alienvault.com

Reverse Engineering Malware

9-11 minutes

The Alien Labs team does a lot of malware analysis as a part of their security research. I interviewed a couple members of our Labs team, including Patrick Snyder, Eddie Lee, Peter Ewane and Krishna Kona, to learn more about how they do it.

Here are some of the approaches and tools and techniques they use for reverse engineering malware, which may be helpful to you in your own malware hunting endeavors. Please watch the webcast they did recently with Javvad Malik on reverse engineering malware and hear details and examples of how the Labs team investigated OceanLotus, PowerWare and Linux malware in recent situations.

Approaches in reverse engineering a malware sample

- Reverse engineer: The most obvious approach is to completely reverse engineer a piece of malware. This obviously takes a great amount of time, so other approaches are more practical.
- Exploitation techniques: Another approach you can take is to focus on the exploitation techniques of a piece of malware. Occasionally you will see a piece of malware that is using a new

exploitation technique, or is exploiting a zero-day vulnerability. In this case you may be interested only in the specific exploitation technique so you can timebox your analysis and only look at the exploitation mechanisms.

- Obfuscation: Malware will often obfuscate itself and make itself difficult to analyze. You might come across malware that you have seen before without obfuscation. In that case you may only want to focus on reverse engineering the new parts.
- Encryption methods: A common type of malware these days is ransomware. Ransomware essentially encrypts the victim's files and locks them up so that they can't be accessed or read. Oftentimes the authors of ransomware will make mistakes when they implement the encryption mechanisms. So if you focus your research on the encryption mechanisms you might be able to find weaknesses in their implementation and/or you might be able to find hard-coded keys or weak algorithms.
- C&C communication: This is something that is pretty commonly done when looking at malware. Analysts often want to figure out what the communication protocol is between a piece of malware on the client's side and the server on the command and control side. The communication protocol can actually give you a lot of hints about the malware’s capabilities.
- Attribution: Murky area - kind of like a dark art. It usually involves a lot of guesswork, knowledge of malicious hacking teams and looking at more than one piece of malware.
- Categorization and clustering: You can reverse engineer malware from a broader point of view. This involves looking at malware in

bulk and doing a broad-stroke analysis on lots of different malware, rather than doing a deep dive.

Techniques

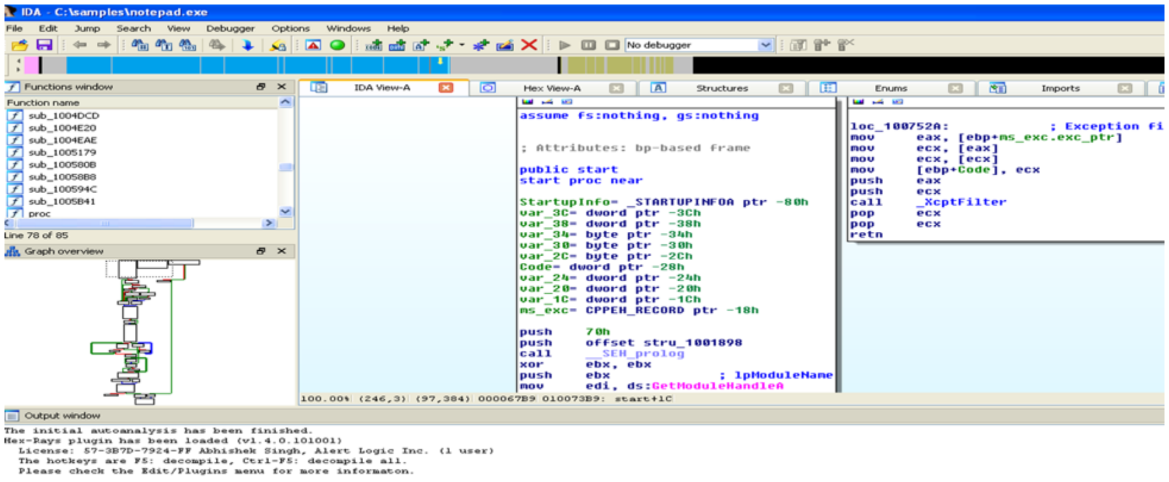
Now, let’s look at techniques that can be utilized while analyzing malware.

- First of all, we use **static analysis.** This is the process of analyzing malware or binaries without actually running them. It can be as simple as looking at metadata from a file. It can range from doing disassembly or decompilation of malware code to symbolic execution, which is something like virtual execution of a binary without actually executing it in a real environment.
- Conversely, **dynamic analysis** is the process of analyzing a piece of malware when you are running it in a live environment. In this case, you are often looking at the behavior of the malware and looking at the side effects of what it is doing. You are running tools like process monitor and sysmon to see what kinds of artifacts a piece of malware produces after it is run.
- We also use **automated analysis**. Oftentimes if you are looking at malware you want to automate things just to speed up the process to save time. However, use caution, as with automated analysis sometimes things get missed because you are trying to do things generically.
- If a piece of malware contains things like anti-debugging routines or anti-analysis mechanisms, you may want to perform a **manual analysis**. You need to pick the right tools for the job.

Tools

- IDA Pro is a really good tool for analyzing various samples of malware with diverse backgrounds. It also has a good add-on called HEX Rays Decompiler, which is a tool that can convert assembly language into more easily read pseudocode. It can help you in understanding the functionality of the code more quickly than looking at assembly language. When you open a sample in IDA Pro, you see the entry point of the malware. It has a graph view as well and you can switch between both hex code and the graph view. It will give you a quick representation of the mapping of the flow of execution as well. It has an SDK you can use if you like to develop plug-ins and automate and extract some of the useful information. IDA Pro also has a Python API that you can use if you prefer Python. The tool also has debugging functionality, but mainly it is used for static reverse engineering of malware.

  Here's IDA Pro:



```
assume fs:nothing, gs:nothing

; Attributes: bp-based frame

public start
start proc near

StartupInfo= _STARTUPINFOA ptr -80h
var_3C= dword ptr -3Ch
var_38= dword ptr -38h
var_34= byte ptr -34h
var_30= byte ptr -30h
var_2C= byte ptr -2Ch
Code= dword ptr -28h
var_24= dword ptr -24h
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
ms_exc= CPPEH_RECORD ptr -18h

push    70h
push    offset stru_1001898
call    __SEH_prolog
xor     ebx, ebx
push    ebx             ; lpModuleName
mov     edi, ds:GetModuleHandleA
```

```
loc_100752A:                ; Exception fi
mov     eax, [ebp+ms_exc.exc_ptr]
mov     ecx, [eax]
mov     ecx, [ecx]
mov     [ebp+Code], ecx
push    eax
push    ecx
call    _XcptFilter
pop     ecx
pop     ecx
retn
```

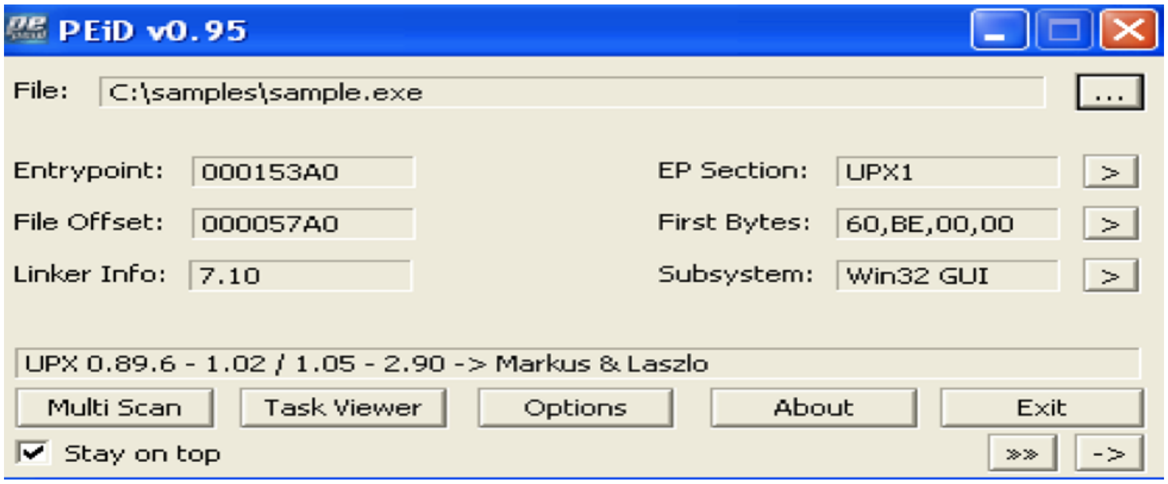
```
100.00% (246,3) (97,384) 000067B9 010073B9: start+1C
```

```
The initial autoanalysis has been finished.
Hex-Rays plugin has been loaded (v1.4.0.101001)
  License: 57-3B7D-7924-FF Abhishek Singh, Alert Logic Inc. (1 user)
  The hotkeys are F5: decompile, Ctrl-F5: decompile all.
  Please check the Edit/Plugins menu for more informaton.
```

- There are also debuggers like The GNU Project Debugger (GDB), WinDbg and Wind River.

- For Windows samples, PEiD, PEStudio, PE32 tools are great. You can also use these tools on executables and get some initial classification of your samples.

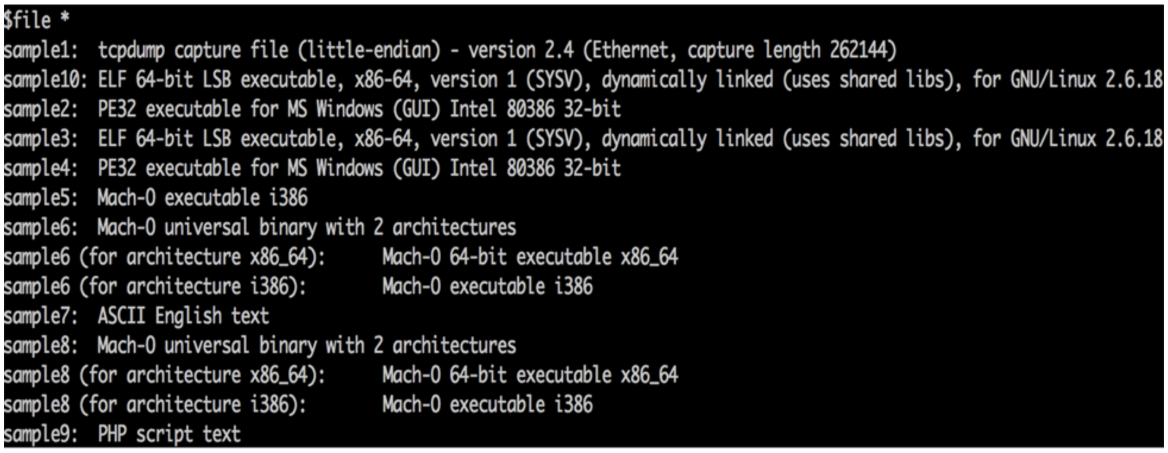
  Here's PEiD:



```
File: C:\samples\sample.exe

Entrypoint: 000153A0          EP Section: UPX1
File Offset: 000057A0         First Bytes: 60,BE,00,00
Linker Info: 7.10             Subsystem: Win32 GUI

UPX 0.89.6 - 1.02 / 1.05 - 2.90 -> Markus & Laszlo

[✓] Stay on top
```

- Other tools that we use include strings, file, and otool. They help us initially find the platform of the sample. If you look at some snapshots of these tools, they can tell you where the entry point of that sample is, what section, and if the sample is packed. They can also detect more than a hundred packers and can detect decrypters and compilers as well.

  Here's the file utility:

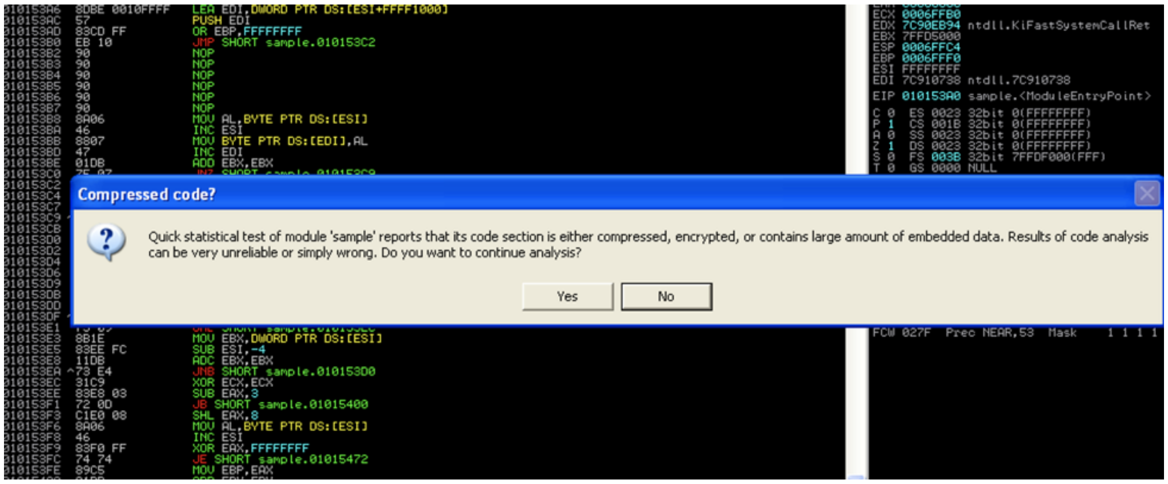


```
$file *
sample1:  tcpdump capture file (little-endian) - version 2.4 (Ethernet, capture length 262144)
sample10: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.18
sample2:  PE32 executable for MS Windows (GUI) Intel 80386 32-bit
sample3:  ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.18
sample4:  PE32 executable for MS Windows (GUI) Intel 80386 32-bit
sample5:  Mach-O executable i386
sample6:  Mach-O universal binary with 2 architectures
sample6 (for architecture x86_64):      Mach-O 64-bit executable x86_64
sample6 (for architecture i386):        Mach-O executable i386
sample7:  ASCII English text
sample8:  Mach-O universal binary with 2 architectures
sample8 (for architecture x86_64):      Mach-O 64-bit executable x86_64
sample8 (for architecture i386):        Mach-O executable i386
sample9:  PHP script text
```

Generally, when we get a bunch of samples or an archive of samples from open-source feed, we use a file utility to find out if the file is a regular executable or for a Windows platform or OSX or Linux, or is it just a text file or a script.

- Immunity Debugger is another popular debugger. If you open a sample in Immunity Debugger, it will give you an alert saying that the sample is packed and be asked if you want to proceed with the analysis. If you continue the analysis, you see an entry point, which is where it pushes all of the registers to stack. You can use this debugger to step through the execution of the sample to see the unpacked sample in memory. You can continue analyzing the samples step-by-step and use the debuggers in the tool for finding the malware's activities and the effects it has on the system.

Here's Immunity Debugger:

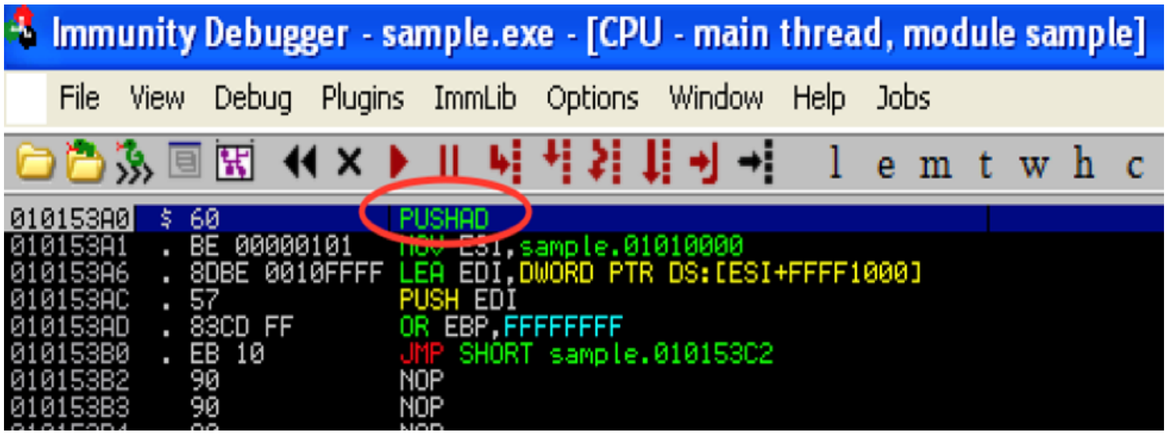


```
010153A6  8DBE 0010FFFF  LEA EDI,DWORD PTR DS:[ESI+FFFF1000]
010153AC  57             PUSH EDI
010153AD  83CD FF        OR EBP,FFFFFFFF
010153B0  EB 10          JMP SHORT sample.010153C2
010153B2  90             NOP
010153B3  90             NOP
010153B4  90             NOP
010153B5  90             NOP
010153B6  90             NOP
010153B7  90             NOP
010153B8  8A06           MOV AL,BYTE PTR DS:[ESI]
010153BA  46             INC ESI
010153BB  8807           MOV BYTE PTR DS:[EDI],AL
010153BD  47             INC EDI
010153BE  01DB           ADD EBX,EBX
010153C0
010153C2
010153C4
010153C7
010153C9
010153CB
010153D0
010153D2
010153D4
010153D6
010153D9
010153DB
010153DD
010153DF
010153E1
010153E3  8B1E           MOV EBX,DWORD PTR DS:[ESI]
010153E5  83EE FC        SUB ESI,-4
010153E8  11DB           ADC EBX,EBX
010153EA ^73 E4          JNB SHORT sample.010153D0
010153EC  31C9           XOR ECX,ECX
010153EE  83E8 03        SUB EAX,3
010153F1  72 0D          JB SHORT sample.01015400
010153F3  C1E0 08        SHL EAX,8
010153F6  8A06           MOV AL,BYTE PTR DS:[ESI]
010153F8  46             INC ESI
010153F9  83F0 FF        XOR EAX,FFFFFFFF
010153FC  74 74          JE SHORT sample.01015472
010153FE  89C5           MOV EBP,EAX
```

```
ECX 0006FFB0
EDX 7C90EB94 ntdll.KiFastSystemCallRet
EBX 7FFD5000
ESP 0006FFC4
EBP 0006FFF0
ESI FFFFFFFF
EDI 7C910738 ntdll.7C910738
EIP 010153A0 sample.<ModuleEntryPoint>
C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
FCW 027F  Prec NEAR,53  Mask    1 1 1 1
```

Compressed code?

Quick statistical test of module 'sample' reports that its code section is either compressed, encrypted, or contains large amount of embedded data. Results of code analysis can be very unreliable or simply wrong. Do you want to continue analysis?



Immunity Debugger - sample.exe - [CPU - main thread, module sample]

```
010153A0  $ 60           PUSHAD
010153A1  . BE 00000101  MOV ESI,sample.01010000
010153A6  . 8DBE 0010FFFF LEA EDI,DWORD PTR DS:[ESI+FFFF1000]
010153AC  . 57           PUSH EDI
010153AD  . 83CD FF      OR EBP,FFFFFFFF
010153B0  . EB 10        JMP SHORT sample.010153C2
010153B2    90           NOP
010153B3    90           NOP
```

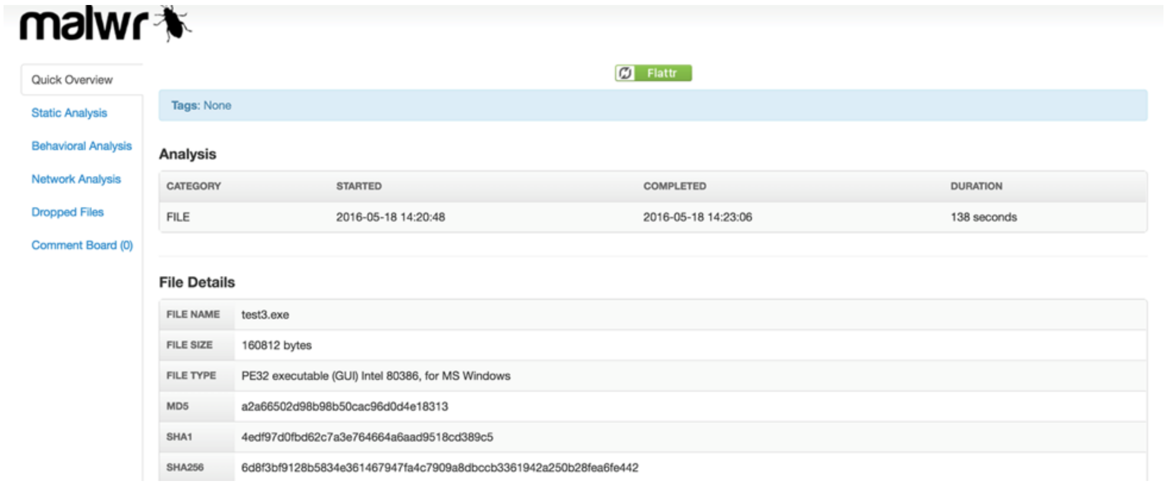
```
010153B4    90          NOP
010153B5    90          NOP
```

For capturing network traffic, we use [Wireshark]() or [TCPDump]().

For monitoring the activity on the system, we use system monitor and [Regshot]().

Sandboxes are another important step in reverse engineering malware, as often there are functionalities malware doesn't exhibit unless it is running in a suitable environment. One sandbox, [malwr](), comes from the people who built Cuckoo Sandbox. With malwr, you submit a sample and run it inside a VM. You can then run various dynamic analysis tools and static analysis tools referenced above and turn this into a nice, readable report.

Here is malwr:



Tags: None

Analysis

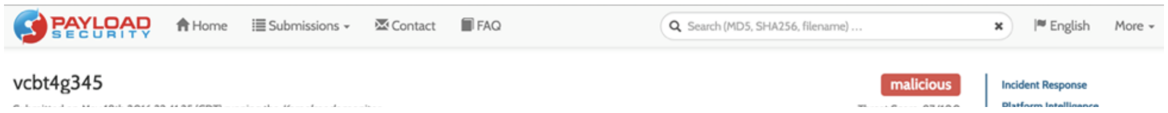
CATEGORY	STARTED	COMPLETED	DURATION
FILE	2016-05-18 14:20:48	2016-05-18 14:23:06	138 seconds

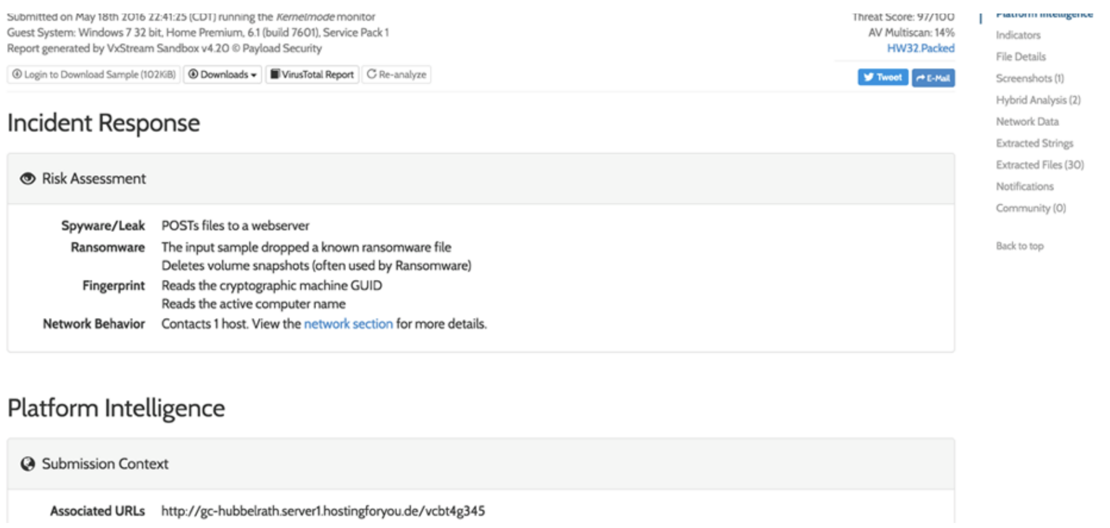
File Details

FILE NAME	test3.exe
FILE SIZE	160812 bytes
FILE TYPE	PE32 executable (GUI) Intel 80386, for MS Windows
MD5	a2a66502d98b98b50cac96d0d4e18313
SHA1	4edf97d0fbd62c7a3e764664a6aad9518cd389c5
SHA256	6d8f3bf9128b5834e361467947fa4c7909a8dbccb3361942a250b28fea6fe442

- Another Sandbox that is relatively new is [Hybrid-Analysis](). It is made by Payload Security and it functions very similar to malwr, but they have some of their own custom sandboxes running that may or may not be based on Cuckoo.

Here is Hybrid-Analysis:





Submitted on May 18th 2016 22:41:25 (CDT) running the *Kernelmode* monitor
Guest System: Windows 7 32 bit, Home Premium, 6.1 (build 7601), Service Pack 1
Report generated by VxStream Sandbox v4.20 © Payload Security

Threat Score: 97/100
AV Multiscan: 14%
HW32.Packed

Incident Response

Risk Assessment

Spyware/Leak	POSTs files to a webserver
Ransomware	The input sample dropped a known ransomware file
	Deletes volume snapshots (often used by Ransomware)
Fingerprint	Reads the cryptographic machine GUID
	Reads the active computer name
Network Behavior	Contacts 1 host. View the network section for more details.

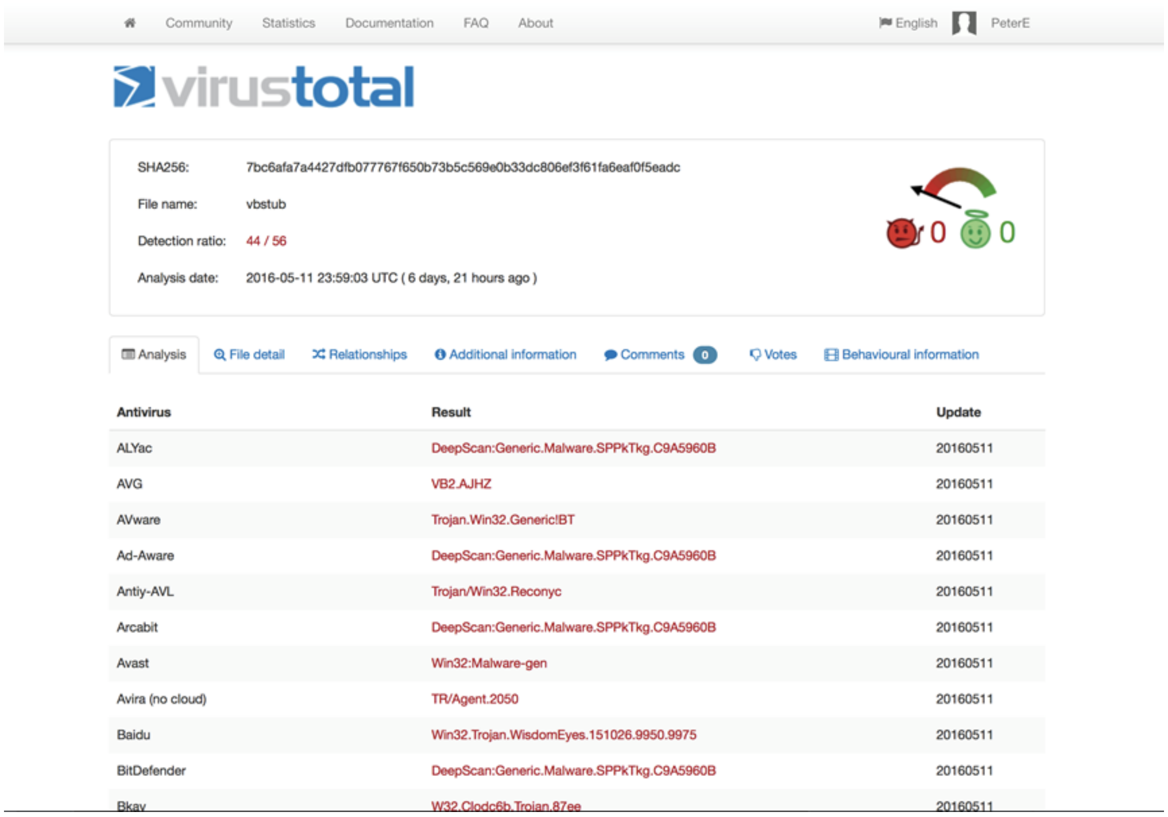
Platform Intelligence

Submission Context

Associated URLs http://gc-hubbelrath.server1.hostingforyou.de/vcbt4g345

Another major Sandbox tool for identifying malware is VirusTotal. VirusTotal is owned by Google, and they arguably have the biggest repository of both malware and known file types in general layout. If you are looking for any particular malware, it typically shows up in VirusTotal.

Here is VirusTotal:

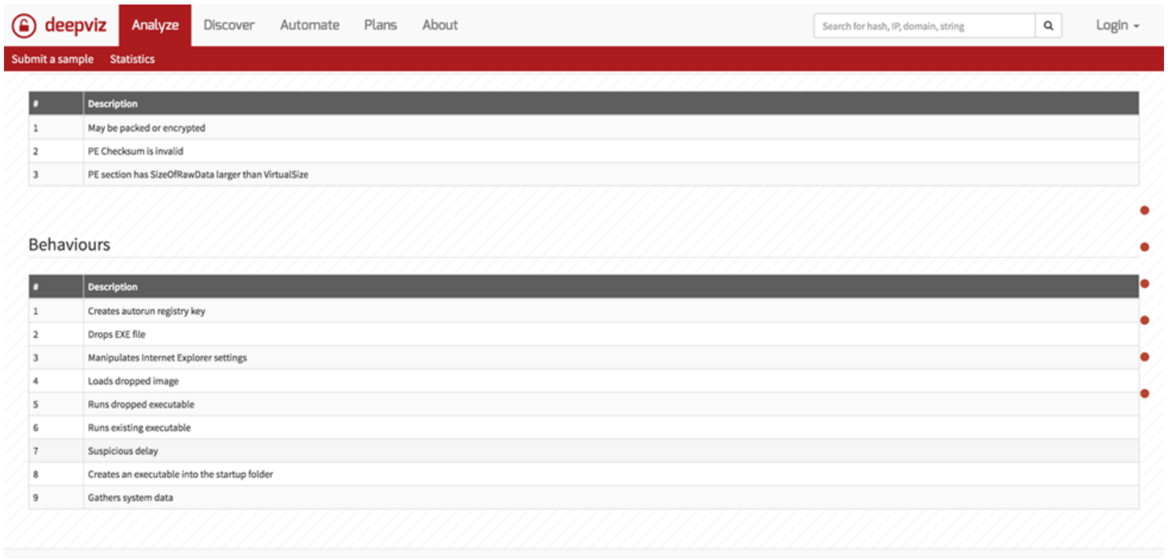


SHA256:	7bc6afa7a4427dfb077767f650b73b5c569e0b33dc806ef3f61fa6eaf0f5eadc
File name:	vbstub
Detection ratio:	44 / 56
Analysis date:	2016-05-11 23:59:03 UTC (6 days, 21 hours ago)

Antivirus	Result	Update
ALYac	DeepScan:Generic.Malware.SPPkTkg.C9A5960B	20160511
AVG	VB2.AJHZ	20160511
AVware	Trojan.Win32.Generic!BT	20160511
Ad-Aware	DeepScan:Generic.Malware.SPPkTkg.C9A5960B	20160511
Antiy-AVL	Trojan/Win32.Reconyc	20160511
Arcabit	DeepScan:Generic.Malware.SPPkTkg.C9A5960B	20160511
Avast	Win32:Malware-gen	20160511
Avira (no cloud)	TR/Agent.2050	20160511
Baidu	Win32.Trojan.WisdomEyes.151026.9950.9975	20160511
BitDefender	DeepScan:Generic.Malware.SPPkTkg.C9A5960B	20160511
Bkav	W32.Clodc6b.Trojan.87ee	20160511

Another new contender is DeepViz. DeepViz is being developed very actively, with new features on a regular basis. DeepViz functions very similarly to other Sandboxes, but sometimes it is beneficial to submit the same sample to multiple sandboxes to see if the behavior matches up or if it reacts differently.

Here is DeepViz:



#	Description
1	May be packed or encrypted
2	PE Checksum is invalid
3	PE section has SizeOfRawData larger than VirtualSize

Behaviours

#	Description
1	Creates autorun registry key
2	Drops EXE file
3	Manipulates Internet Explorer settings
4	Loads dropped image
5	Runs dropped executable
6	Runs existing executable
7	Suspicious delay
8	Creates an executable into the startup folder
9	Gathers system data

Which brings us to Cuckoo. Cuckoo is a malware analysis system. It contains many different tools, including some of the dynamic and static analysis tools that we mentioned earlier. Also, it is free. While other sandboxes are free, you are sharing your data by using them. If you set up Cuckoo on your own system you can keep everything localized and keep it to yourself, especially if you are analyzing something you don't want the world to know about yet.

Here is Cuckoo:

To find out more about OTX there is a documentation center. You can also see information on our forums. There is a section specifically for OTX where you can see pulses. Also, just a few weeks ago we announced some enhancements to the OTX API. If you are a blogger, please note you can now embed pulses. So if you write a blog, you can just simply embed it within so users can read it and directly download the IoCs and other information. Read more.

Connecting OTX to your USM platform helps you to manage risk better and effectively take action on threats. A free trial of AlienVault USM is available.

About the Author: Kate Brew

Kate has over 15 years experience in product management and marketing, primarily in information security.

Read more posts from Kate Brew ›