

02-Basic Static Analysis

CYS5120 - Malware Analysis

Bahcesehir University
Cyber Security Msc Program

Dr. Ferhat Ozgur Catak ¹ Mehmet Can Doslu ²

¹ozgur.catak@tubitak.gov.tr

²mehmetcan.doslu@tubitak.gov.tr

2017-2018 Fall

Table of Contents

- 1 Malware Analysis Methods
 - General Analysis Methods
 - Dynamic vs Static
 - Analysis Process
 - Key Points
 - Analysis Blocking Methods
 - Malware Analysis Environment Setup
- 2 Introduction
 - Basic Static Techniques
 - Antivirus
 - Hash Control
 - Review The Text of a File
 - Packed and Obfuscated Malware
 - Packing Files
- 3 Windows OS
 - Portable Executable File Format
 - Tools for Windows
 - Conclusion
- 4 Linux
 - Tools
 - REMnux
- 5 Lab
 - Lab

Table of Contents

- 1 **Malware Analysis Methods**
 - General Analysis Methods
 - Dynamic vs Static
 - Analysis Process
 - Key Points
 - Analysis Blocking Methods
 - Malware Analysis Environment Setup

- 2 Introduction
 - Basic Static Techniques
 - Antivirus
 - Hash Control
 - Review The Text of a File

- Packed and Obfuscated Malware
- Packing Files

- 3 Windows OS
 - Portable Executable File Format
 - Tools for Windows
 - Conclusion

- 4 Linux
 - Tools
 - REMnux

- 5 Lab
 - Lab

General Analysis Methods

General Analysis Methods

There are two different malware analysis approach.

Static Analysis

Static analysis basically has two different methods.

- ▶ Signature Search
 - ▶ Signature-based technologies track known threats.
 - ▶ Algorithms can quickly and efficiently scan an object to determine its digital signature.
- ▶ Code analysis

Dynamic Analysis

- ▶ Behaviour Analysis
 - ▶ An object's behavior is analyzed for suspicious activities.
 - ▶ Attempts to perform actions that are clearly abnormal or unauthorized would indicate the object is malicious/suspicious.
 - ▶ Examples: attempt to discover a sandbox environment, disabling security controls, installing rootkits, registering for autostart.

Dynamic vs Static

Dynamic Analysis

- ▶ Fast and relatively easy
- ▶ Only one working path can be seen

Static Analysis

- ▶ Further investigation possible
- ▶ Slow and difficult

Result

- ▶ It is necessary to use two together!

Analysis Process

Analysis Process

- ▶ You can start with your choice.
- ▶ Recommended process
 - ▶ Basic Static
 - ▶ Basic Dynamic
 - ▶ Advance Static ⇒ Advance Dynamic

Key Points for Analysis

Key Points

- ▶ Don't focus in details, focus on the goal
- ▶ There are multiple ways to reach your analysis goals and multiple tools that can be used. If one does not work, use the other.
- ▶ Analysis is a race, *new techniques will be released every day*, be prepared for it.

Analysis Blocking Methods I

The Development of Malware

- ▶ It is valuable for the malware producer to **hide the software** instead of exposing itself to the user.
- ▶ The **evolution of malware** follows the steadily increasing amount of Internet nodes, and so does the level of sophisticated techniques **malware uses to multiply it self and camouflage its functionality**.

Reasons for obfuscating malware includes, (not limited to), the following:

- ▶ **Preventing code analysis** Suspicious code instructions are assets to look for during malware analysis. Obfuscating code disguises the actual malware content and hides the instructions from the analysts.
- ▶ **Prolong the dissection process** Obfuscating code complicates and dwell out malware analyses, and thus increases the life span of the malware
- ▶ **Evade detection** Each different sample is called a **variance** of the original form, and changing appearance makes detection more difficult for applications that use signatures to match files, such as antivirus applications.

Analysis Blocking Methods II

Analysis Blocking Methods

- ▶ Malware writers have taken code obfuscation even one step further, and resorted to **self mutating** programs that can more efficiently evade detection and complicate analyses.
- ▶ The malware changes itself dynamically either during running time, while being dormant in a computer or upon infecting another host.
- ▶ Malware using such techniques
 - ▶ Oligomorphic Mutation
 - ▶ Polymorphic Mutation
 - ▶ Metamorphic Mutation

Oligomorphic Mutation

Oligomorphic Mutation ¹

- ▶ One of the first and simplest methods used to obfuscate malware was to encrypt its content
- ▶ The malware must be **exposed** in its original form upon execution, and a **decryption key** is used to unlock the malware's original content.
- ▶ The algorithm used for decryption is called the **decryptor**, and is in the *simplest case static in each copy of the sample*.
 - ▶ *Reverse Engineering* techniques can be applied to find the *decryption key* and the *decryptor*

Definition

- ▶ If the sample is capable of generating a multiple decryptors n , where $n > 1$ but still a *limited* set, it is capable of an *oligomorphic* mutation
- ▶ The malware is able to generate different variances of itself upon mutation, and n different signatures must be made to detect it using signature based detection techniques.

¹https://brage.bibsys.no/xmlui/bitstream/handle/11250/251364/348799_FULLTEXT01.pdf ▶

Polymorphic Mutation

Polymorphic Mutation

- ▶ Polymorphic malware can dynamically change the available decryptors to *an extensive number*, making an large amount of actually used decryption methods
- ▶ The polymorphic mutation makes it **practically impossible** to *create signatures* for all variances.



(a) Oligomorphic and polymorphic malware mutating itself while spreading. For oligomorphic malware, there are n fixed different decryptors chosen by the malware producer making a set of n different variances of the malware. For polymorphic malware, the amount of decryptors is arbitrary large and generated automatically.

Metamorphic Mutation

Metamorphic mutation

- ▶ Instead of changing *decryptors* and *encryption methods*, metamorphic malware **mutates its own body**, usually upon propagation.
- ▶ In some cases, a metamorphic malware sample can *decompile itself*, change its body in source code form and recompile to generate a completely new variance of itself



(b) Metamorphic malware mutating itself while spreading using a variety of code transformation techniques. The sample m_0 appears completely different from its mutated variance m_1 , but the functionality remains untouched.

Malware Analysis Environment Setup

Malware Analysis Environment Setup

- ▶ Virtual or Physical Machine
- ▶ Internet Connection
- ▶ Tools
 - ▶ Process Explorer, Process Hacker
 - ▶ Process Monitor, CaptureBAT, Regshot
 - ▶ WireShark, Network Monitor
 - ▶ FakeNet, Honeyd, netcat

Table of Contents

- 1 Malware Analysis Methods
 - General Analysis Methods
 - Dynamic vs Static
 - Analysis Process
 - Key Points
 - Analysis Blocking Methods
 - Malware Analysis Environment Setup
- 2 Introduction
 - Basic Static Techniques
 - Antivirus
 - Hash Control
 - Review The Text of a File

- Packed and Obfuscated Malware
- Packing Files
- 3 Windows OS
 - Portable Executable File Format
 - Tools for Windows
 - Conclusion
- 4 Linux
 - Tools
 - REMnux
- 5 Lab
 - Lab

Basic Static Techniques

Basic Static Techniques

- ▶ **The first step** in Malware Analysis is to perform a **static analysis** of the malware. Static analysis is necessary for analyzing **the code** or for analyzing **its structure** and for determining **its functionality**.
- ▶ The malware **does not run** in the static analysis phase. The stage in which the code is run is **dynamic analysis**.

Techniques

- ▶ Detection of malware with an antivirus
- ▶ Checking the hash values of files
- ▶ Obtaining information by examining the **texts**, **functions** and **titles** of the malware

Detection of Malware With an Antivirus

Detection with an Antivirus

- ▶ They detect by comparing file signatures with malware software database.
- ▶ This can be overcome with **any file signature change** in the malware.
- ▶ By using sites such as **VirusTotal**, malware can be scanned in **more than one antivirus engine**.



Checking File Signatures

Checking File Signatures

- ▶ **Hashing** is a commonly used method for detecting malware. **MD5** and **SHA1** are some hash algorithms used for malware detection.
- ▶ FCIV : File Checksum Integrity Verifier
- ▶ Alternative: **WinMD5 GUI**
- ▶ It can be searched on the internet for hash values to determine if there is harmful content in the file.

```
$ fciv -md5 -sha1 malware.exe  
$ md5sum malware.exe  
$ shasum malware.exe
```

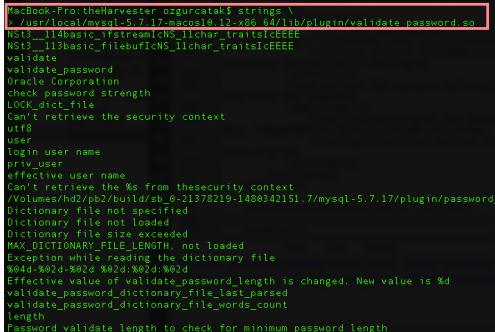
Review The Text of a File

- ▶ A part of the content of a program consists of **texts**.
 - ▶ These texts may be related to a **message**, **URL link**, or **file copy**.
 - ▶ Such analyzes also make it easier to understand the function of the program.
- ▶ Any text that contains malicious software can be understood when examined.
 - ▶ There are programs for examining this type of data, independent of ASCII or Unicode text format.
 - ▶ Significant text data can be displayed via these programs.

Review The Text of a File II

Linux OS - strings

- ▶ **Linux:** *strings* command is used to get text values from executable files.
- ▶ **Windows:** *BinText*

A terminal window with a black background and green text. The command 'strings \n /usr/local/mysql-5.7.17-macos10.12-x86_64/lib/plugin/validate_password.so' is entered and highlighted with a red box. The output lists various strings found in the file, including technical identifiers, company names, and password validation parameters.

```
MacBook-Pro:theHarvester ozgurcatak$ strings \n /usr/local/mysql-5.7.17-macos10.12-x86_64/lib/plugin/validate_password.so
NST3__114basic_ifstreamIcNS_11char_traitsIcEEEE
NST3__113basic_filebufIcNS_11char_traitsIcEEEE
validate
validate_password
Oracle Corporation
check password strength
LOCK_dict_file
Can't retrieve the security context
utf8
user
login user name
priv_user
effective user name
Can't retrieve the %s from the security context
/Volumes/hd2/pb2/build/sb_0-21378219-1480342151.7/mysql-5.7.17/plugin/password
Dictionary file not specified
Dictionary file not loaded
Dictionary file size exceeded
MAX_DICTONARY_FILE_LENGTH, not loaded
Exception while reading the dictionary file
%04d.%02d.%02d %02d.%02d.%02d
Effective value of validate_password_length is changed. New value is %d
validate_password_dictionary_file_last_parsed
validate_password_dictionary_file_words_count
length
Password validate length to check for minimum password length
```

Figure: Example output of *strings* command

Review The Text of a File III

DLLs/URLs

With this command, it is possible to check which DLLs are used in the executable code, whether any URL link is provided.

```
7 http://ts-ara.ws.symantec.com/sha230-155-ca.cer0/
MacBook-Pro:Presentations ozgurcatak$ strings winrar-x64-550.exe |grep dll
COMCTL32.dll
SHLWAPI.dll
USER32.dll
GDI32.dll
ADVAPI32.dll
SHELL32.dll
Fole32.dll
KERNEL32.dll
Delete=rarlng.dll
Delete=rarext.dll
Delete=rarext32.dll
Delete=rarext64.dll
7zxa.dll
RarExt.dll
```

Figure: Example output of *strings* command for DLLs

Review The Text of a File IV

```
C:>strings bp6.ex_
VP3
VW3
t$@
D$4
99.124.22.1 ④
e-@
GetLayout ①
GDI32.DLL ③
SetLayout ②
M}C
Mail system DLL is invalid.!Send Mail failed to send message. ⑤
```

Figure: Example output of *strings* command for a malware

Review The Text of a File V

```
MacBook-Pro:Presentations ozgurcatak$ strings winrar-x64-550.exe |grep http
<asmv3:windowsSettings xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings"
http://ocsp.thawte.com0
,http://crl.thawte.com/ThawteTimestampingCA.crl0
http://ts-ocsp.ws.symantec.com07
+http://ts-aia.ws.symantec.com/tss-ca-g2.cer0<
+http://ts-crl.ws.symantec.com/tss-ca-g2.crl0(
https://secure.comodo.net/CPS0C
2http://crl.comodoca.com/COMODORSACodeSigningCA.crl0t
2http://crt.comodoca.com/COMODORSACodeSigningCA.crt0$
http://ocsp.comodoca.com0
;http://crl.comodoca.com/COMODORSACertificationAuthority.crl0q
/http://crt.comodoca.com/COMODORSAAddTrustCA.crt0$
http://ocsp.comodoca.com0
https://secure.comodo.net/CPS0C
2http://crl.comodoca.com/COMODORSACodeSigningCA.crl0t
2http://crt.comodoca.com/COMODORSACodeSigningCA.crt0$
http://ocsp.comodoca.com0
;http://crl.comodoca.com/COMODORSACertificationAuthority.crl0q
/http://crt.comodoca.com/COMODORSAAddTrustCA.crt0$
```

Figure: Example output of *strings* command for URLs

Packed and Obfuscated Malware

Packing or Obfuscation

- ▶ Malware writers often use *packing* or *obfuscation* to make their files more difficult to detect or analyze.
- ▶ *Obfuscated* programs are ones whose execution the malware author has attempted to hide.
- ▶ *Packed* programs are a subset of obfuscated programs in which the malicious program is compressed and cannot be analyzed.
- ▶ *Legitimate* programs almost always include many strings.
 - ▶ Malware that is packed or obfuscated contains very few strings. If upon searching a program with Strings,
 - ▶ you find that it has only a few strings, it is probably either obfuscated or packed,

Packing Files I

Packing Files

- ▶ When the packed program is run, a small wrapper program also runs to decompress the packed file and then run the unpacked file.

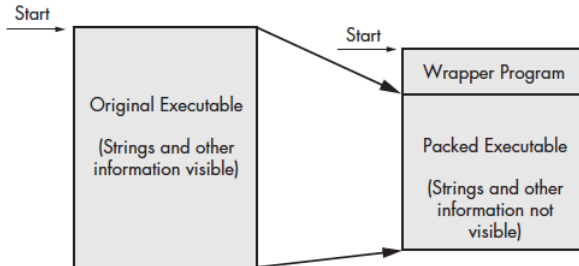


Figure: The file on the left is the original executable, with all strings, imports, and other information visible. On the right is a packed executable. All of the packed file's strings, imports, and other information are compressed and invisible to most static analysis tools.

Packing Files II

PEiD

- ▶ One way to detect packed files is with the PEiD program.
- ▶ You can use PEiD to detect the type of packer or compiler employed to build an application, which makes analyzing the packed file much easier.
- ▶ As you can see, PEiD has identified the file as being packed with UPX version 0.89.6-1.02 or 1.05-2.90
- ▶ When a program is packed, you must unpack it in order to be able to perform any analysis.

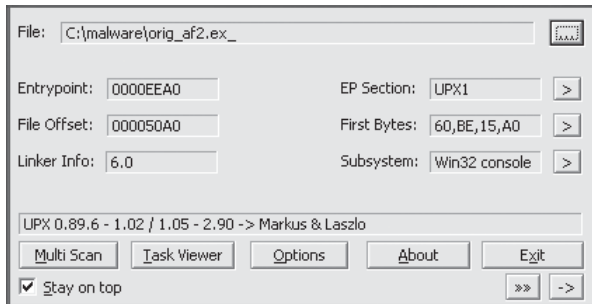


Table of Contents

- 1 Malware Analysis Methods
 - General Analysis Methods
 - Dynamic vs Static
 - Analysis Process
 - Key Points
 - Analysis Blocking Methods
 - Malware Analysis Environment Setup
- 2 Introduction
 - Basic Static Techniques
 - Antivirus
 - Hash Control
 - Review The Text of a File

- Packed and Obfuscated Malware
 - Packing Files
- 3 Windows OS
 - Portable Executable File Format
 - Tools for Windows
 - Conclusion
 - 4 Linux
 - Tools
 - REMnux
 - 5 Lab
 - Lab

Portable Executable File Format

Definition

- ▶ The Portable Executable (PE) file format is used by Windows **executables**, **object code**, and **DLLs**.
- ▶ The PE file format is a data structure that contains the information necessary for the Windows OS loader to manage the wrapped executable code.
- ▶ **File extensions:** .acm, .ax, .cpl, .dll, .drv, .efi, .exe, .mui, .ocx, .scr, .sys, .tsp

Tools for Windows

Packet Analysis

- ▶ LordPE, XPELister
- ▶ PEiD
- ▶ UPX
- ▶ OllyDump, HideOD, OllyAdvanced, OllyScript
- ▶ Quick Unpack

System Analysis

- ▶ Sys Internals
- ▶ RegShot
- ▶ MD5sum
- ▶ CaptureBAT

Code Analysis

- ▶ IDA Pro
- ▶ OllyDbg
- ▶ BinText
- ▶ XORSearch

Conclusion

Conclusion

- ▶ Using a suite of relatively simple tools,
 - ▶ we can perform static analysis on malware **to gain a certain amount of insight** into its function.
- ▶ But **static analysis** is typically only **the first step**, and further analysis is usually necessary.
- ▶ The next step is **setting up a safe environment** so you can **run the malware** and perform basic **dynamic analysis**.

Table of Contents

- 1 Malware Analysis Methods
 - General Analysis Methods
 - Dynamic vs Static
 - Analysis Process
 - Key Points
 - Analysis Blocking Methods
 - Malware Analysis Environment Setup
- 2 Introduction
 - Basic Static Techniques
 - Antivirus
 - Hash Control
 - Review The Text of a File

- Packed and Obfuscated Malware
 - Packing Files
- 3 Windows OS
 - Portable Executable File Format
 - Tools for Windows
 - Conclusion
 - 4 Linux
 - Tools
 - REMnux
 - 5 Lab
 - Lab

Linux

Tools

- ▶ For the Linux operating system, there are fewer malware than Windows.
- ▶ **Linux is mostly preferred by software developers** when considering the usage rates of personal computers.
- ▶ However, **this number is increasing** day by day with the development of subjects such as **IoT** and **mobile**, and it is also taking place within the **scope of consumer electronics**.
- ▶ The number of Linux operating system devices is increasing day by day.

REMnux I

REMnux

- ▶ A distribution called **REMnux** is used to perform malware analysis on the Linux platform.
- ▶ This distribution will be used for the analysis of malware in Linux environment.



REMnux II

Statically Examine Files

- Inspect file properties using [pescanner](#), [pestr](#), [pyew](#), [readpe](#), [pedump](#), [peframe](#), [signsrch](#), [readpe.py](#).
- Investigate binary files in-depth using [bokken](#), [vivbin](#), [udcli](#), [RATDecoders](#), [radare2](#), [yara](#), [wxHexEditor](#).
- Deobfuscate contents with [xorsearch](#), [unxor.py](#), [Balbuzard](#), [NoMoreXOR.py](#), [brxor.py](#), [xortool](#).
- Examine memory snapshots using [Rekall](#), [Volatility](#).
- Assess packed files using [densityscout](#), [bytehist](#), [packerid](#), [upx](#).
- Extract and carve file contents using [hachoir-subfile](#), [bulk_extractor](#), [scalpel](#), [foremost](#).
- Scan files for malware signatures using [clamscan](#) after refreshing signatures with [freshclam](#).
- Examine and track multiple malware samples with [mas](#), [viper](#), [malttrieve](#), [Ragpicker](#).
- Work with file hashes using [nsrlookup](#), [Automater](#), [hash_id](#), [ssdeep](#), [totalhash](#), [virustotal-search](#), [vt](#).
- Define signatures with [yaraGenerator.py](#), [autorule.py](#), [IOCextractor.py](#), [rule-editor](#).

REMnux III

Handle Network Interactions

- Analyze network traffic with [wireshark](#), [ngrep](#), [tcpick](#), [tcpxtract](#), [tcpflow](#), [tcpdump](#).
- Intercept all laboratory traffic destined for IP addresses using [accept-all-ips](#).
- Analyze web traffic with [burpsuite](#), [mitmproxy](#), [CapTiffer](#), [NetworkMiner](#).
- Implement common network services using [fakedns](#), [fakesmtp](#), [inetsim](#), "ircd start", "httpd start".

Examine Browser Malware

- Deobfuscate JavaScript with [SpiderMonkey \(js\)](#), [d8](#), [rhino-debugger](#) and [Firebug](#).
- Define JavaScript objects for SpiderMonkey using [/usr/share/remnux/objects.js](#).
- Clean up JavaScript with [js-beautify](#).
- Retrieve web pages with [wget](#) and [curl](#).
- Examine malicious Flash files with [swfdump](#), [flare](#), [RABCDAsm](#), [xxxswf.py](#), [extract_swf](#).
- Analyze Java malware using [idx_parser.py](#), [cfr](#), [jad](#), [jd-gui](#), [Javassist](#).
- Inspect malicious websites and domains using [thug](#), [Automater](#), [pdnstool.py](#), [passive.py](#).

Examine Document Files

- Analyze suspicious Microsoft Office documents with [officeparser.py](#), [oletools](#), [libolecf](#), [oledump.py](#).
- Examine PDFs using [pdfid](#), [pdfwalker](#), [pdf-parser](#), [pdfdecompress](#), [pdfxray_lite](#), [pyew](#), [peepdf](#).
- Extract JavaScript or SWFs from PDFs using [pdfextract](#), [pdfwalker](#), [pdf-parser](#), [swf_mastah](#).
- Examine shellcode using [shellcode2exe.py](#), [sctest](#), [dism-this](#), [unicode2hex-escaped](#), [m2elf](#), [dism-this.py](#).

REMnux IV

Investigate Linux Malware

- Disassemble and debug binaries using [bokken](#), [vivbin](#), [edb](#), [gdb](#), [udcli](#), [radare2](#), [objdump](#).
- Examine the system during behavioral analysis with [sysdig](#), [unhide](#), [strace](#), [ltrace](#).
- Examine memory snapshots using [Rekall](#), [Volatility](#).
- Decode Android malware using [Androwarn](#), [AndroGuard](#).

Lab I

Windows

- ▶ Lab 1: Malware hash value
 - ▶ Malware: com.parental.control.v4_1.0.apk
 - ▶ Hash algorithm: md5
 - ▶ Tool: fciv.exe
- ▶ Lab 2: Malware string search
 - ▶ Malware: candy_corn
 - ▶ Tool: Bintext
- ▶ Lab 3: Getting Information from PE head
 - ▶ Malware: defref.exe
 - ▶ CFF Explorer (or Lord PE)
- ▶ Lab 4: Packer detection with PEID
 - ▶ Malware: tnnbtib.exe
 - ▶ Pack with **upx**
 - ▶ Tool: PEid

