MALWARE ANALYSIS

Basic Static Analysis

Dr. Zeyad Safaa Younus Saffawi

Techniques

- Malware analysis often begins with **antivirus scanning** to confirm whether a file is malicious.
- Analysts may also compute **hashes** to uniquely identify and track malware samples.
- Additionally, **valuable information** can be gathered from examining a file's **strings, functions, and headers**, which can provide insights into its functionality, behavior, and potential indicators for detection.

Antivirus Scanning

- A practical initial step in malware analysis is to run the suspicious file through **multiple antivirus programs**. While antivirus tools can provide useful insights, they are not infallible. These tools primarily rely on **databases of known malicious code signatures**, complemented by behavioral and pattern-matching techniques to identify potentially harmful files.
- However, **antivirus software faces several limitations**. **Malware authors** can easily **modify their code**, thereby **altering its signature and evading detection by traditional scanners**.
- Additionally, rare or newly emerging malware may remain undetected because it has not yet been incorporated into antivirus databases, highlighting the need for complementary analysis techniques beyond standard antivirus scanning.

VirusTotal

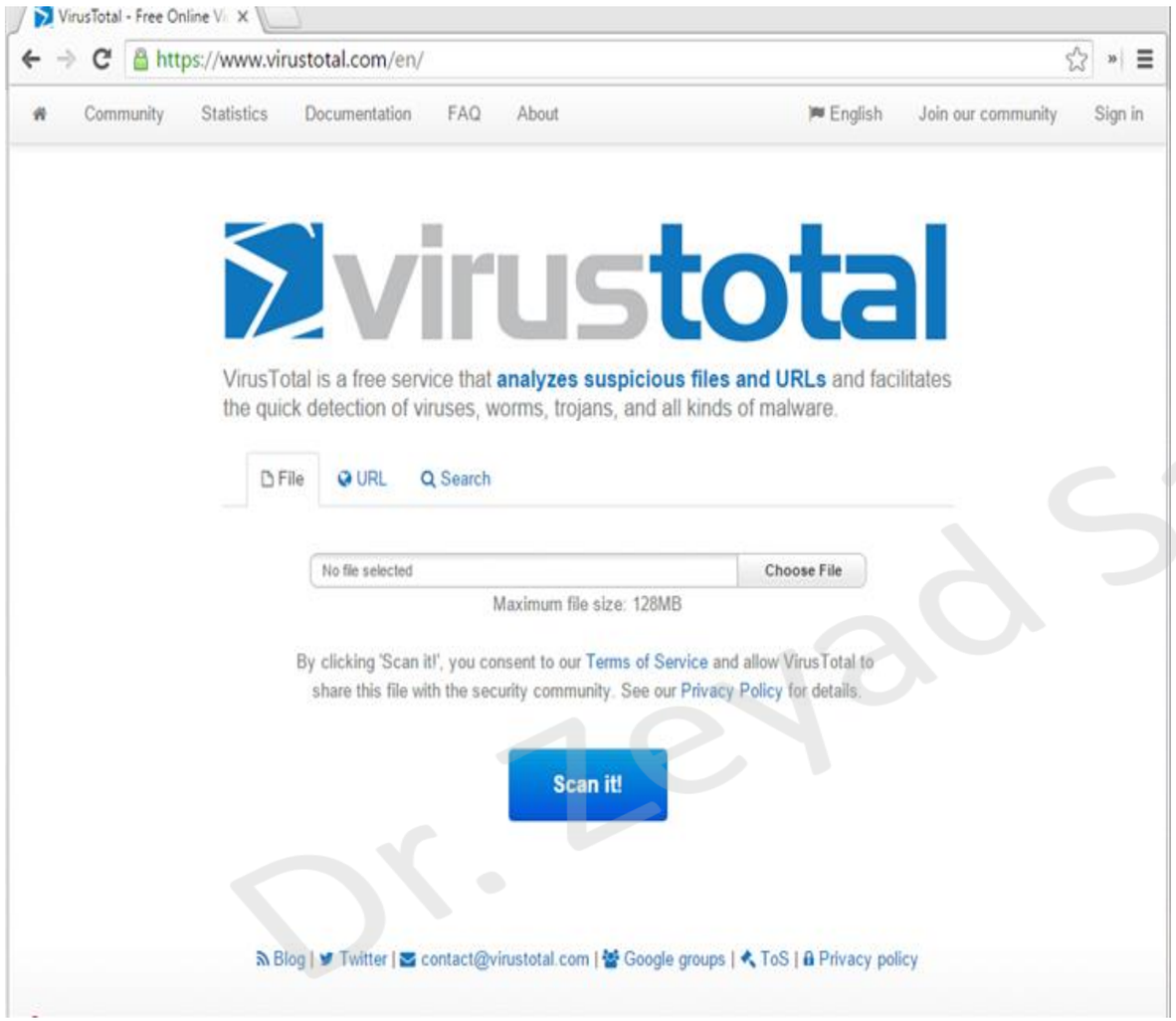
- Because different antivirus programs employ distinct signatures and heuristics, it is often advantageous to scan a suspected malware sample using multiple antivirus engines.

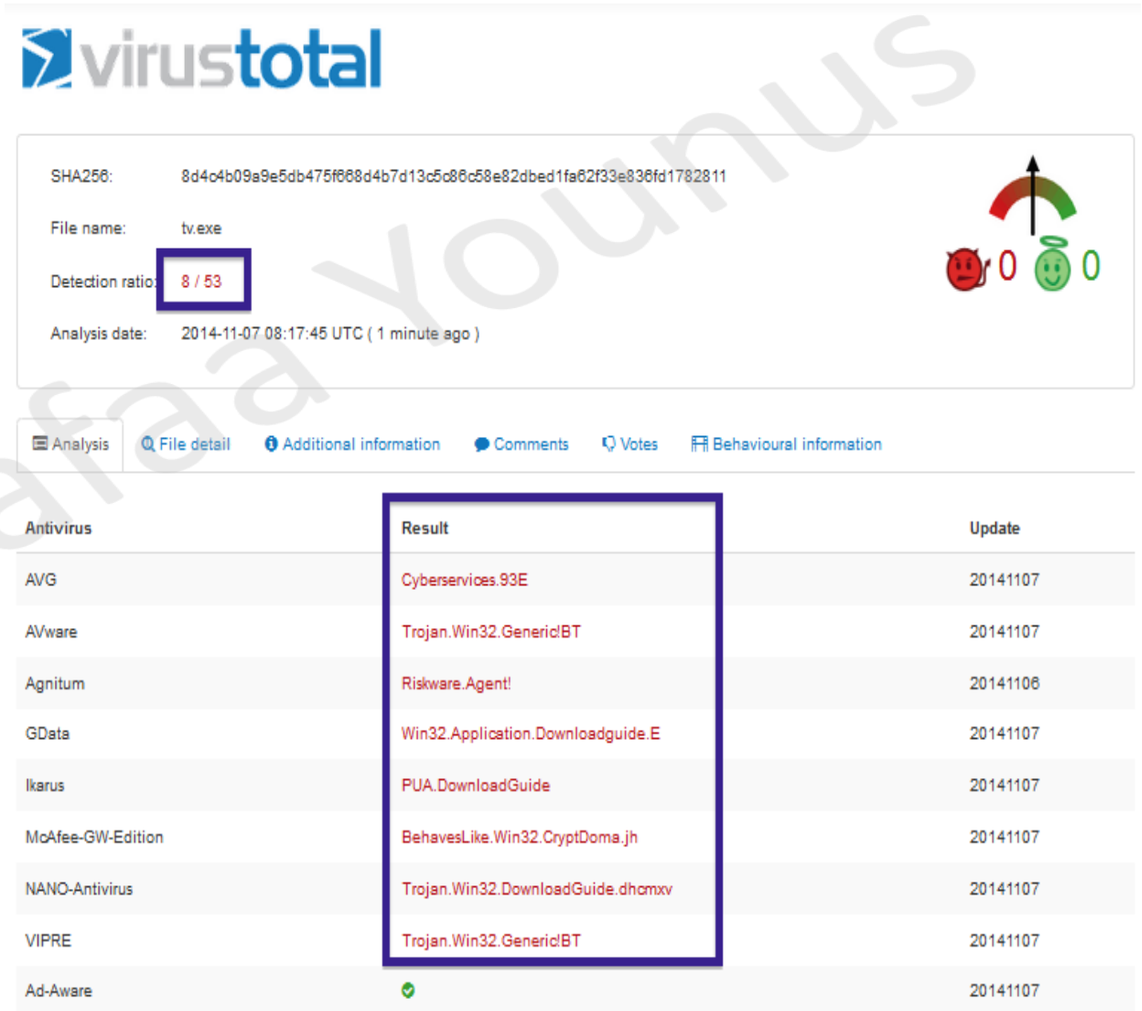
- **VirusTotal** (http://www.virustotal.com/) is an online platform that facilitates this process by allowing users to upload files for simultaneous scanning by numerous antivirus tools. The platform generates a comprehensive report that includes the total number of engines that identified the file as malicious, the associated malware names, and additional relevant metadata.

VirusTotal

- VirusTotal is particularly important in malware analysis because it enables researchers and security professionals to quickly assess whether a file is potentially harmful using multiple antivirus engines.
- By aggregating results from different engines, analysts can reduce false negatives and obtain a more reliable indication of maliciousness. The platform also provides rapid preliminary analysis without the need to install and run multiple antivirus programs locally, and its multi-engine approach increases the likelihood of detecting malware that might evade a single antivirus tool.
- Furthermore, VirusTotal generates a comprehensive report that assists in gathering critical information for further investigation, creating indicators of compromise (IOCs), and supporting defensive measures. Overall, **VirusTotal serves** as a crucial **first step** in malware analysis, offering fast, reliable, and comprehensive insight into potential threats.

VirusTotal





Hashing: A Fingerprint for Malware

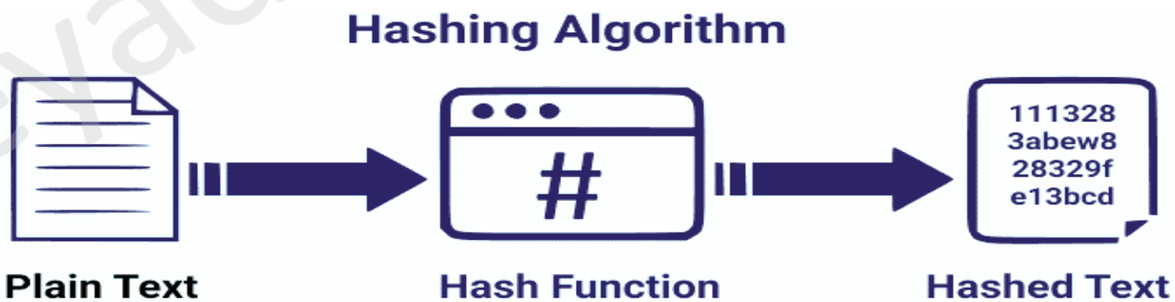
- **Hashing** is a technique used to generate a fixed-length digital fingerprint of a file using a cryptographic hash function. It is a fundamental technique extensively employed in malware analysis to generate a unique digital fingerprint for each malicious file.
- By processing a file through a hashing algorithm, analysts obtain a fixed-length value that uniquely represents the file's contents, enabling precise identification, tracking, and comparison of malware samples across different systems and environments.
- This approach is particularly valuable for detecting known malware, verifying file integrity, and supporting forensic investigations.

Hashing: A Fingerprint for Malware

- The most commonly used hashing algorithms in malware analysis include the **Message-Digest Algorithm 5 (MD5)**, **Secure Hash Algorithm 1 (SHA-1)**, and **Secure Hash Algorithm 256 (SHA-256)**.
- Each of these algorithms produces a unique hash value, though they differ in terms of cryptographic strength and collision resistance.
- By leveraging these hash functions, security professionals can create reliable **identifiers** for malware, facilitating automated detection, correlation of threats across networks, and the development of **indicators of compromise** (**IOCs**).

Hashing: A Fingerprint for Malware

- The generated hash can be used as a reference identifier to label samples, share them with other analysts, and search threat intelligence databases such as **VirusTotal** to determine whether the file has been previously analyzed or classified as malicious.
- While hashing is effective for identifying known malware, it is ineffective against modified or obfuscated samples, as even minor changes in the file result in a completely different hash.



Hashing: A Fingerprint for Malware

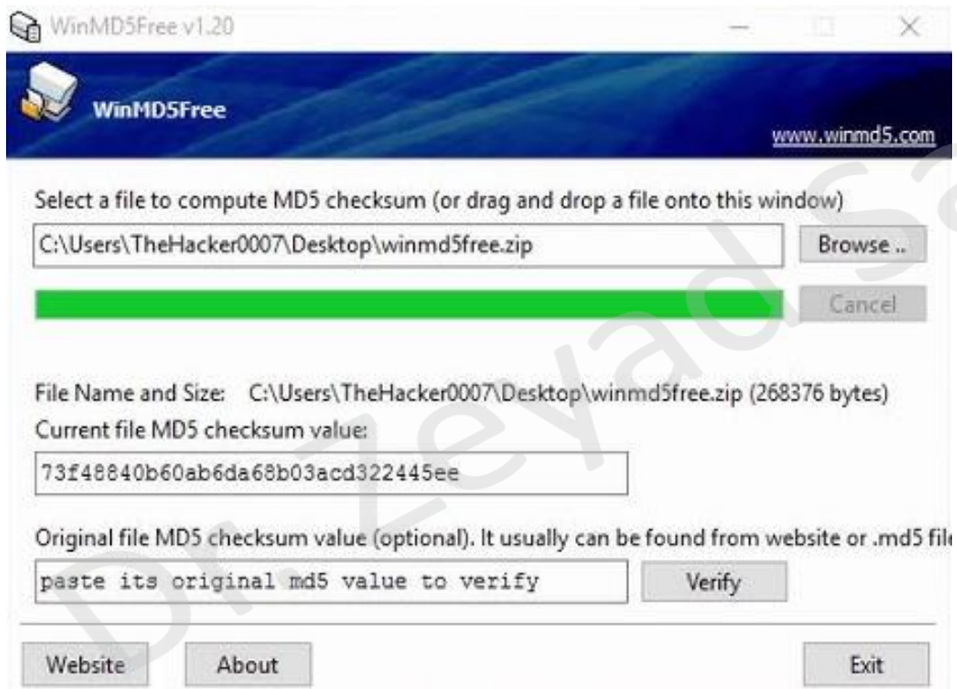
- For example, using a freely available hashing tool such as **md5deep**, one can calculate the MD5 hash of a file, such as the Solitaire program included with Windows. The resulting hash value in this instance is:

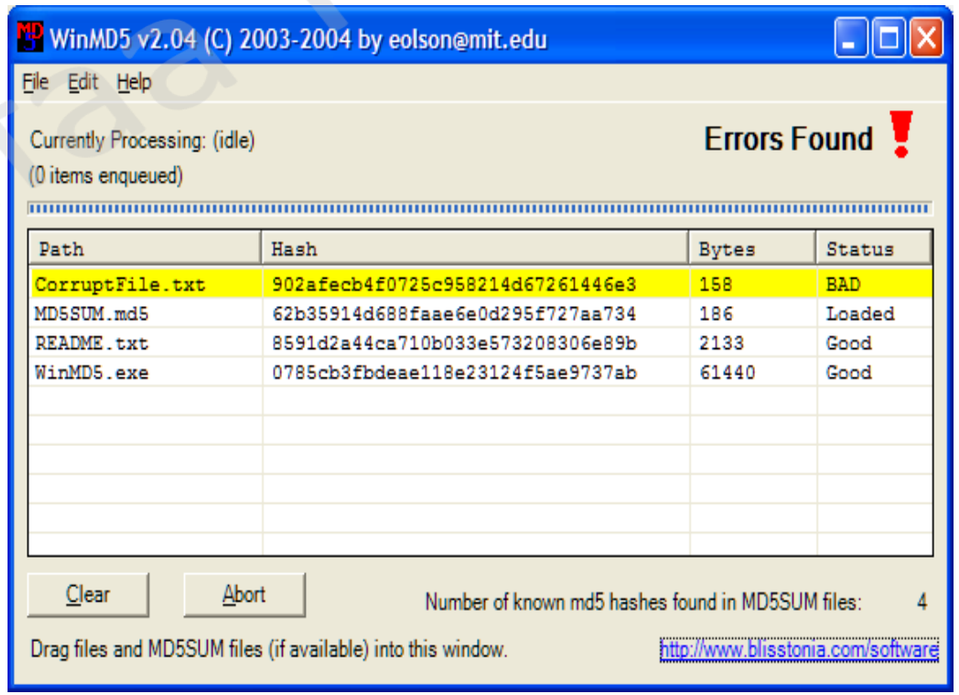
```
C:\>md5deep c:\WINDOWS\system32\sol.exe
373e7a863a1a345c60edb9e20ec32311  c:\WINDOWS\system32\sol.exe
```

- The hash is **373e7a863a1a345c60edb9e20ec32311**.
- This hash can serve as a unique identifier or label for the file, enabling security analysts to share the value with colleagues to facilitate collaborative identification of malware. Additionally, the hash can be searched in online databases to determine whether the file has already been analyzed or classified as malicious, providing a rapid method for cross-referencing and confirming the file's status.

Win MD5 and hash calculator tools

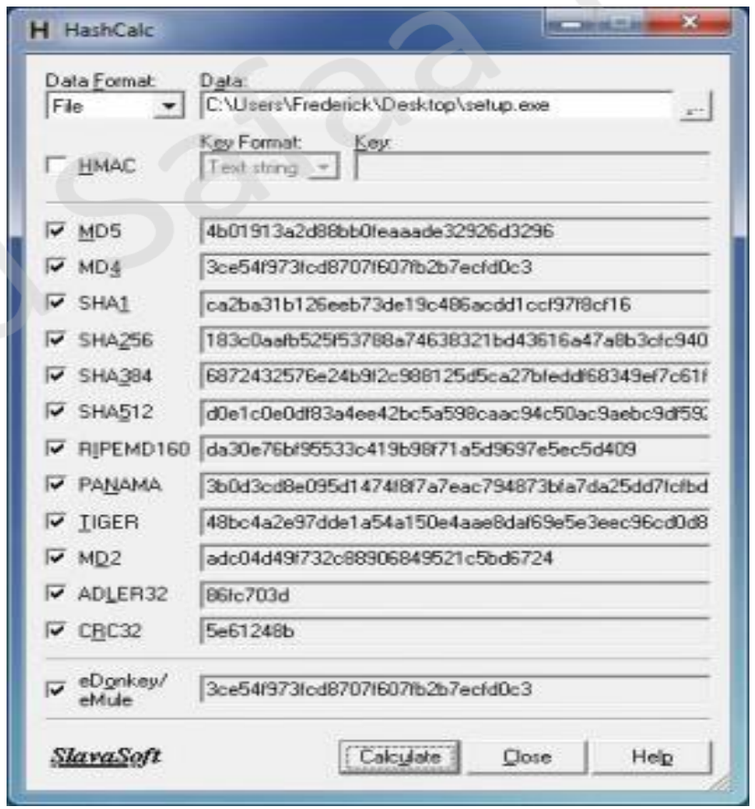
- **MD5** is a widely used utility that enables the creation and comparison of MD5 checksums. It allows analysts to compare the contents of two files or to verify a file against a known checksum string. Alternatively, the application can be used to calculate a checksum for a single file, producing a unique digital fingerprint that can serve as a reliable identifier for file verification, malware detection, and forensic analysis.





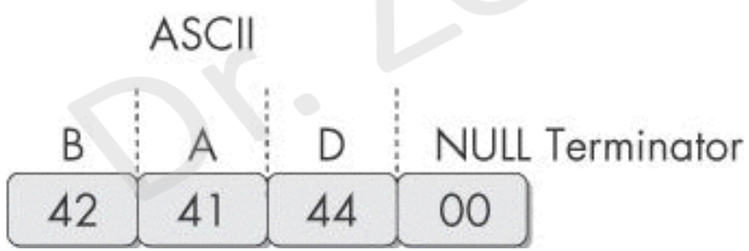
hash calculator (hash calc) tool

- The **Hash Calculator (HashCalc)** tool is a utility that computes the hash of a given input using a variety of cryptographic algorithms. By generating unique hash values, HashCalc allows analysts to verify file integrity, compare files, and identify malware through consistent digital fingerprints. Its support for multiple algorithms provides flexibility and enhances reliability in malware analysis and forensic investigations.

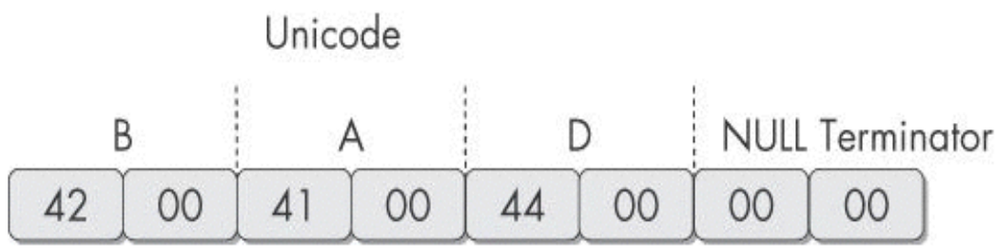


Finding Strings

- In computer programs, a **string** is defined as a sequence of characters.
- Strings are typically **terminated** by a **null character (0x00)** to indicate the end of the sequence.
- Character encoding standards determine how characters are represented in memory.
- For instance, **ASCII** characters are encoded using **8 bits** (1 byte per character), whereas **Unicode** characters generally use **16 bits** (2 bytes per character) to accommodate a wider range of symbols and international characters. Understanding string encoding is essential in malware analysis, as malicious programs often embed important indicators, commands, or data within string sequences.



ASCII representation of the string BAD



Unicode representation of the string BAD

Finding Strings

- The **Strings** program is designed to search for sequences of three or more ASCII or Unicode characters in a file, followed by a string termination character. While this approach can reveal meaningful text embedded in a program, it is important to note that not all detected sequences represent actual strings.
- For example, a sequence of bytes such as 0x56, 0x50, 0x33, 0x00 may be interpreted by Strings as the string "VP3," yet these bytes could instead represent a memory address, CPU instructions, or other program data. Consequently, it is the analyst's responsibility to review and filter the extracted sequences to distinguish valid strings from irrelevant or misleading data.

Finding Strings command

- Most invalid strings are obvious because they do not represent actual text.
- For example, the following excerpt shows the result of running Strings against the file bp6.ex_:
  - Bold items can be ignored
  - GetLayout and SetLayout are Windows functions
  - GDI32.DLL is a Dynamic Link Library
- **Dynamic Link Library (DLL) files** contain executable code that can be shared among multiple applications. DLLs allow programs to modularize functionality by providing common routines, functions, and resources that can be accessed by different software components without duplicating code. Understanding DLL files is important in malware analysis because malicious code can be embedded within DLLs to execute harmful operations while remaining less conspicuous within a system.

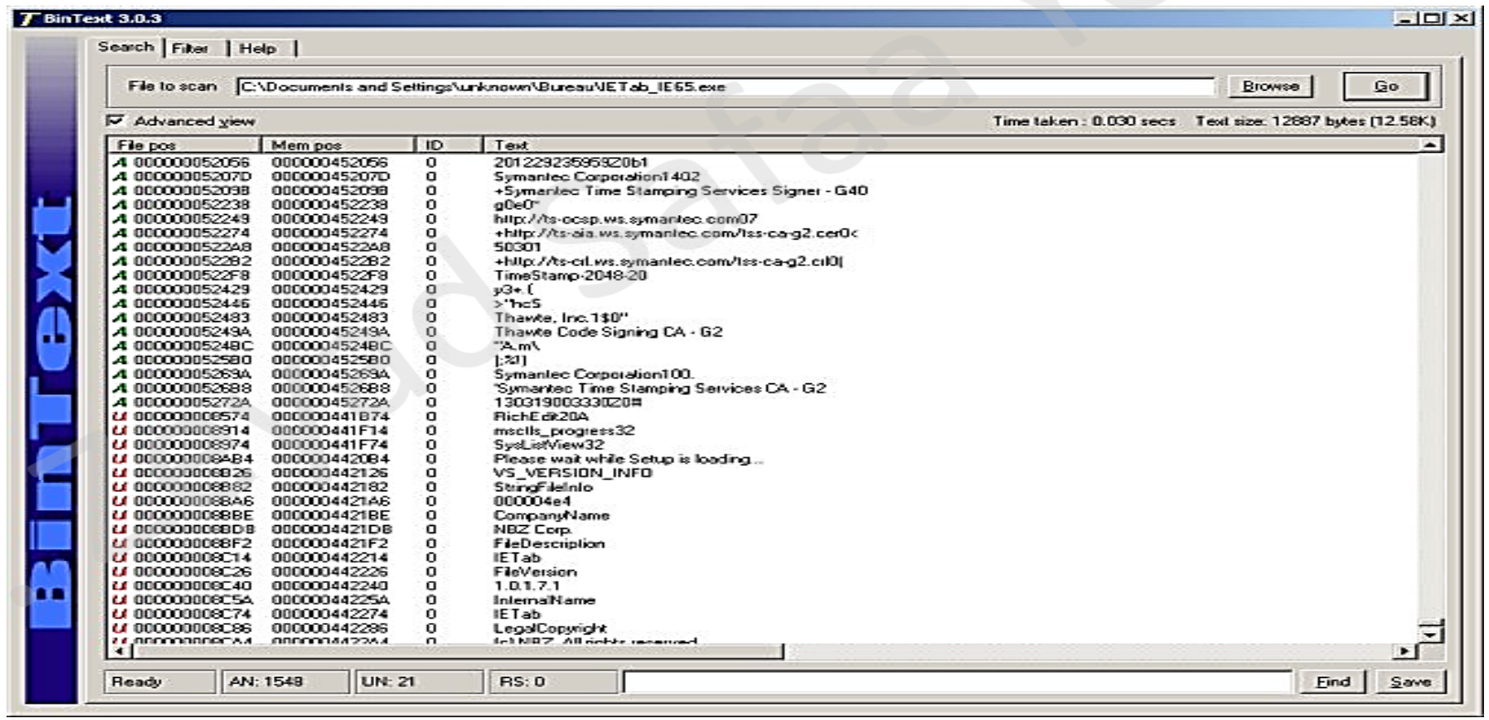
```
C:>strings bp6.ex_
VP3
VW3
t$@
D$4
99.124.22.1 ❹
e-@
GetLayout ❶
GDI32.DLL ❸
SetLayout ❷
M}C
Mail system DLL is invalid.!Send Mail failed to send message. ❺
```

IP address most likely one that the malware will use

error message.

BinText tool

- **BinText** is a useful tool for extracting ASCII and Unicode strings from binary files, helping analysts uncover embedded text within executables and libraries.

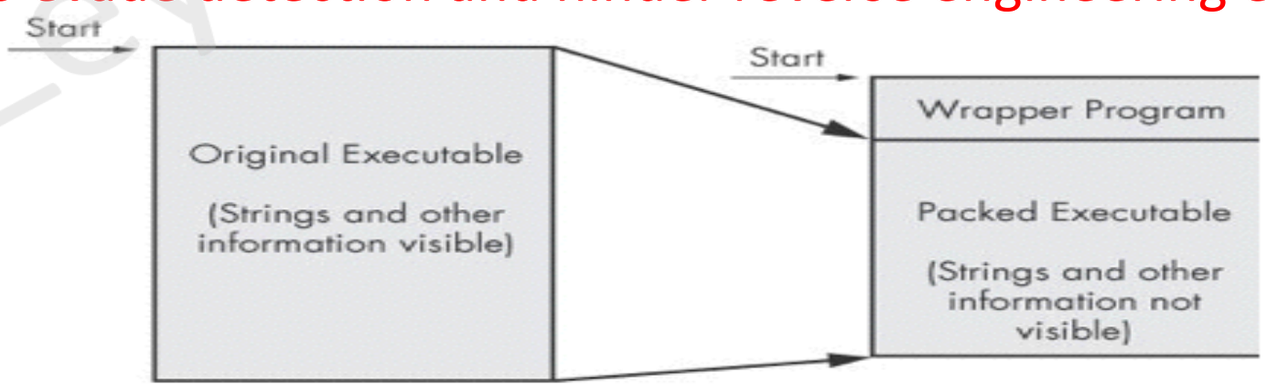


Packed and Obfuscated Malware

- **Packing** and **obfuscation** are techniques commonly employed to make malicious files more difficult to detect or analyze.
- **Obfuscated programs** are those in which the malware author intentionally attempts to conceal the program's execution logic and behavior.
- **Packed programs** represent a subset of obfuscated programs, where the malicious code is compressed or encrypted, preventing direct analysis of the original program instructions.
- In contrast, legitimate software typically contains a large number of readable strings that reflect normal functionality and user interaction.
- Malware that is packed or obfuscated, however, often contains very few visible strings.
- Therefore, when analyzing a program using string extraction tools, the presence of only a limited number of strings may indicate that the file is obfuscated or packed, suggesting potential malicious intent.

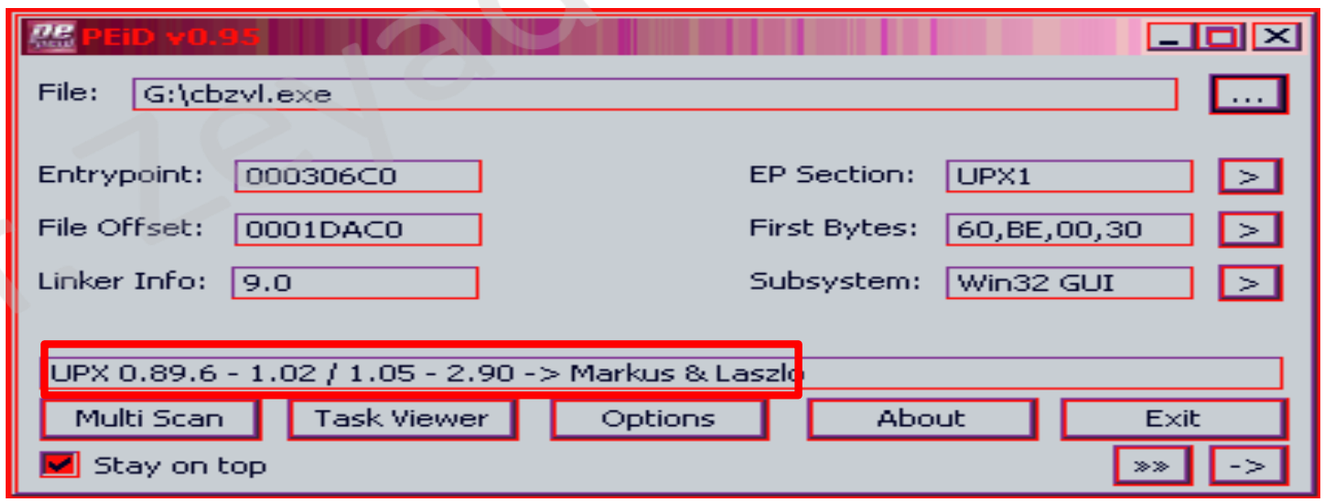
Packed Malware

- When a program is **packed**, its code is compressed, similar to a Zip file, which renders both the program instructions and embedded strings unreadable during static analysis.
- Packed programs typically include a small **wrapper**, a piece of code responsible for decompressing the original program at runtime. When the packed program is executed, the wrapper first runs to unpack the compressed code and then transfers control to the original, now-decompressed executable.
- From a static analysis perspective, only the wrapper program is visible and analyzable prior to execution, making it challenging to understand the true functionality of the packed malware. This technique is commonly used by malware authors to evade detection and hinder reverse engineering efforts.



Detecting Packers with PEiD

- A **packed file** is an executable that has been compressed or encrypted in order to conceal its original code. Malware authors commonly employ packing techniques to evade detection mechanisms and to complicate static and dynamic analysis. Identifying whether an executable is packed, and determining the packing method used, is therefore an important step in malware analysis.
- **PEiD** is a specialized tool used to detect the packer or compiler applied to an executable file, such as **UPX** or **ASPack**. By identifying the packing technology, PEiD assists analysts in selecting appropriate unpacking strategies and facilitates deeper analysis of the underlying malicious code.



References

- Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press.
- Bowne, S. (n.d.). *CNIT 126 Ch 0: Malware Analysis Primer & 1: Basic Static Techniques* [PowerPoint slides]. SlideShare. https://www.slideshare.net/slideshow/cnit-126-ch-0-malware-analysis-primer-1-basic-static-techniques/71040577
- Bowne, S. (n.d.). *CNIT 126: Ch 0-1 Basic Static Techniques (Part 1)* [Video]. YouTube. https://www.youtube.com/watch?v=4ZY4LC2XpFM