Static Analysis

Static Analysis Steps

- Determine file type
- Cryptographic hash (fingerprinting the malware)
- Extracting Strings
- Detecting file obfuscations (packers, cryptors)
- Submission to multi AV scanning engines
- Pattern matching using YARA
- Fuzzy Hashing and comparison
- Inspecting PE Imports
- Inspecting PE Header

File Type

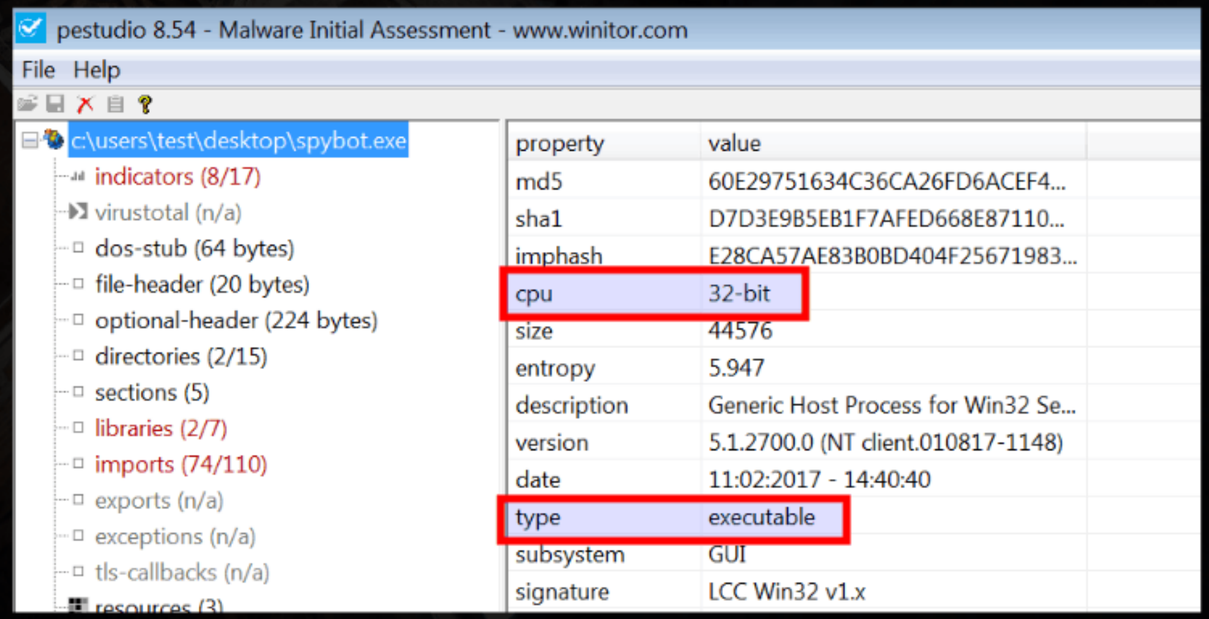
- Helps in determining the architecture the malware is targeting

Example: *If the file type is PE, it can be deduced that the target is Windows*

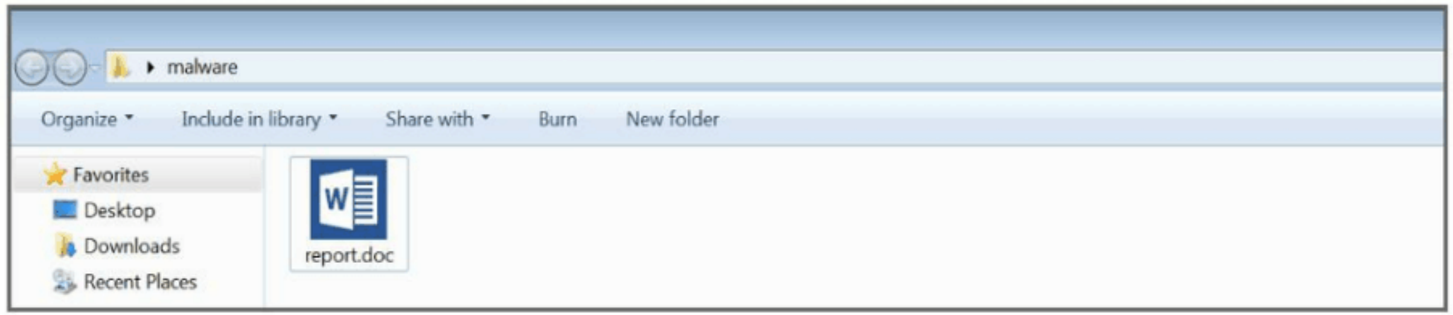
- File extension is not the indicator of file type
- Tools: ***File utility*** (UNIX based systems) or ***pestudio*** (Windows)

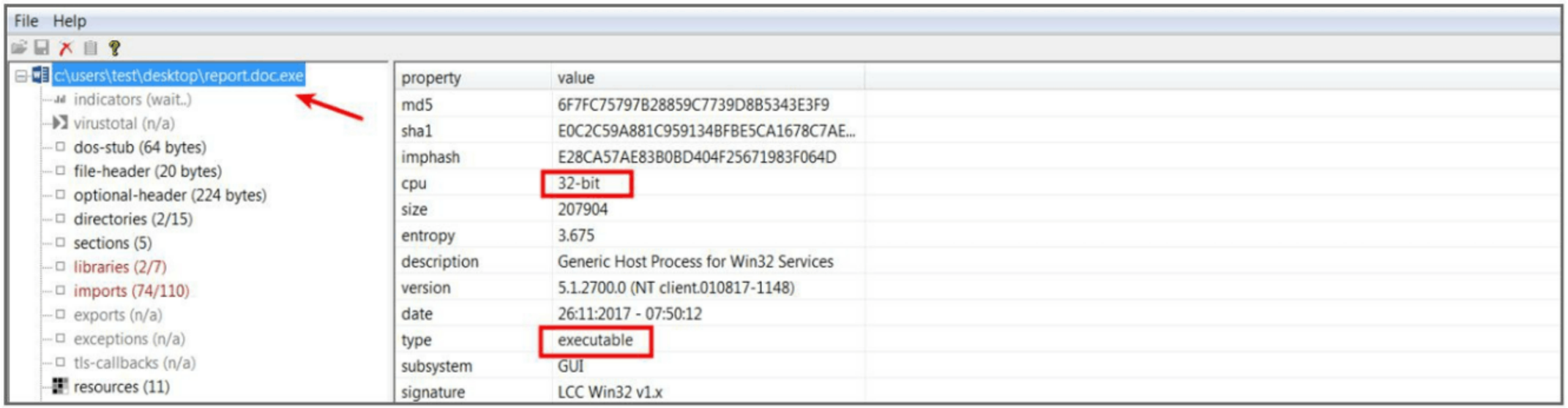
Determining File Type

```
root@kratos:~/Desktop/malware# file spybot.exe
spybot.exe: PE32 executable (GUI) Intel 80386, for MS Windows
root@kratos:~/Desktop/malware#
root@kratos:~/Desktop/malware# file spybot.pdf
spybot.pdf: PE32 executable (GUI) Intel 80386, for MS Windows
root@kratos:~/Desktop/malware#
```



In the following example, the malicious file was made to look like a word document by changing the file extension from "***.exe***" to "***.doc.exe***"



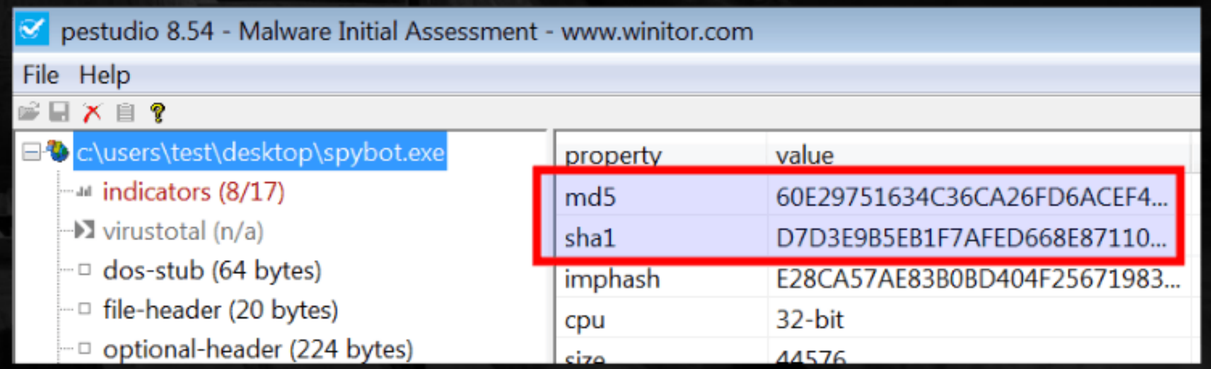


Cryptographic Hash (Fingerprinting the Malware)

- Used to Share with other researchers to help them identify the malware
- Helps in online search to determine if the malware is already identified
- Can serve as a unique identifier throughout the course of analysis
- Tools: ***md5sum, sha256sum, sha1sum*** (Linux) or ***pestudio*** (Windows)

Determining the Cryptographic Hash

```
root@kratos:~/Desktop/malware# md5sum spybot.exe
60e29751634c36ca26fd6acef4d9554e  spybot.exe
root@kratos:~/Desktop/malware# sha256sum spybot.exe
c6c9d204f39b8828c1b40a43b2cc3657a44bb44bcd7f1a098c41837eb99ec69a  spybot.exe
root@kratos:~/Desktop/malware# sha1sum spybot.exe
d7d3e9b5eb1f7afed668e87110a546f856331f68  spybot.exe
root@kratos:~/Desktop/malware#
```



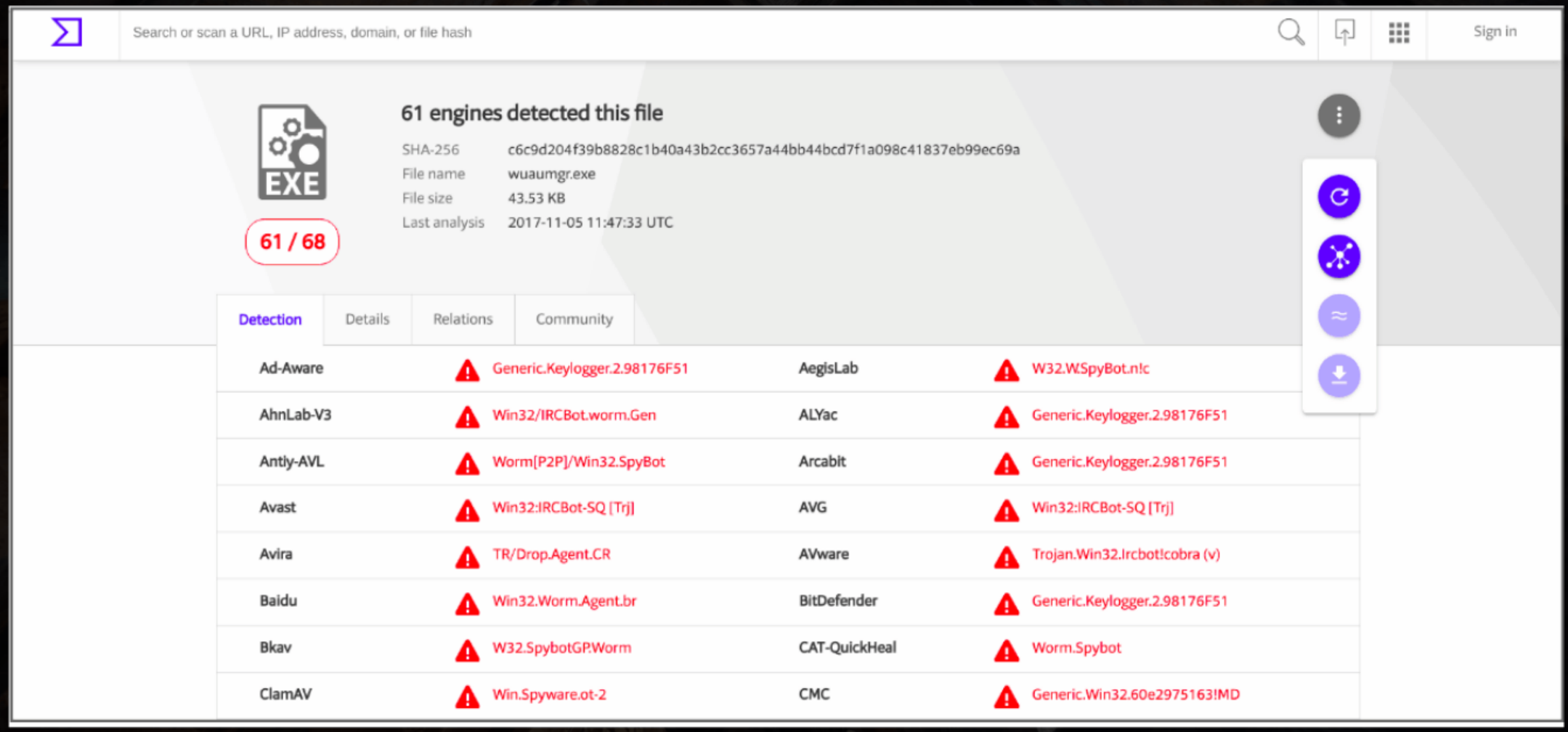
Strings

- Strings are plain text ***ASCII*** and ***UNICODE*** characters embedded within a file
- Can give clues about the functionality and commands associated with a malicious file
- Can contain references to interesting strings like domain name, file name, URL etc.
- Tools ***strings utility*** (Linux) or ***pestudio*** (Windows)

Extracting Strings

```
root@kratos:~/Desktop/malware# strings -a spybot.exe
!This program cannot be run in DOS mode.
.text
`.bss
.data
.idata
.rsrc
more
SynFlooding: %s port: %i delay: %i times:%i.
bla bla blaaaasdasd
Portscanner startip: %s port: %i delay: %ssec.
Portscanner startip: %s port: %i delay: %ssec. logging to: %s
kuang
sub7
%i.%i.%i.0
scan
redirect %s:%i > %s:%i
Keylogger stoped
stopkeylogger
Keylogger logging to %s
Keylogger active output to: DCC chat
Keylogger active output to: %s
error already logging keys to %s use "stopkeylogger" to stop
```

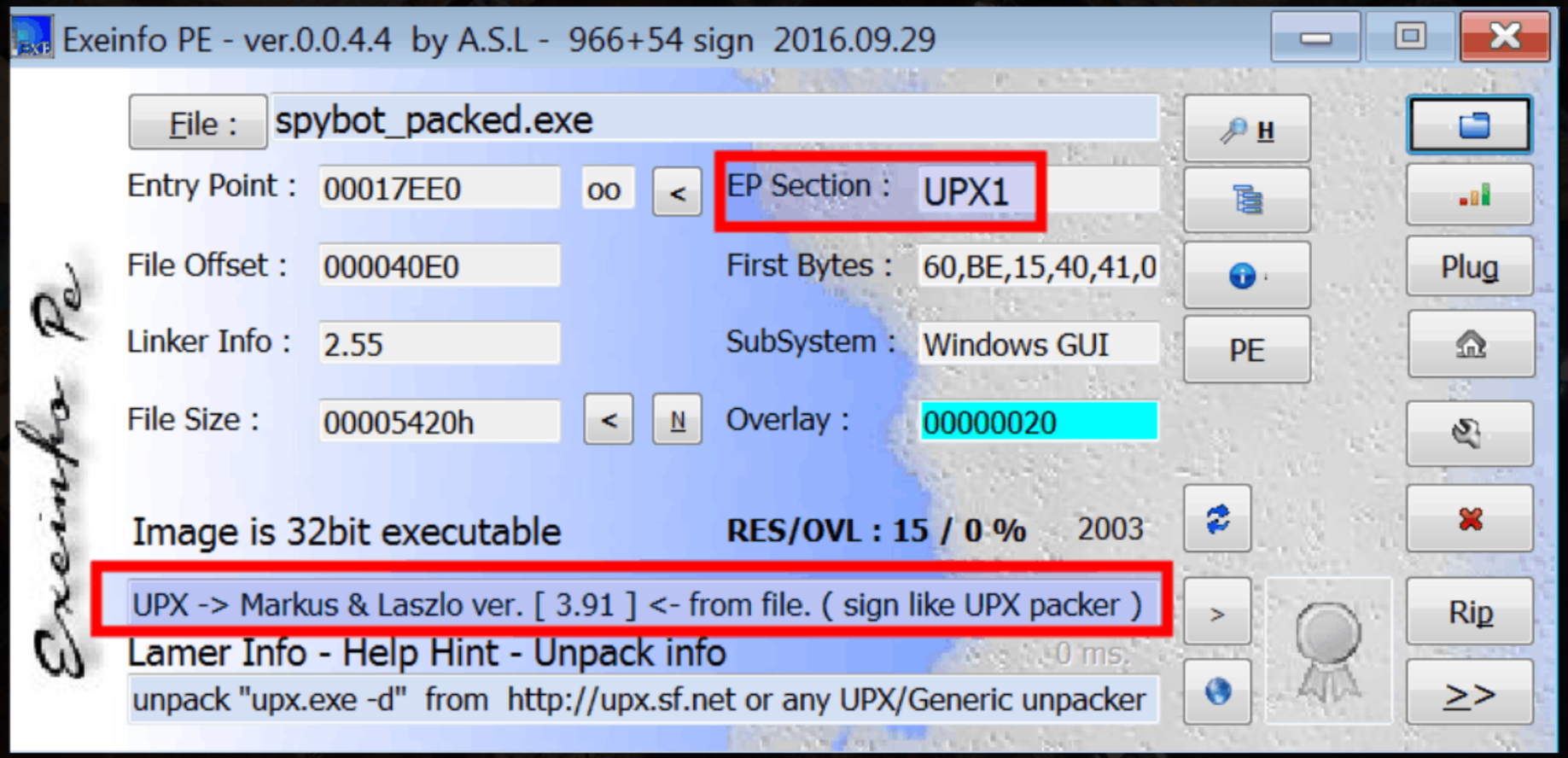
Submission to Multi AV Scanning Engine (such as VirusTotal) - will help in determining if the malicious code signatures exist for the suspect file.



File Obfuscation

- Malware authors obfuscate malware to make their files difficult to detect & analyze
- Malware authors often use Packers & Cryptors to obfuscate the file to evade detection
- Obfuscation results in less number of strings & functions
- Determining if the malware is obfuscated can help in identifying if the sample is malicious

Detecting File Obfuscation



Pattern Matching Using YARA

- YARA helps in classifying and identifying malware samples
- Relies on rules based on textual or binary patterns
- Rule consists of a set of strings and a boolean expression which determine its logic

Running YARA Example 1

```
root@kratos:~/Desktop/malware# cat malicious.yara
rule malicious
{
    meta:
    description = "Indicates Malware Behaviour"

    strings:
    $a = "Synflooding" nocase
    $b = "Portscanner" nocase
    $c = "Keylogger" nocase

    condition:
    any of them
}
root@kratos:~/Desktop/malware# yara -s malicious.yara spybot.exe
malicious spybot.exe
0x89d4:$a: SynFlooding
0x8a19:$b: Portscanner
0x8a48:$b: Portscanner
0x8951:$c: Keylogger
0x8c4a:$c: Keylogger
```

Running YARA Example 2

```
root@kratos:~/Desktop/malware# strings spybot_packed.exe
!This program cannot be run in DOS mode.
UPX0
UPX1
```

```
root@kratos:~/Desktop/malware# cat upx_packed.yara
rule UPX_packed
{
    meta:
    description = "Indicates UPX Packer"

    strings:
    $a = "UPX0" nocase
    $b = "UPX1" nocase

    condition:
    all of them
}
root@kratos:~/Desktop/malware# yara -s upx_packed.yara spybot_packed.exe
UPX_packed spybot_packed.exe
0x178:$a: UPX0
0x1a0:$b: UPX1
root@kratos:~/Desktop/malware#
```

Fuzzy Hashing & Comparison

- Technique to compare different items and determine percentage similarity.
- Helps in determining the malwares samples for similarities.
- Helps in determining the variants of the same malware.
- Helps in determining the malwares associated with the same actor group
- Tool: ***ssdeep***

In the below screenshot md5sum does not show similarity whereas ssdeep shows ***99%*** similarity between the samples

```
root@kratos:~/Desktop/malware# md5sum *
48c1d7c541b27757c16b9c2c8477182b  aiggs.exe
92b91106c108ad2cc78a606a5970c0b0  jnas.exe
root@kratos:~/Desktop/malware#
root@kratos:~/Desktop/malware# ssdeep *
ssdeep,1.1--blocksize:hash:hash,filename
384:l3gexUw/L+JrgUon5b9uSDMwE9Pfg6NgrWoBYi51mRvR6JZlbw8hqIusZzZWe:pIAKG91Dw1hPRpcnu+,"
/root/Desktop/malware/aiggs.exe"
384:l3gexUw/L+JrgUon5b9uSDMwE9Pfg6NgrWoBYi51mRvR6JZlbw8hqIusZzZXe:pIAKG91Dw1hPRpcnud,"
/root/Desktop/malware/jnas.exe"
root@kratos:~/Desktop/malware#
root@kratos:~/Desktop/malware# cd ..
root@kratos:~/Desktop# ssdeep -lrpa malware
malware/jnas.exe matches malware/aiggs.exe (99) ←

malware/aiggs.exe matches malware/jnas.exe (99) ←
```

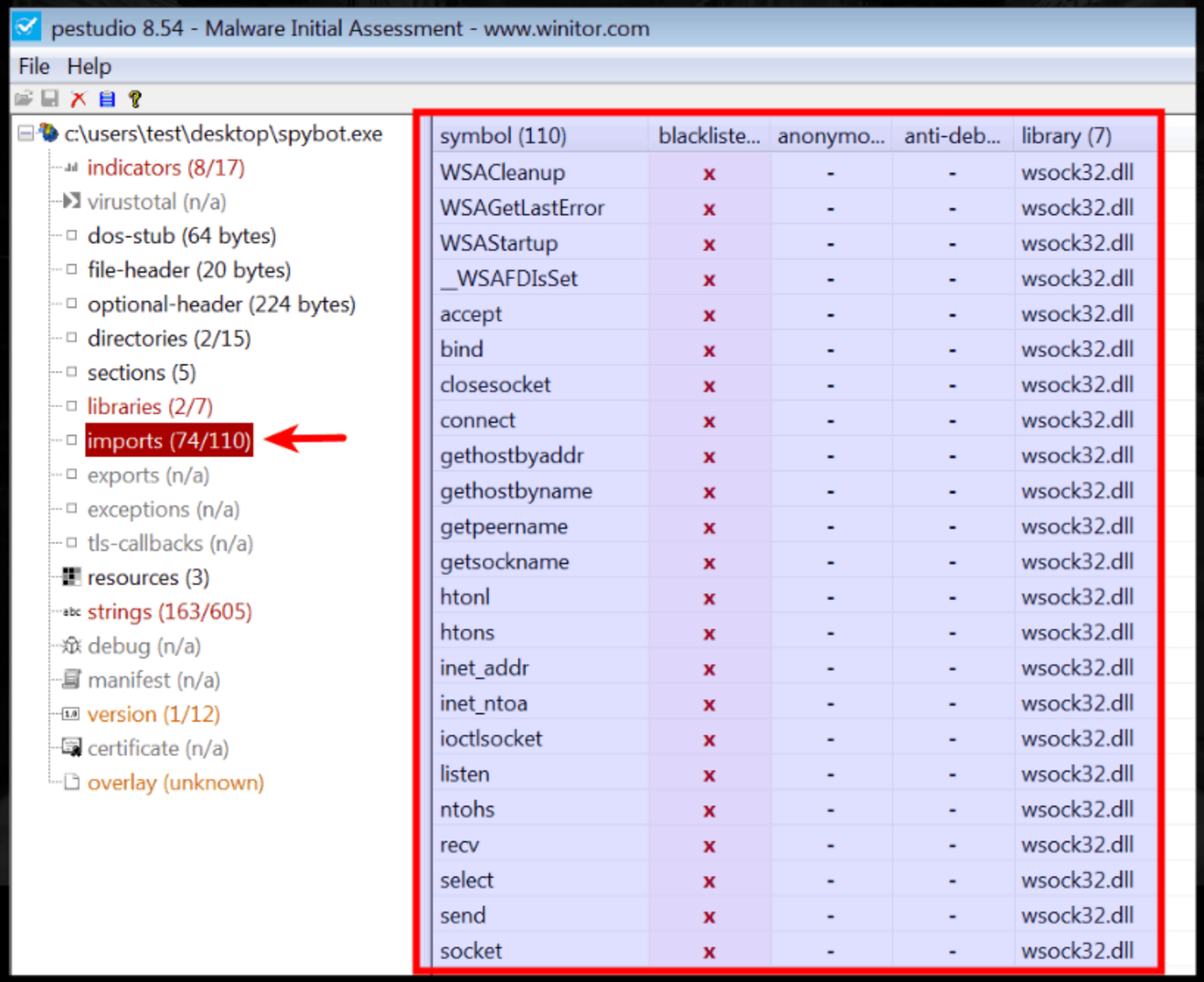
Inspecting PE Imports

- Executable loads multiple shared libraries and call API functions to perform certain actions like resolving domain names, establishing an HTTP connection etc

- Determining the type of shared library and list of API calls imported by an executable can give an idea on the functionality of the malware

- Tool: ***pestudio***

Below screenshot shows the API calls imported from ***wsock32.dll***, this indicates that the malware has network functionality



Inspecting PE Header

The PE header contains useful information such as:

- **Imports:** Functions from other libraries that malware relies on
- **Exports:** Functions in the malware that will be called by other programs
- **Time Stamp:** Specifies when the program was compiled, it can give an idea of when the malware was first created, this can help in building a timeline of the attack campaign
- **Resources:** Strings, icons, and other information included in the file. Very often attackers store information like additional binary, decoy documents, configuration data in the resource section

Following is an example of a malware binary whose timestamp was modified to a older date in ***1987***. In this case, even though the actual compilation timestamp could not be detected but such characteristics can help you identify anomalous behaviour.

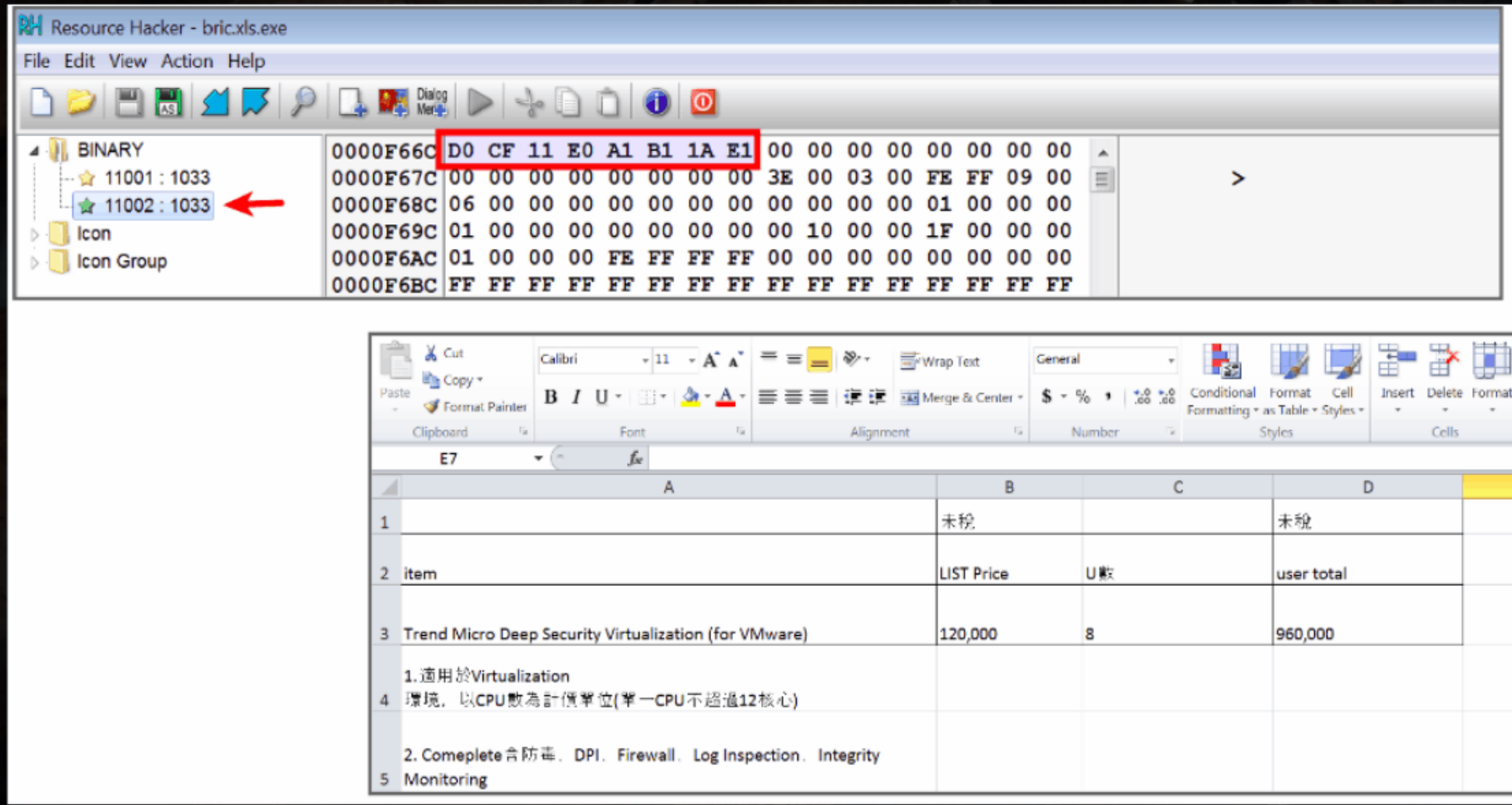
pestudio 8.74 - Malware Initial Assessment - www.winitor.com

File Help

- c:\users\diablo\desktop\pigeon.exe
  - indicators (7/20)
  - virustotal (network error)
  - dos-stub (12 bytes)
  - file-header (Sep.1987)
  - optional-header (suspicious)
  - directories (3)
  - sections (2/2)
  - libraries (0)
  - imports (n/a)
  - exports (0)
  - tls-callbacks (n/a)
  - resources (1/11)
  - strings (0/2/2/1/5372)
  - debug (n/a)
  - manifest (n/a)
  - version (n/a)
  - certificate (n/a)
  - overlay (n/a)

property	value
md5	CBD967A435CBFADB45794CA66AAA5826
sha1	2BEC1A4525043A3AD58292ED900D50FA964398C4
sha256	42C76F27AE13E5B32A8D4ADC2278A4D5E70D5D23799F2B2C7580D36B9E44E34D
first-bytes (hex)	4D 5A 00 00 00 00 00 00 00 00 00 00 50 45 00 00 4C 01 02 00 46 53 47 21 00 00 00 00 00 00 00 00 E0
first-bytes (text)	M Z . . . . . P E . . . L . . . . F S G ! . . . . .
size	365417 bytes
entropy	7.918
imphash	n/a
cpu	32-bit
signature	n/a
entry-point (hex)	
file-version	n/a
file-description	n/a
file-type	executable
subsystem	GUI
compiler-stamp	Fri Sep 11 07:05:02 1987
debugger-stamp	n/a

In the following example, the malware stored a decoy excel document in its resource section, upon execution the malware displays this decoy document to the user as a diversion.



Lab 1 - The case of Remcos RAT

While examining a system, the system admin noticed an unusual process running on the server. The system admin has collected the file (host.exe) associated with that process and sent it you for further analysis. Analyze the file and answer the below questions

- What is the file type?
- Determine the cryptographic hash
- Is the file packed/obfuscated?
- Are there any interesting strings?
- Which malware imports suggest the use of network activity?
- Is the file malicious?