

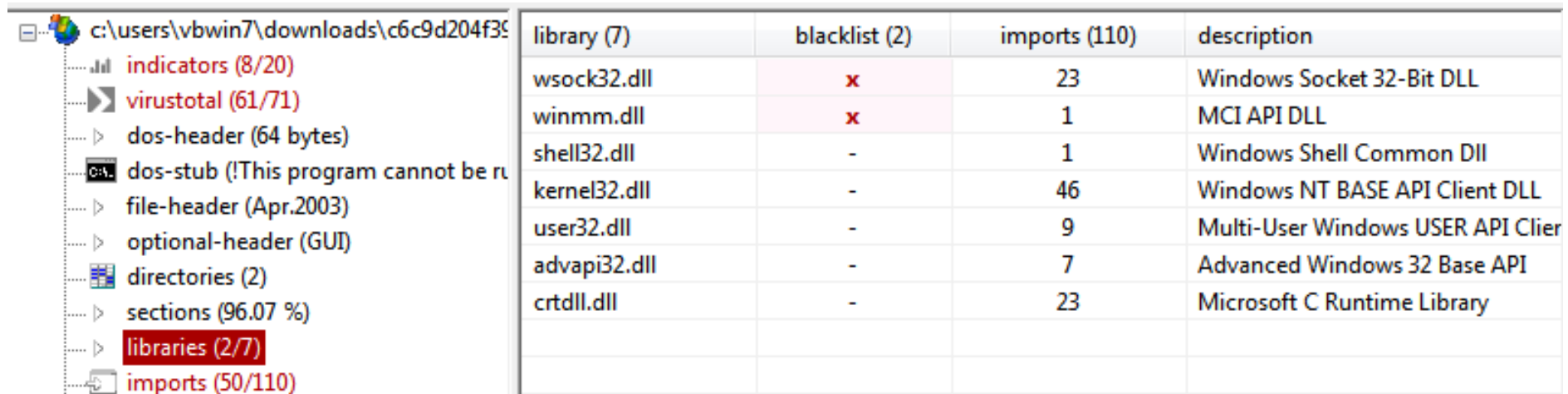
# Malware Analysis Techniques

- **Static analysis**

- Studying the malware **without actually executing it**.
- Static analysis involves extracting useful information from the suspect binary to make informed **decision on how to classify or analyse it** and where to focus the subsequent analysis on.
- The biggest advantage of this approach is that there is **little chance of the malware executing and escaping the system** it is analysed on. (It is safe.)
- The disadvantage is that without executing the suspect binary, it is sometime hard to fully analyse its behaviour.
- **pestudio** is a useful tool for inspecting PE (portable executable) file (e.g, .exe, .dll, .sys, .com, .ocx etc) dependencies.

# Static Analysis

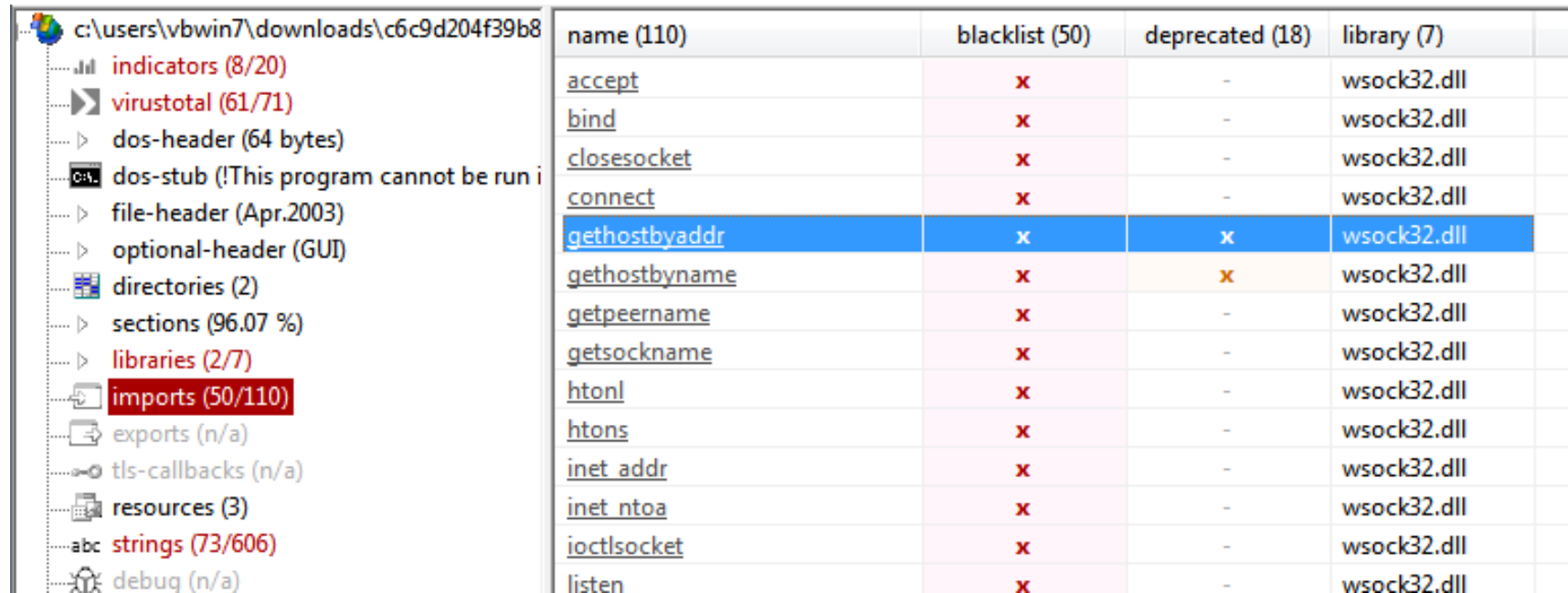
- **pestudio** is a useful tool for inspecting file dependencies: In this example of the malware spybot.exe, we can see what DLLs this executable depends on by clicking 'libraries' button. Note that wsock32.dll, which is related to networking, is blacklisted. (Indeed the malware tries to connect to a malicious server.)



library (7)	blacklist (2)	imports (110)	description
wsock32.dll	x	23	Windows Socket 32-Bit DLL
winmm.dll	x	1	MCI API DLL
shell32.dll	-	1	Windows Shell Common Dll
kernel32.dll	-	46	Windows NT BASE API Client DLL
user32.dll	-	9	Multi-User Windows USER API Clier
advapi32.dll	-	7	Advanced Windows 32 Base API
crtdll.dll	-	23	Microsoft C Runtime Library

# Static Analysis

- By clicking the "imports" button, we can view a list of APIs functions that the blacklisted DLL imports.



name (110)	blacklist (50)	deprecated (18)	library (7)
<a href="#">accept</a>	x	-	wsock32.dll
<a href="#">bind</a>	x	-	wsock32.dll
<a href="#">closesocket</a>	x	-	wsock32.dll
<a href="#">connect</a>	x	-	wsock32.dll
<a href="#">gethostbyaddr</a>	x	x	wsock32.dll
<a href="#">gethostbyname</a>	x	x	wsock32.dll
<a href="#">getpeername</a>	x	-	wsock32.dll
<a href="#">getsockname</a>	x	-	wsock32.dll
<a href="#">htonl</a>	x	-	wsock32.dll
<a href="#">htons</a>	x	-	wsock32.dll
<a href="#">inet_addr</a>	x	-	wsock32.dll
<a href="#">inet_ntoa</a>	x	-	wsock32.dll
<a href="#">ioctlsocket</a>	x	-	wsock32.dll
<a href="#">listen</a>	x	-	wsock32.dll

# Static Analysis

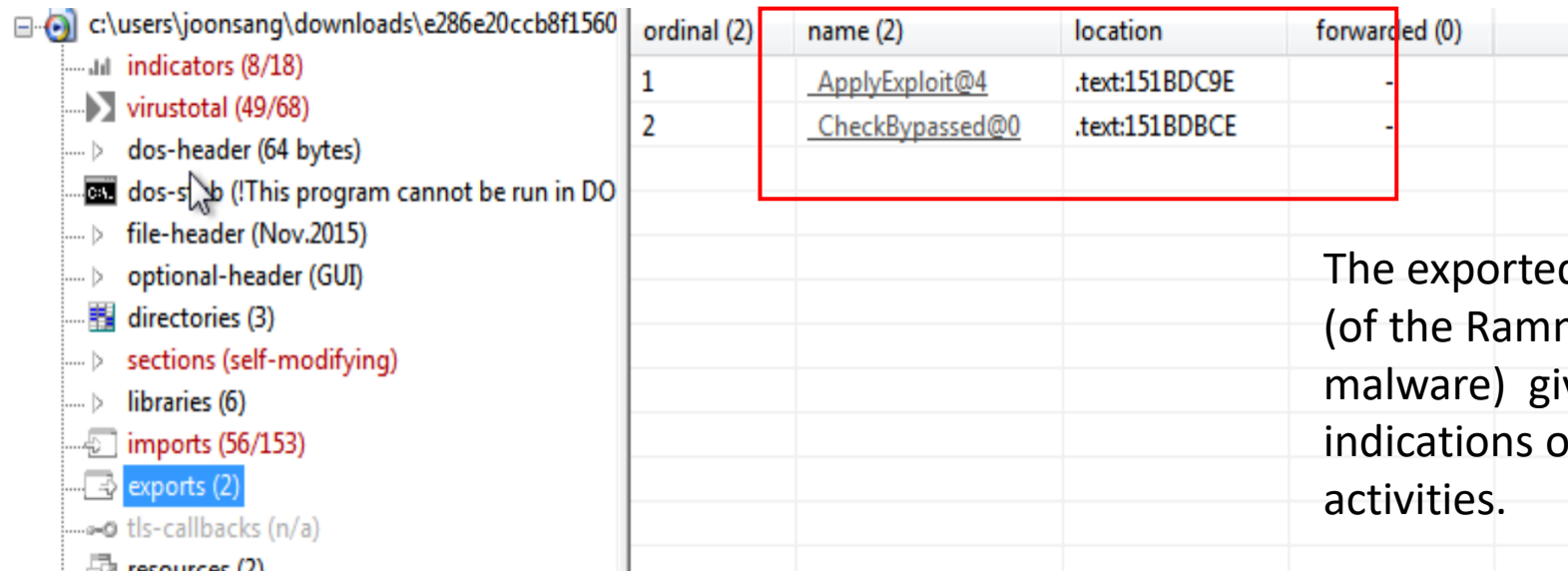
- Inspecting exports

- The executable and DLL can export functions, which can be used by other programs.
- An attacker often creates a DLL that exports functions containing malicious functionality. Exported functions can be used to indicate the malware's capability.

ordinal (2)	name (2)	location	forwarded (0)
1	<u>ApplyExploit@4</u>	.text:151BDC9E	-
2	<u>CheckBypassed@0</u>	.text:151BDBCE	-

# Static Analysis

- Inspecting exports



ordinal (2)	name (2)	location	forwarded (0)
1	<u>_ApplyExploit@4</u>	.text:151BDC9E	-
2	<u>_CheckBypassed@0</u>	.text:151BDBCE	-

The exported functions  
(of the Ramnit  
malware) give  
indications of malicious  
activities.

# Static Analysis

- Examining PE section table and sections
  - **The actual content of the executable files is divided into sections.** The sections are followed by the PE header. These sections represent either code or data and they have in-memory attributes such as read/write.
    - ✓ Section name `.text` or `CODE`: It contains executable code.
    - ✓ Section name `.data` or `DATA`: It contains read/write data and global variables.
    - ✓ Section name `.rdata` : It contains read-only data.
    - ✓ Section name `.idata`: It contains the import table.
    - ✓ Section name `.edata`: It contains export information.
    - ✓ Section name `.rsrc`: It contains the resources used by the executable such as icon, menus, strings, etc.

# Static Analysis

- It is possible for an attacker or an obfuscation software to create sections with different names.
  - ✓ If uncommon section names are encountered, the given binary is suspicious and further analysis should be performed.

\\users\\vbwin7\\desktop\\spybot\_packed.exe

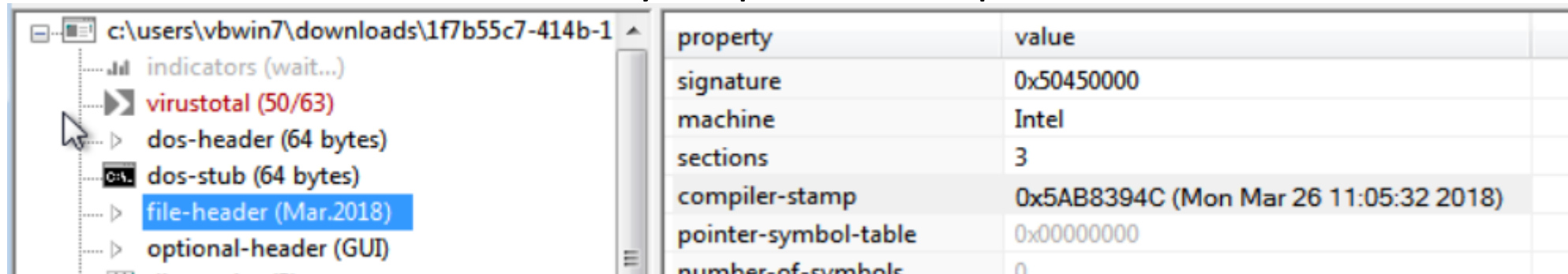
- indicators (9/23)
- virustotal (warning)
- > dos-header (64 bytes)
- dos-stub (!This program cannot be run in DO
- > file-header (Apr.2003)
- > optional-header (GUI)
- directories (2)
- > **sections (self-modifying)**
- > libraries (2/7)
- imports (4/10)
- exports (n/a)
- tls-callbacks (n/a)
- resources (3)
- bc strings (6/400)
- debug (n/a)

property	value	value
name	UPX0	UPX1
md5	n/a	B0F275CF7C9226F690ECE0D..
file-ratio (97.35 %)	0.00 %	77.38 %
file-cave (0 bytes)	n/a	15872 bytes
entropy	n/a	7.880
raw-address	0x00000200	0x00000200
raw-size (19968 bytes)	0x00000000 (0 bytes)	0x00003E00 (15872 bytes)
virtual-address	0x00401000	0x00414000
virtual-size (98304 bytes)	0x00013000 (77824 bytes)	0x00004000 (16384 bytes)
entry-point (0x00017C40)	-	x
writable	x	x
executable	x	x
shareable	-	-

In this example, the names UPX0, UPX1 are not conventional. It can be inferred that the malware was obfuscated using UPX (packer).

# Static Analysis

- Examining the compilation timestamp
  - The PE header contains information that specifies when the binary was compiled; examining the field can give an idea of when the malware was first created. This information can be useful in building a timeline of the attack campaign.
  - It is also possible that an attacker modifies the timestamp to prevent an analysis from knowing the actual timestamp. A compile timestamp can sometimes be used to classify suspicious samples.



The screenshot shows a file explorer window with the path `c:\users\vbwin7\downloads\1f7b55c7-414b-1`. The directory tree on the left includes 'indicators (wait...)', 'virustotal (50/63)', 'dos-header (64 bytes)', 'dos-stub (64 bytes)', 'file-header (Mar.2018)' (highlighted), and 'optional-header (GUI)'. To the right, a table displays the following properties and values:

property	value
signature	0x50450000
machine	Intel
sections	3
compiler-stamp	0x5AB8394C (Mon Mar 26 11:05:32 2018)
pointer-symbol-table	0x00000000
number-of-symbols	0



# Static Analysis

- Examining resources

- The resources required by the executable file such as icons, menu, dialog and strings are stored in the resources section (.rsrc) of an executable file.
- Often, attackers store information such as additional binary, decoy documents and configuration data in the resource section.
  - ✓ For example, a fake icon for hiding the malicious executable can be stored.
- Examining the resource can reveal valuable information about the origin, company name, program author details and copyright information.

# Static Analysis

[illegible]

In this example, PE "resources" are present. In fact, this malware is using a fake (Windows) icon looking like this:



# Malware Analysis Techniques

- **Dynamic analysis**

- The other popular analysis method for malware is dynamic analysis.
- Dynamic analysis lets the malware run in a jailed environment or sandbox.
  - ✓ Virtual machines, if configured correctly, can provide a sandbox environment.
- Dynamic analysis is sometimes performed after static analysis.
- A sandbox is a stand-alone environment that allows safely viewing or executing the program while keeping it contained.
- When malware is run in a sandbox, there is always some possibility that it may escape and infect other systems.

# Malware Analysis Techniques

## ➤ Advantage

- ✓ Dynamic analysis gives analysers clear information about the behaviour of malware.

## ➤ Disadvantage (limitation)

- ✓ Recent malware is **often written with anti-sandbox** safeguards. - Even when the sandbox does work, it only reports functionality and may not report everything that the malware does.
- ✓ Dynamic analysis should be conducted with process analysis and network traffic analysis to get the best result.