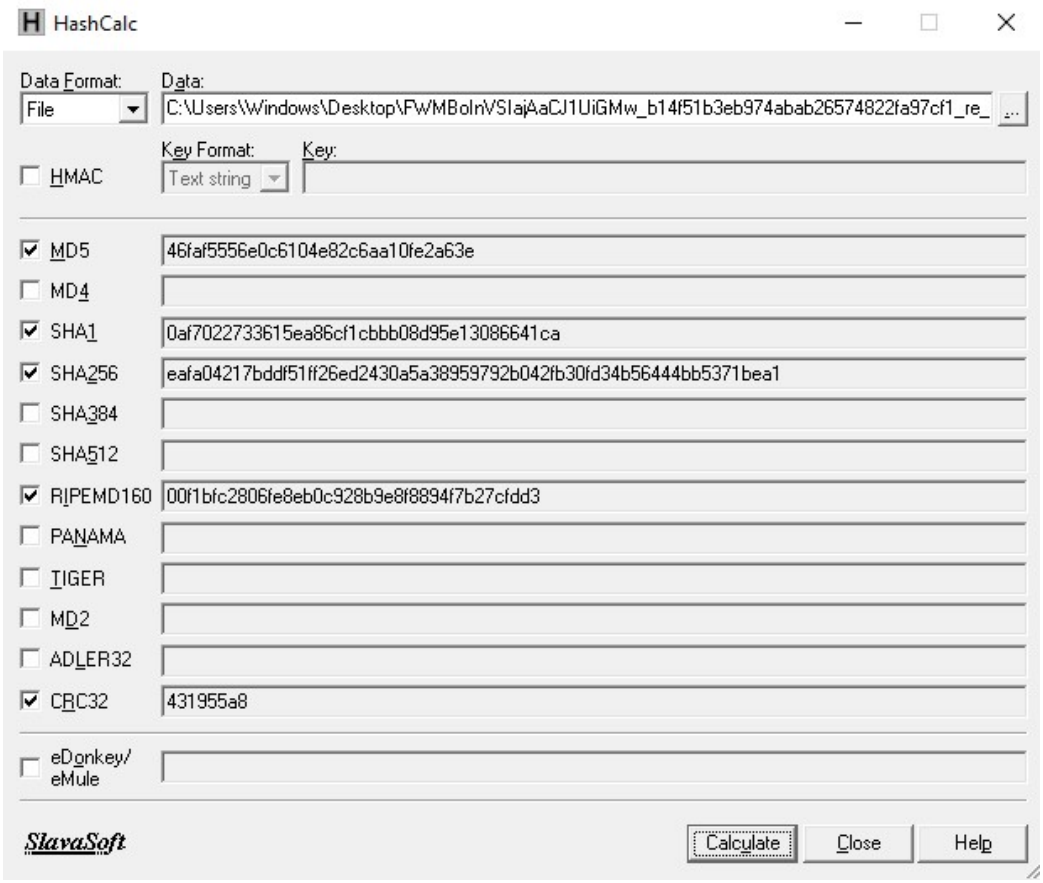

Figure 17: Hash Values of a_message.ps1

- Strings Output of **a.message.ps1**

```
C:\Users\Windows\Desktop>strings a_message.ps1

Strings v2.54 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

$MYCOMMAND = "YAFMRFJSQEZEAUdOUwFCWENEU0JJREc="
$byteString = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($MYCOMMAND))
$DECODED = $(for ($i = 0; $i -lt $byteString.length; ) {
$byteString[$i] -bxor 33
$i++
Write-Output [System.Text.Encoding]::UTF8.GetString($DECODED)

C:\Users\Windows\Desktop>
```

Figure 18: String output of a_message.ps1

- CyberChef Output after analyzing the code in *Figure 18*

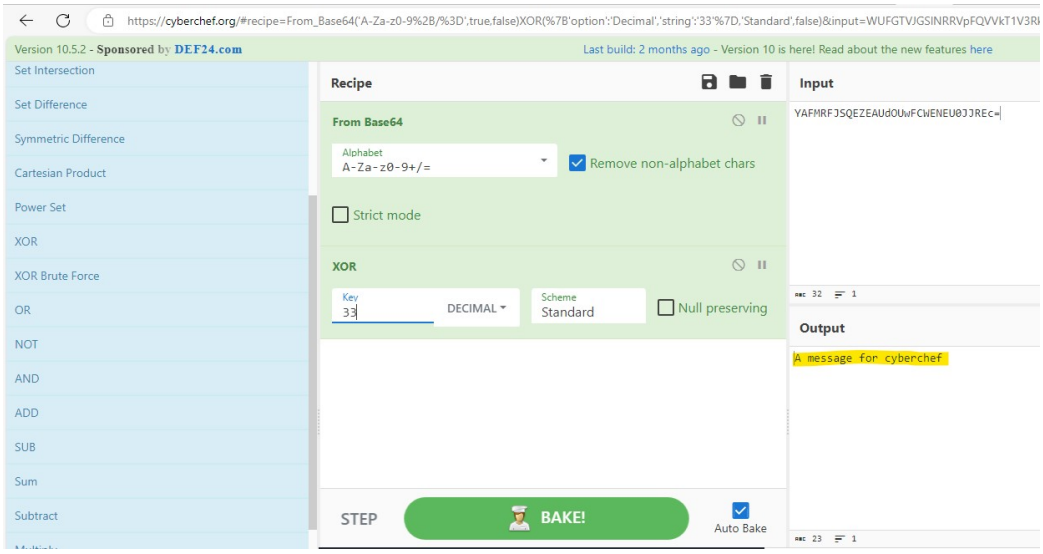

Figure 19: Cyberchef output for a_message.ps1

5 Conclusion

Static Analysis is very useful in detecting and gaining insights on various executable files and linking files using various utilities available to create signatures, detect based on signatures, analyze it's behaviour based on functions it imports and exports.

However, Static analysis doesn't unveil complete behavioral aspects and can be challenged by polymorphic or encrypted malware.