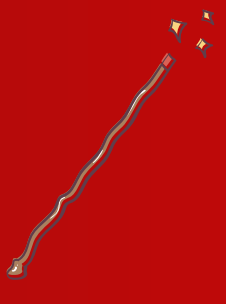
All in One Malware Analysis Cheat Sheat & Roadmap

Introduction

Two years ago, when I embarked on my journey into malware analysis, I sought to enhance my skills through various courses and resources available on the internet. However, I soon encountered numerous challenges and gaps in knowledge that weren't adequately addressed in these materials. This journey has been one of persistence and dedication, and I've come to realize that successful malware analysis hinges on having the right resources, putting in consistent effort, and dedicating time to the practice.

Throughout my two-year journey, I've connected with malware analyst experts from around the globe, including Spain, Brazil, the Philippines, Egypt, and Ireland. Their insights and experiences have been invaluable in shaping my understanding and approach to malware analysis. Inspired by the collective wisdom and the shared challenges faced by many in this field, I decided to create an all-in-one malware analysis cheat sheet roadmap.

This roadmap is designed for those who, like me, have found themselves stuck and unsure of what to learn or where to start. It aims to consolidate essential resources, techniques, and best practices in one place, making the path to proficiency in malware analysis more accessible and structured. Whether you are a beginner or someone looking to refine your skills, this cheat sheet will guide you through the key steps and tools necessary for effective malware analysis.



All in One Malware Analysis Cheat Sheat & Roadmap

Malfav

What available in this guidebook

In this guidebook, you will discover a thorough compilation of both essential and advanced resources, along with valuable suggestions to enhance your malware analysis skills. The guide covers:

1. Essential Tools for Basic Static and Dynamic Analysis:
   - A detailed overview of indispensable tools for both static and dynamic analysis, suitable for all skill levels.

1. Windows Internals:
   - In-depth information on the internal mechanisms of the Windows operating system, crucial for understanding malware behavior.

1. Windows API Functions:
   - Comprehensive details on important Windows API functions and their implications for malware analysis.

1. Networking for Malware Analysis:
   - Key networking concepts and practices essential for analyzing malware's network activities.

1. Cheat Sheet for Document Malware Analysis:
   - A practical cheat sheet for analyzing malware embedded in documents, with actionable tips and techniques.

1. Toolkit for Malicious APK Analysis:
   - A curated set of tools and methods for analyzing malicious Android applications (APKs)

.

1. Toolkit for Rootkit Detection:

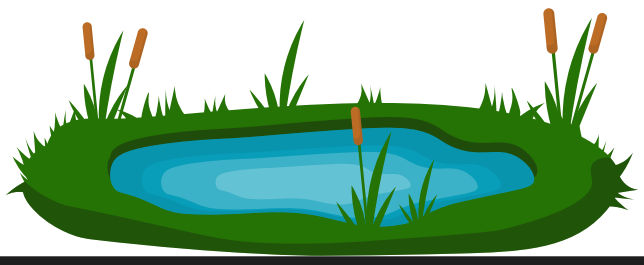
   - Resources and techniques specifically designed to identify and analyze rootkits.

1. List of Malicious API Functions:
   - A comprehensive list of API functions commonly used by malware, aiding in detection and analysis.

1. List of Suspicious Strings:
   - A compilation of strings that are often associated with malicious activity, useful for spotting potential threats.

2. Miscellaneous:

   - Additional resources and tips that don't fit into the other categories but are valuable for a holistic approach to malware analysis.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Introduction

When analyzing malware, it's crucial to focus on key aspects to ensure an effective investigation and avoid unnecessary detours. This checklist helps streamline your analysis by addressing the most important questions:

1. Determine the Malware's Purpose:
   - What is the primary goal of the malware (e.g., data theft, disruption)?
   - Who are the intended targets?

1. Identify the Infection Vector:
   - How does the malware initially infect the system (e.g., phishing, vulnerabilities)?
   - What are the delivery methods?

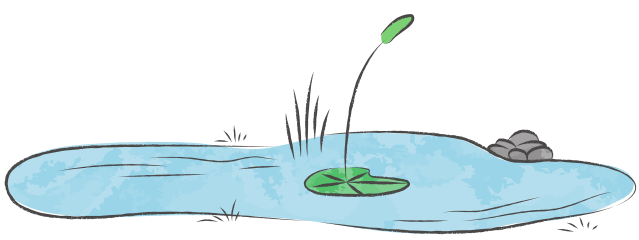
1. Analyze the Malware's Behavior:
   - What actions does the malware perform once executed (e.g., file modifications, network activity)?
   - Does it use persistence mechanisms?

1. Evaluate the Impact:
   - What potential damage does the malware cause?
   - Are there known indicators of compromise (IOCs)?
2. Investigate Command and Control (C2) Infrastructure:
   - How does the malware communicate with its C2 servers?
   - What is the purpose of this communication?

1. Examine Evasion Techniques:
   - What methods does the malware use to avoid detection (e.g., obfuscation, anti-debugging)?
   - Does it employ anti-analysis measures?

1. Identify and Document Components:
   - What are the key components of the malware (e.g., dropper files, payloads)?
   - How do these components interact?
2. Review Historical Context:
   - Is there relevant threat intelligence about this malware?
   - How does it fit into known attack patterns or threat actor profiles?

By addressing these key aspects, you can effectively focus your analysis and avoid getting sidetracked by less important details.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Flow of Analysis:

Retrieve Samples:
-Collect malware samples from various sources for analysis.

OSINT Samples and Reports:
-Gather open-source intelligence (OSINT) on the samples and review existing reports for context.

Automate Analysis:
-Utilize automated tools to perform initial scans and preliminary analysis.

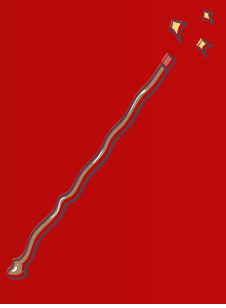
Maldoc or Fileless Analysis:
-Analyze document-based malware (maldocs) or fileless malware for specific characteristics and behaviors.

Static Analysis:
-Examine the malware without executing it to understand its structure and code.

Behavior Analysis:
-Observe and analyze the malware's behavior during execution to identify actions and impacts.

Reverse Engineering:
Deconstruct the malware to understand its inner workings and techniques used.

All in One Malware Analysis Cheat Sheat & Roadmap



Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Sample Sources:

From Client:

Directly obtained from clients or affected organizations.
Internet:

Collected from various online sources, including forums and threat intelligence websites
.
VirusTotal:

Samples available through VirusTotal's public analysis and repository.

MalwareBazaar:

A platform for sharing and obtaining malware samples.
GitHub:

Repositories where malware samples or related tools may be hosted.

Malshare:

A repository providing access to a wide range of malware samples.

Any.run:

Interactive malware analysis platform offering samples and analysis results.

Honeypot:

Samples collected from honeypots set up to attract and capture malware.

Internal Database:

Samples stored within your organization’s own malware database.

DFIR Activities:

Samples obtained through digital forensics and incident response (DFIR) activities.

All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

OSINT samples, reports, analysis:

OSINT Samples, Reports, and Analysis

Hash Lookups: Verify sample hashes.

VirusTotal: Check analysis results and reputation.

Any.Run: Interactive malware analysis.

Tri.age: Analyze malware samples.

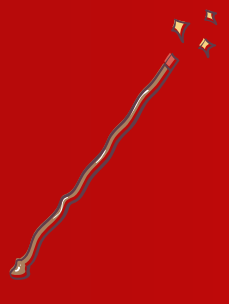
s.threatbook.com: Threat intelligence reports.

HybridAnalysis: Dynamic and static analysis results.

JoeSandbox: Advanced malware analysis.

Metadefender: Multi-engine malware scanning.

ti.qianxin.com: Threat intelligence insights.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Analysis Results and IOC Network Analysis:

AbuseIPDB: Check IP reputation.

Censys: Search for exposed devices.

Passive DNS via VT: Analyze DNS records.

Shodan: Discover internet-connected devices.

FOFA: Search for exposed services.

Validin: Validate domain and IP data.

URLhaus: Track malicious URLs.

urlscan.io: Analyze URL behavior.

AlienVault: Threat intelligence and IOCs.

threatbook.io: Threat intelligence reports

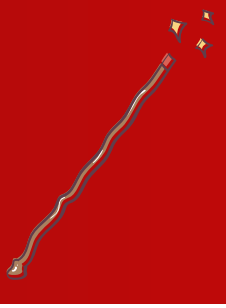
.

GitHub Search: Search for related samples.

Google Search: Find additional information.

Twitter Search: Monitor social media for relevant data.

All in One Malware Analysis Cheat Sheat & Roadmap



Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Automate and AV Analysis:

-Online Sandboxes:

Any.Run: Interactive analysis environment.

VirusTotal: Multi-engine scanning and reports.

Tri.age: Malware analysis and threat intelligence.

Threatbook: Threat analysis and reports.

AntiScan.me: Multi-engine scanning service.

FileScan: Analyze files and malware behavior.

HybridAnalysis: Detailed dynamic and static analysis.
Intezer: Code reuse detection and malware analysis.

JoeSandbox: Advanced malware analysis.

Metadefender: Multi-engine malware scanning.

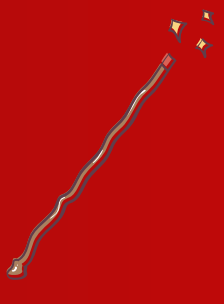
-Local Sandboxes and AV Scanners:

CAPE: Customizable malware analysis platform.

Saferwall: Community-driven malware analysis.

MultiAV: Scans files with multiple antivirus engines.

Note: Analyze all findings and understand the context of the malware before proceeding with reverse engineering.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Fileless Analysis:

- Read the Code: Examine the script or code directly.
- Beautify: Format the code for better readability.
- Deobfuscate: Remove obfuscation to reveal the true code.
- PSUnveil: Tool for analyzing obfuscated PowerShell scripts.
- Manual Analysis: Inspect the code and behavior manually.
- CMDWatcher: Monitor and analyze command-line activities.
- Refer to Behavior Analysis: Cross-check with behavioral analysis findings for context.

All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

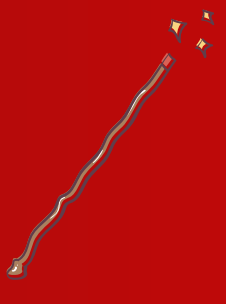
Source : https://fareedfauzi.github.io

Static Analysis:

Static Analysis involves examining the malware's code or binary without executing it. This process includes reviewing the code structure, identifying strings and metadata, and comparing against known signatures. Analyzing imports and dependencies, understanding control flow, and decompiling when needed are also crucial. Additionally, static analysis may involve examining any network packets or encoded data within the sample.

Static Analysis Flow :
Note : Static Analysis Flow added by Malfav

- Examine Code: Review the code or script without execution.
- Identify Strings: Search for readable text within the code.
- Analyze Metadata: Check file properties and headers for insights.
- Check for Known Signatures: Compare against known malware signatures.
- Review Imports/Dependencies: Look at libraries and APIs used by the malware.
- Understand Control Flow: Map out the logical structure and flow of the code.
- Decompile: Convert binary code into a readable format if necessary.
- Analyze Packets: Examine any network packets or encoded data.



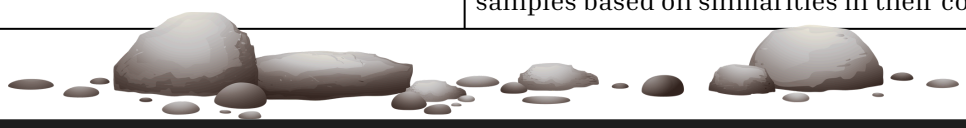
All in One Malware Analysis Cheat Sheat & Roadmap

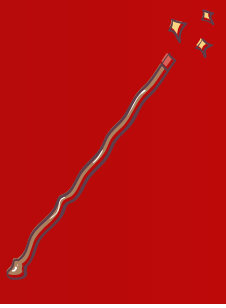
Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Static Analysis Tools :

Tool activity	Description
file	Determines the file type of a file.
TRiD	File identification tool using a database of file signatures.
Exiftool	Tool for reading, writing, and editing metadata in various file types.
DIE (Detect It Easy)	Detects and identifies packer, compiler, and other characteristics of executable files.
EXEinfoPE	Analyzes and detects various properties of PE (Portable Executable) files.
PEStudio	Analyzes PE files to identify anomalies, suspicious patterns, and potential malware indicators.
PEBear	Analyzes PE files and extracts information about their structure, sections, imports, and more.
CAPA	Analyzes malware behavior and identifies code patterns using static analysis techniques.
Floss	Extracts strings from malware samples and analyzes their behavior.
strings - -a	Extracts printable strings from binary files, including malware samples.
xorsearch	Searches for XOR-encoded strings in binary files.
base64dump	Decodes and extracts base64-encoded strings from binary files.
Resource Hacker	Views, modifies, adds, and deletes resources in Windows executables.
SSDeep	can help in classifying and categorizing malware samples based on similarities in their content





All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Behavior Analysis :

Behavior Analysis involves running the malware in a controlled environment to observe its actions and understand its functionality. This approach helps identify the real-world impact of the malware and how it operates once executed. By monitoring the malware in a sandbox or isolated virtual machine, analysts can safely study its behavior without risking infection of the host system. Key activities to observe include changes to the file system, registry modifications, network communications, and processes created or terminated. This analysis helps uncover the malware's persistence mechanisms, data exfiltration methods, and command-and-control interactions. By understanding these behaviors, analysts can develop effective detection and mitigation strategies.

Behavior Analysis Components:

Process Monitoring: Observe the creation, modification, and termination of processes, including commands executed by the malware.

Network Monitoring: Track network activity, such as connections to remote servers and data exfiltration.

File System Monitoring: Monitor changes to the file system, including creation, modification, and deletion of files.

Registry Monitoring: Watch for modifications to the system registry, including new or altered keys and values.

Logging and Detection: Record and analyze logs to detect malicious activity and gather forensic evidence.

WinAPI Monitoring: Track Windows API calls made by the malware to understand its interactions with the operating system.

All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

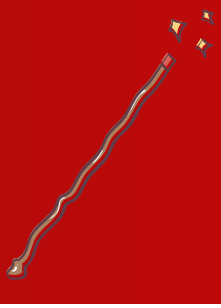
Source : https://fareedfauzi.github.io

Behavior Analysis Toolkit:

All-in-One	Process Monitoring	Network Monitoring	File System Monitoring	Registry Monitoring
ProcMon	Process Hacker	ProcessHacker	ProcMon	Regshot
SysAnalyzer	Process Explorer	TCPView	ProcDot	
	CMDWatcher	FakeNet	DirWatch	
	ProcWatch	Wireshark		
	HollowsHunter	Fiddler		
	PECapture	TCPDump		
	WriteProcessMemory			
	Moneta			

Loggin and Detection:

Logging and Detection	API Monitoring	
Sysmon	APIMonitor	
Powershell	APILogger	
Auditd		
SysmonForLinux		
Aurora		
EDR		



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Reverse Engineering:

Reverse engineering is inherently subjective, with approaches varying among analysts. When I perform reverse engineering, I start by understanding the malware sample, including its origin and initial behavior. I conduct static analysis to examine the code, strings, functions, and embedded resources without executing the malware. This involves disassembling or decompiling the code to understand its logic.

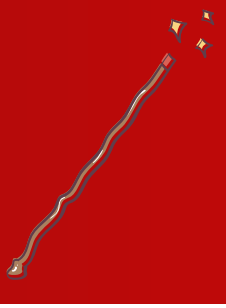
Next, I perform dynamic analysis by executing the malware in a controlled environment to observe its behavior, including interactions with the operating system, network activity, and changes to the file system or registry. I also conduct behavioral analysis to understand its objectives, such as persistence mechanisms and data exfiltration methods.

I dive deeper into the code through detailed analysis, identifying key functions and obfuscation techniques, often using a debugger. API monitoring helps track the malware's interactions with the operating system, while memory analysis uncovers hidden or obfuscated code. Logging and detection tools capture detailed logs of the malware's activity, which I correlate with known indicators of compromise. I document all findings and prepare comprehensive reports for stakeholders, ensuring a thorough understanding of the malware.

Reverse Engineering Toolkit:
Note : This section added by Malfav

- IDA: Interactive Disassembler, a powerful tool for reverse engineering binary code.
- Ghidra: A free and open-source software reverse engineering suite developed by the NSA.
- Cutter: A GUI frontend for the radare2 framework, designed for ease of use in reverse engineering.
- Binary Ninja: A reverse engineering platform with a focus on interactivity and ease of use.

All in One Malware Analysis Cheat Sheat & Roadmap

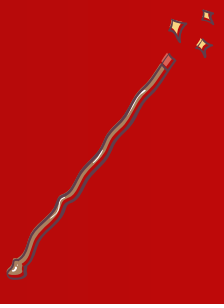


Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

What looking for in IDA:

- Find Main function: Locate the primary entry point.
- Strings reference: Identify useful strings in the code.
- Decompile: Convert binary to readable code.
- Watch Graph View: View code flow graphically.
- Relabel functions: Rename functions for clarity.
- Insert comments: Add notes to the code.
- Focus on WinAPIs: Examine Windows API calls.
- Research API documentation: Look up API details.
- Research hardcoded strings: Find embedded text.
- List functions: Enumerate function names.
- Pull Lumina: Retrieve Lumina data.
- Ask ChatGPT: Seek help with ChatGPT.
- Run Flare CAPA Explorer plugin: Use CAPA plugin for analysis.
- Run IDAClu: Execute IDAClu tool.
- Run IDA-names: Use IDA-names for renaming
- Run FindCrypt: Use FindCrypt tool.
- Run AntiVM: Check for virtual machine detection.
- Run AntiDebugSeeker: Search for debugging checks.
- Rebase segment via debugger: Adjust code base in debugger.
- Github/GitLab research: Search for code on GitHub/GitLab.
- Search unique hex values online: Look up hex values online.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

x64Dbg :

x64Dbg is an advanced, open-source debugger for Windows that supports both 32-bit and 64-bit applications. It provides a comprehensive suite of tools for analyzing and debugging software, making it particularly useful for reverse engineering and malware analysis. With x64Dbg, users can examine and manipulate program execution in detail, set breakpoints, step through code, and inspect memory and register states.

The debugger features a user-friendly interface with multiple views and panels, including disassembly, memory, and stack traces, which allow for in-depth code analysis. It supports a range of debugging techniques, such as single-stepping through instructions, examining variables, and analyzing call stacks. Additionally, x64Dbg offers powerful scripting capabilities and plugin support, enabling users to extend its functionality and automate repetitive tasks. This versatility makes x64Dbg a valuable tool for both novice and expert analysts in understanding and dissecting complex software behaviors.

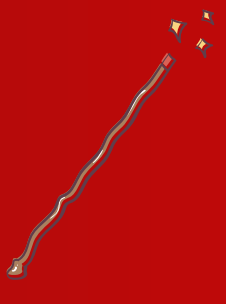
All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

What Looking for inx64Dbg :

- Breakpoint on interesting function: Set breakpoint on key functions.
- Breakpoint on unpack stuff: Break on unpacking functions like VirtualAlloc.
- Dump unpack stuff: Extract unpacked data.
- Dump shellcode stuff: Extract shellcode.
- Watch return value of function: Monitor function returns.
- Enable ScyllaHide: Use ScyllaHide for anti-debugging.
- Use graph view: Visualize code flow.
- Find strings reference: Locate string references.
- Use xAnalyzer: Analyze with xAnalyzer.
- Setting the events to Break on: Configure entry breakpoints.
- Add Exception filters: Set exception filters to 00000000-FFFFFFFF.
- Follow in dump, memory: Analyze memory dumps.
- Watch call stack: Monitor call stack.
- Watch Threads: Observe threads.
- Watch Handles: Track handles.
- Use RunDLL32 command: Run DLLs with RunDLL32.
- Supply parameter: Provide necessary parameters.
- Research WinAPI param and return value: Check API parameters and returns.
- Watch function input (param) and output (return value on EAX): Monitor function inputs and outputs
- Disable ASLR: Turn off ASLR.
- Self injection: Dump process with Process Hacker.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

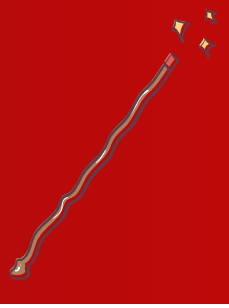
API Hooking to analysis

API hooking is a technique used in malware analysis to monitor and intercept calls made to specific functions in the Windows API. By setting breakpoints on these API functions, analysts can observe how the malware interacts with the operating system. This approach is particularly useful for understanding the behavior of malware that relies heavily on API calls for its operation.

To analyze malware using API hooking, an analyst may set breakpoints on a range of APIs that are relevant to the suspected behavior of the malware. This often involves identifying and hooking functions involved in file operations, network communications, process creation, and other critical activities. By "blindly" hooking these APIs, analysts can capture and inspect function calls regardless of the specific intent or behavior of the malware.

The process typically involves loading the malware into a debugger or hooking framework and setting breakpoints on API functions. When the malware executes and makes calls to these functions, the debugger halts execution, allowing the analyst to examine the parameters and return values. This can reveal key details about the malware's actions, such as file modifications, registry changes, or network connections.

API hooking provides a powerful way to monitor the interactions between malware and the system, offering insights into its operational logic and potential impacts. However, it requires careful selection of APIs to avoid overwhelming amounts of data and to focus on functions that are most likely to reveal critical information.



All in One Malware Analysis Cheat Sheat & Roadmap

Malfav

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

API Hooking to analysis malware

- # Typically for unpacking:
- VirtualAlloc: Memory allocation
- VirtualProtect: Memory protection changes

- # AntiDebug:
- IsDebuggerPresent: Debugger presence check

- # Enum process:
- CreateToolhelp32Snapshot: Process snapshot creation
- Process32First: First process enumeration
- Process32Next: Next process enumeration

- # Check file what file being written
- CreateFileW: File creation (wide chars)
- CreateFileA: File creation (ANSI)

- # Execute unpacked code:
- CreateProcessInternalW: Process creation
- NtWriteVirtualMemory: Memory write operations
- NtResumeThread: Thread resume
- CreateRemoteThread: Remote thread creation
- CreateThread: Thread creation

- # API Hashing:
- GetProcAddress: Procedure address retrieval
- LoadLibraryA: Library loading (ANSI)

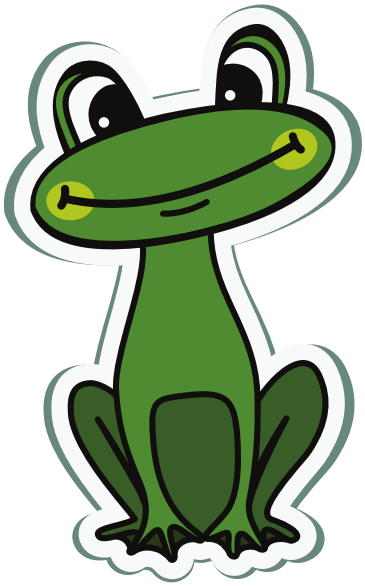
All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

API Hooking to analysis malware

Downloader:
urldownloadtofile: Download files from the internet
shellexec: Execute files or commands

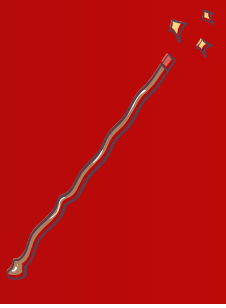


Dropper:
findresource: Locate a resource in an executable
loadresource: Load a resource into memory
lockresource: Lock a resource for access
sizeofresource: Get the size of a resource

Keylogger
getkeystate: Get the state of a key
getasynckeystate: Get the state of a key asynchronously
setwindowshook: Install a hook procedure for input events

C2 Server
internetopenurla: Open a URL for internet communication
socket: Create a network socket for communication

Note: This Section Added by Malfav



All in One Malware Analysis Cheat Sheat & Roadmap

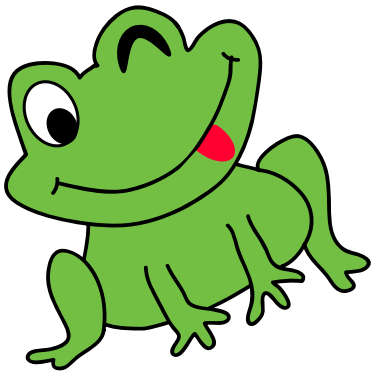
Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

API Hooking to analysis malware

Encryption:
cryptacquirecontext: Acquire a handle to a cryptographic context
cryptgenkey: Generate a cryptographic key
cryptencrypt: Encrypt data
cryptdecrypt: Decrypt data

Resource Management:
findresource: Locate a resource in an executable
loadresource: Load a resource into memory
lockresource: Lock a resource for access
sizeofresource: Get the size of a resource

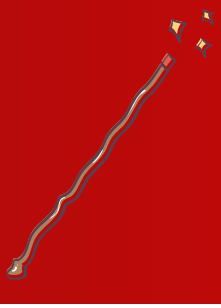


Data Transfer:
internetopen: Initialize an internet session
internetconnect: Connect to a specified server
httprequest: Send HTTP requests
send: Send data over a network connection
recv: Receive data from a network connection

Data Wiping:
deletefile: Delete a specified file
setfilepointer: Move the file pointer to a specified location
writefile: Write data to a file (can be used to overwrite file content)

Data Exfiltration:
createthread: Create a new thread (can be used to exfiltrate data in the background)
writeprocessmemory: Write data to another process's memory
readprocessmemory: Read data from another process's memory
urlmon: Download and upload data over HTTP/HTTPS

Note: This Section Added by Malfav



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

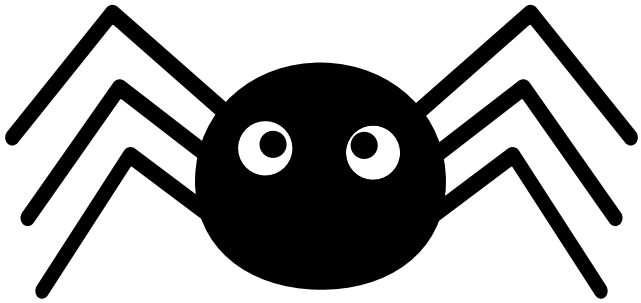
Source : https://fareedfauzi.github.io

API Hooking to analysis malware

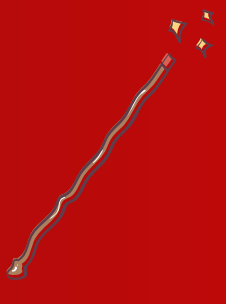
Registry Manipulation:
regcreatekeyex: Create or open a registry key
regsetvalueex: Set the value of a registry key
regqueryvalueex: Query the value of a registry key
regdeletekey: Delete a registry key
regdeletevalue: Delete a value from a registry key

File System Changes:
createfile: Create or open a file
writefile: Write data to a file
deletefile: Delete a file
movefile: Move or rename a file
copyfile: Copy a file from one location to another

These APIs cover a range of functionalities that malware might use to perform actions such as encrypting data, managing resources, transferring data over networks, wiping data, exfiltrating information, manipulating the registry, and making changes to the file system.



Note: This Section Added by Malfav



All in One Malware Analysis Cheat Sheat & Roadmap

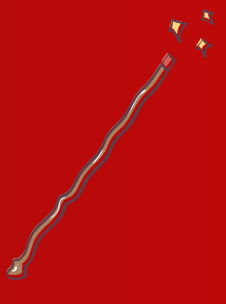
Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Common Assembly Instructions:

- MOV: Move data from one location to another.
- PUSH: Push data onto the stack.
- POP: Pop data from the stack.
- CALL: Call a procedure or function.
- RET: Return from a procedure or function.
- JMP: Jump to a specified address (unconditional jump).
- JE/JZ: Jump if equal/zero (conditional jump).
- JNE/JNZ: Jump if not equal/not zero (conditional jump).
- CMP: Compare two values.
- ADD: Add two values.
- SUB: Subtract one value from another.
- MUL: Multiply two values.
- DIV: Divide one value by another.
- AND: Perform a bitwise AND operation.
- OR: Perform a bitwise OR operation.
- XOR: Perform a bitwise XOR operation.
- NOT: Perform a bitwise NOT operation (invert bits).
- INT: Generate a software interrupt.
- NOP: No operation (does nothing).

Note: This Section Added by Malfav



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

WinAPI In Malware

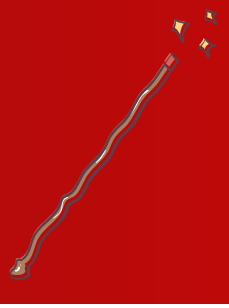
In malware analysis, it's crucial to understand that the use of Windows APIs alone does not indicate malicious behavior. To accurately assess whether API usage is part of a malicious activity, you must analyze several key aspects.

Firstly, examine the context of API usage. This involves understanding the overall purpose and behavior of the malware. Are the APIs being used in a way that aligns with known malicious activities, such as data exfiltration, system manipulation, or stealth operations? The context provides insights into why certain APIs are called and what the malware aims to achieve.

Secondly, review the parameters supplied to each API. The parameters can offer detailed information about the specific actions being performed. For instance, if an API call involves file operations, the parameters might indicate the file paths or operations being executed, such as reading, writing, or deleting files.

Finally, analyze the sequence of API calls. Malware often relies on a series of API calls to perform complex tasks. By examining the order and combination of API calls, you can discern patterns and identify how the malware progresses through its various stages, such as downloading a payload, setting up persistence, or exfiltrating data.

Together, these elements—context, parameters, and sequence—help in discerning whether the API usage is part of a malicious strategy or just routine system operations. This comprehensive approach is essential for accurate malware analysis and detection.



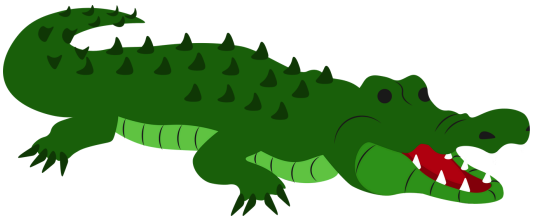
All in One Malware Analysis Cheat Sheat & Roadmap

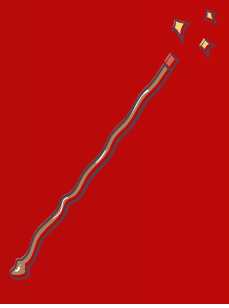
Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Common operations:

Registry operations	Mutexes	Processes and Threads	File operations	Windows Services
RegCreateKey	CreateMutex	CreateProcess	CreateFile	OpenSCManager
RegDeleteKey	OpenMutex	ExitProcess	WriteFile	CreateService
RegSetValue		CreateRemoteThread	ReadFile	OpenService
RegOpenKey		CreateThread	SetFilePointer	ChangeServiceConfig2 W
RegGetValue		GetThreadContext	DeleteFile	StartService
		SetThreadContext	CloseFile	
		TerminateProcess	MoveFile	
		CreateProcessInternalw	GetTempPath	
		ShellExecute		
		WinExed		
		ResumeThread		





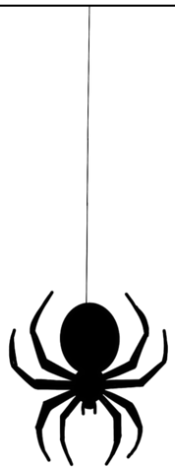
All in One Malware Analysis Cheat Sheat & Roadmap

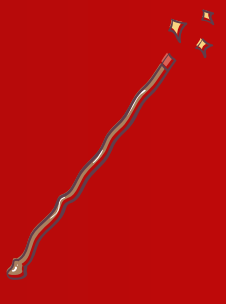
Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Common operations: Keylogging

	Description	WinAPI Function			
Installs a	hook procedure that monitors key presses.	SetWindowsHookEx			
Retrieves	the current state of the specified virtual key.	GetAsyncKeyState			
Retrieves	the status of the specified virtual key.	GetKeyState			
Retrieves	the status of all virtual keys.	GetKeyboardState			
Retrieves	a handle to the foreground window.	GetForegroundWindow			
Retrieves	the text of the specified window's title bar.	GetWindowText			
Retrieves	the text description of a key.	GetKeyNameText			
Retrieves	the active input locale identifier (formerly called	GetKeyboardLayout	Retrieves the active input locale identifier (formerly called the keyboard layout).		





All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Common operations: Backdoor connection

WinAPI Function	Description
WSAStartup	Initiates the use of the Winsock DLL by a process.
socket	Creates a socket that is bound to a specific transport service provider.
bind	Associates a local address with a socket.
listen	Places a socket in a state where it is listening for an incoming connection.
accept	Accepts a connection on a socket.
connect	Attempts to make a connection to another socket.
send	Sends data on a connected socket.
recv	Receives data from a connected socket.
read	Reads data from a file descriptor.
write	Writes data to a file descriptor.
shutdown	Disables sends or receives on a socket.
closesocket	Closes an existing socket.
WSACleanup	Terminates use of the Winsock DLL.
InternetOpen	Initializes an application's use of the WinINet functions.
InternetConnect	Initiates a connection to the specified URL.
InternetOpenUrl	Opens a URL on the internet.
InternetReadFile	Reads data from a handle opened by the InternetOpenUrl or InternetConnect function.
InternetCloseHandle	Closes a single Internet handle.
WinHttpOpen	Initializes the use of WinHTTP functions.
WinHttpConnect	Connects to an HTTP server.
WinHttpOpenRequest	Initializes an HTTP request handle.
WinHttpSendRequest	Sends the specified request to the HTTP server.
WinHttpReceiveResponse	Waits to receive the response to the HTTP request.
WinHttpQueryDataAvailable	Retrieves the amount of data available to be read by a specified request.
WinHttpReadData	Reads data from a specified request.
WinHttpCloseHandle	Closes an open handle.

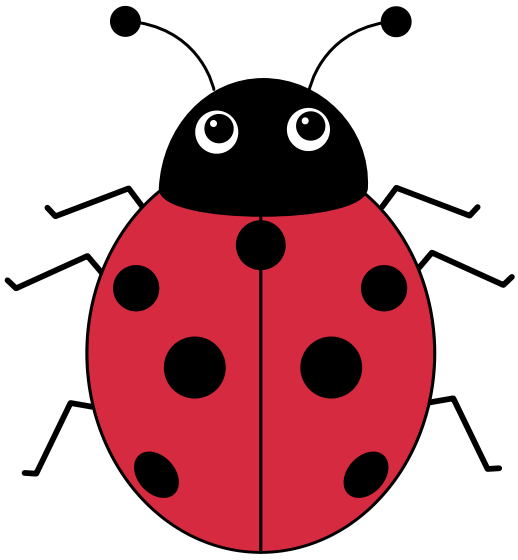
All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Common operations: Process Injection

DLL Injection	PE Injection	Reflective Injection
OpenProcess	OpenThread	CreateFileMapping
VirtualAllocEx	SuspendThread	Nt/MapViewOfFile
WriteProcessMemory	VirtualAllocEx	memcpy
CreateRemoteThread	WriteProcessMemory	Nt/MapViewOfSection
NtCreateThread	SetThreatContext	CreateThread
RtlCreateUserThread	ResumeThread	NtQueueApcThread
	NtResumeThread	CreateRemoteThread
		RtlCreateUserThread



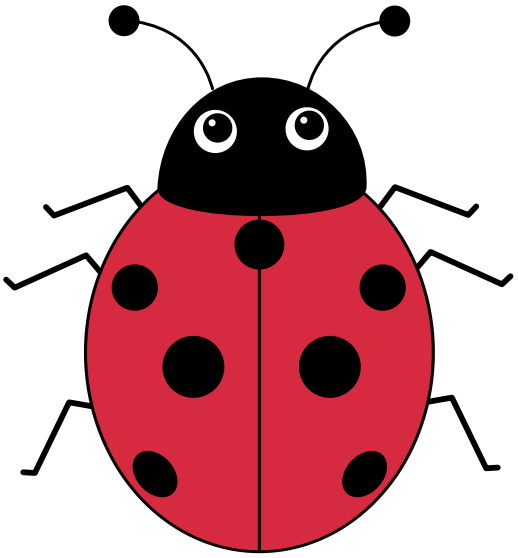
All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Common operations: Process Injection

APC Injection	Hollowing/Process Replacement	AtomBombing
SleepEx	CreateProcess	GlobalGetAtomName
SignalObjectAndWait	NtQueryProcessInformation	NtQueueApcThread
MsgWaitForMultipleObjectsEx	Zw/NtUnmapViewOfSection	GlobalAddAtom
WaitForMultipleObjectsEx	VirtualAllocEx	GlobalGetAtomName
WaitForSingleObjectEx	WriteProcessMemory	QueueUserAPC
Process32First	GetModuleHandle	
Process32Next	WriteProcessMemory	
Thread32First	GetThreadContext	
Thread32Next	ResumeThread	
QueueUserAPC		



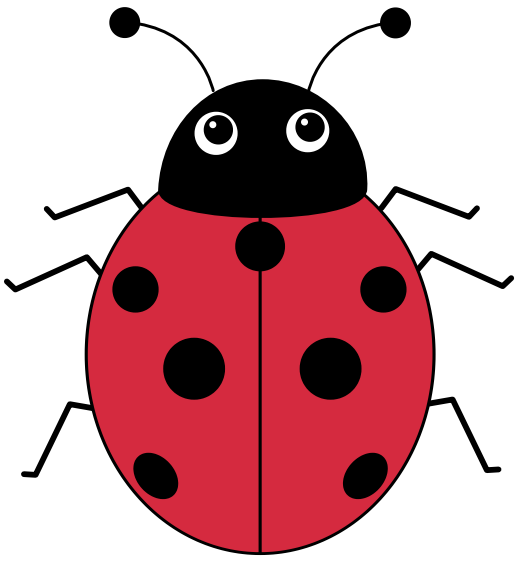
All in One Malware Analysis Cheat Sheat & Roadmap

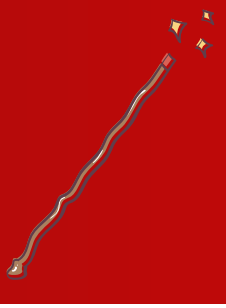
Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Common operations: Process Injection

Process Doppelgänging	Hooking Injection	Propagate Injection	Extra Windows Memory Injection
CreateTransaction	LoadLibraryW	FindWindow	FindWindowsA
CreateFileTransaction	GetProcAdress	FindWindowEx	GetWindowThreadProcessId
NtCreateSection	SetWindowsHookEx	GetProp	OpenProcess
NtCreateProcessEx	PostThreadMessage	OpenProcess	VirtualAllocEx
NtQueryInformationProcess		GetProp	WriteProcessMemory
NtCreateThreadEx		SendNotify	SetWindowLongPtrA
RollbackTransaction		VirtualAllocEx	ReadProcessMemory
		WriteProcessMemory	
		SetProp	
		PostMessage	





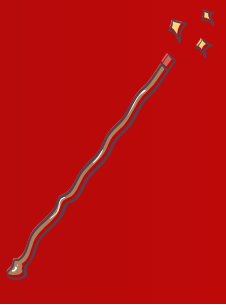
All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Common operations: Process Hooking

WinAPI Function	Description
SetWindowsHookEx	Installs an application-defined hook procedure into a hook chain.
UnhookWindowsHookEx	Removes a hook procedure installed in a hook chain by the SetWindowsHookEx function.
GetWindowLongPtr	Retrieves information about the specified window.
SetWindowLongPtr	Changes an attribute of the specified window.
SetWindowsHookEx	Installs an application-defined hook procedure into a hook chain.
CallNextHookEx	Passes the hook information to the next hook procedure in the current hook chain.



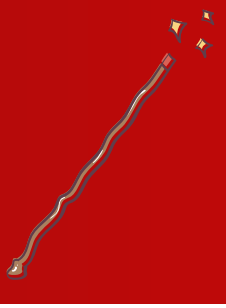
All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Common operations: Resource Related

WinAPI Function	Description
LoadResource	Retrieves a handle that can be used to obtain a pointer to the first byte of the specified resource in memory.
FindResource	Determines the location of a resource with the specified type and name in the specified module.
SizeofResource	Retrieves the size, in bytes, of the specified resource.
LockResource	Retrieves a pointer to the specified resource in memory.
EnumResourceTypes	Enumerates all resource types within a binary module.
EnumResourceNames	Enumerates all resource names of a specified type within a binary module.
EnumResourceLanguages	Enumerates all the language identifiers for the resources of a specified type within a binary module.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Common operations: Enumeration

WinAPI Function	Description
EnumProcesses	Enumerates all processes currently running on the system.
EnumProcessModules	Enumerates all modules (DLLs) loaded into a specified process.
CreateToolhelp32Snapshot	Creates a snapshot of the system, including all processes, threads, and modules.
Process32First	Retrieves information about the first process encountered in a system snapshot taken with CreateToolhelp32Snapshot.
Process32Next	Retrieves information about the next process encountered in a system snapshot taken with CreateToolhelp32Snapshot.
Module32First	Retrieves information about the first module associated with a process in a system snapshot taken with CreateToolhelp32Snapshot.
Module32Next	Retrieves information about the next module associated with a process in a system snapshot taken with CreateToolhelp32Snapshot.
EnumWindows	Enumerates all top-level windows on the screen by passing the handle to each window, in turn, to an application-defined callback function.
FindWindow	Retrieves the handle to the top-level window whose class name and window name match the specified strings.
FindWindowEx	Retrieves the handle to a window whose class name and window name match the specified strings. The function searches child windows, beginning with the one following the specified child window.
EnumDesktopWindows	Enumerates all top-level windows associated with the specified desktop.
RegEnumKey	Enumerates the subkeys of the specified open registry key.
RegEnumValue	Enumerates the values for the specified open registry key.
NetShareEnum	Retrieves information about all shared resources on a server.
NetServerEnum	Retrieves information about all servers of the specified type that are visible in a domain or workgroup.

All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Common operations: Unpacking API

Unpacking API	Description
CreateProcessInternalW	Creates a new process for unpacking the packed executable.
VirtualAlloc or VirtualAllocEx	Allocates memory for the unpacked code and data.
VirtualProtect or ZwProtectVirtualMemory	Changes the protection of a region of memory, often used for code injection.
WriteProcessMemory or NtWriteProcessMemory	Writes data to the memory of another process.
ResumeThread or NtResumeThread	Resumes the execution of a suspended thread.
CryptDecrypt or RtlDecompressBuffer	Decrypts or decompresses packed data.
NtCreateSection + MapViewOfSection or ZwMapView OfSection	Creates a section object and maps a view of a section into the address space of a process.
UnmapViewOfSection or ZwUnmapViewOfSection	Unmaps a mapped view of a section from the address space of a process.
NtWriteVirtualMemory	Writes data to the memory of a specified process.
NtReadVirtualMemory	Reads data from the memory of a specified process.
NtMapViewOfSection	Maps a view of a section of a file into the address space of a process.

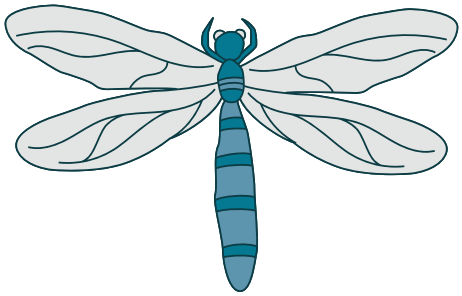
All in One Malware Analysis Cheat Sheat & Roadmap

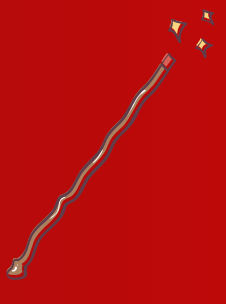
Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Common operations: Anti-Debugging

Anti-debug	Description
IsDebuggerPresent	Checks if the current process is being debugged.
CheckRemoteDebuggerPresent	Checks if a remote process is being debugged.
NtQueryInformationProcess	Retrieves information about a process, including debug flags.
OutputDebugString	Sends a string to the debugger for display.
BeingDebuggeed in PEB	Checks if the process is being debugged by inspecting the Process Environment Block (PEB).
Check ProcessHeap flag	Checks the Process Heap flags for signs of a debugger.
NtGlobalFlag	Retrieves the global debug flag for the current process.
LookupPrivilegeValue	Retrieves the locally unique identifier (LUID) for a privilege.
BlockInput	Blocks keyboard and mouse input to the system.





All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

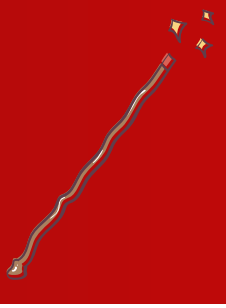
Source : https://fareedfauzi.github.io

Anti Malware Analysis:

- Signatures: Use tools like PEiD to identify known packers through signature matching.
- Strings: Lack of readable strings in the executable often suggests packing.
- Imports: Few or no import functions may indicate the file is packed.
- Sections: Unusual section names, like "UPX," can signal packing.
- Entropy: High entropy values suggest encrypted or compressed data.
- Raw/Virtual Sizes: Significant differences between raw and virtual sizes can indicate packing.

Method to Unpack :

- Statically: Involves reverse engineering the entire unpacking routine from the executable. This method can be complex and time-consuming, often proving to be less practical.
- Dynamically: Uses a debugger to monitor and control the execution of the malware. Breakpoints are set on common unpacking functions like VirtualAlloc, VirtualProtect, and others to observe and extract the unpacked payload during runtime.
- Automated: Utilizes tools and services designed to automate the unpacking process. Examples include Unpac.me, PE-sieve, MalUnpack, and specialized sandboxes that automatically handle the unpacking and analysis of malware.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

API Hashing:

API hashing is a technique used in malware analysis to identify and verify API functions by creating hash values of their names or addresses. This process involves computing a unique hash for each API function used by the malware and comparing these hashes against known values to detect obfuscated or dynamically resolved functions. By analyzing these hashes, analysts can uncover hidden or encrypted API calls, aiding in the reverse engineering and understanding of malware behavior. This method is valuable for recognizing how malware interacts with system functions, especially when these interactions are not immediately apparent due to obfuscation.

Shellcode often uses a precalculated hash to resolve APIs. The process involves:

- Iterating through all loaded modules.
- Hashing each module's name and its exported function names
- Combining these hashes and comparing them to the given hash.
- If a match is found, the function's address is resolved; otherwise, the process continues with the next module.

API Hashing	Description
LoadLibraryA	Loads a dynamic-link library (DLL) into the address space of the calling process.
GetProcAddress	Retrieves the address of an exported function or variable from a specified DLL.
LdrGetProcedureAddress	Retrieves the address of an exported function or variable using the LDT.
GetModuleHandleA	Retrieves a handle to the specified module (DLL or executable file).

All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

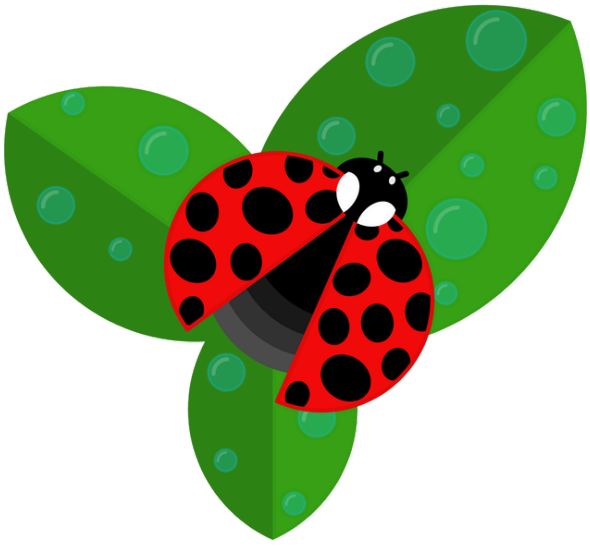
Source : https://fareedfauzi.github.io

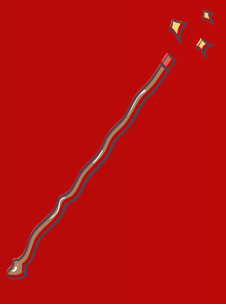
Ransomware 101:

Ransomware is a type of malicious software designed to block access to a victim's files or system until a ransom is paid. Typically, ransomware encrypts the victim's data, rendering it inaccessible without a decryption key. Once the data is encrypted, the malware displays a ransom note demanding payment, usually in cryptocurrency, in exchange for the decryption key.

The methods ransomware uses to infiltrate systems are diverse. It can spread through phishing emails, where malicious attachments or links are disguised as legitimate content, or through malicious downloads that exploit software vulnerabilities. Additionally, ransomware can propagate across networks by exploiting weak security measures or unpatched software.

The impact of a ransomware attack can be devastating, affecting individuals, businesses, and organizations by disrupting operations, causing financial losses, and potentially leading to data breaches. In response, effective prevention strategies include maintaining regular backups, applying timely security patches, and implementing comprehensive awareness training to reduce the risk of infection and mitigate potential damage.





All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

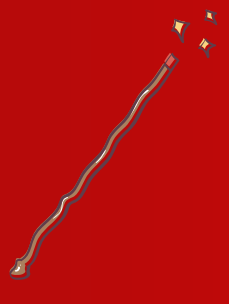
Source : https://fareedfauzi.github.io

Ransomware Flow 101:

- Collect PC Information: Gather system details.
- Target Files: Identify files and directories to encrypt using blacklists or whitelists.
- Locate Files: Find and target specified files, including network shares.
- Generate Key: Create an encryption key.
- Encrypt Files: Encrypt files, either by overwriting or creating new ones and deleting originals.
- Append Extension: Add a ransomware-specific extension to encrypted files.
- Drop Ransom Note: Place a text file with ransom instructions.

Optional Actions:

- Delete Shadow Copies: Remove backups.
- Disable Lock Files: Ensure ransom note visibility.
- Change Wallpaper: Display ransom instructions.
- Connect to C2 Server: Report the attack.
- Enumerate Network Shares: Scan for additional targets.
- Exploit Vulnerabilities: Use system weaknesses.
- Create Persistence: Maintain access.
- Stop Services: Halt critical services.
- Terminate Processes: Kill interfering processes.
-



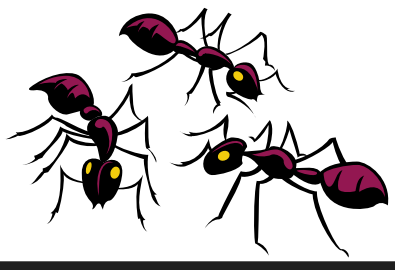
All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Ransomware : Common CryptoAPI encryption

CryptoAPI Function	Description
CryptAcquireContext	Acquires a handle to a Cryptographic Service Provider (CSP) for cryptographic operations.
CryptImportKey	Imports an embedded public key into the cryptographic context for use in encryption.
CryptGenRandom	Generates random bytes suitable for cryptographic purposes, typically used for initialization vectors (IVs).
rand	Generates pseudo-random bytes, often used for generating IVs as an alternative to CryptGenRandom.
GetTickCount	Retrieves the number of milliseconds that have elapsed since the system was started, sometimes used for generating IVs.
CryptGenKey	Generates a symmetric key for use in cryptographic operations such as encryption and decryption.
CryptSetKeyParam	Modifies various aspects of a cryptographic key, such as the key's operation mode or parameters.
CryptExportKey	Exports a cryptographic key, often used for sharing public keys generated by CryptGenKey.
CryptEncrypt	Encrypts data using the specified cryptographic key and algorithm obtained from CryptImportKey and CryptAcquireContext.
CryptDestroyKey	Destroys the cryptographic key by freeing its resources.
CryptDeriveKey	Derives a key from a specified hash value or password.
CryptDecrypt	Decrypts data using the specified cryptographic key and algorithm obtained from CryptImportKey and CryptAcquireContext.
CryptReleaseContext	Releases the handle to a cryptographic service provider (CSP) obtained from CryptAcquireContext.



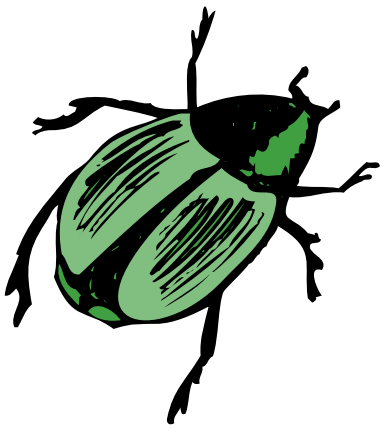
All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Ransomware : File encryption APIs

File Encryption APIs	Description
CreateFile	Opens or creates a file for reading, writing, or both.
SetFilePointer	Moves the file pointer within a file to a specified location.
SetFilePointerEx	Extended version of SetFilePointer with support for large files.
WriteFile	Writes data to a file, typically used for writing encrypted content and key information.
ReadFile	Reads data from a file, usually used for reading the original file contents.
CloseFile	Closes the file handle, releasing system resources.
MoveFile	Renames or moves a file, often used to update file extensions after encryption.



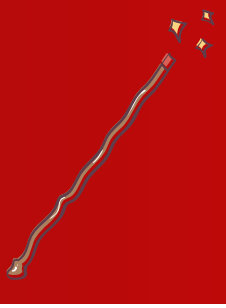
All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Ransomware : Common algorithm of data

Despite reviewing API documentation, understanding algorithm patterns, and conducting preliminary Google searches, fully deciphering malware behavior often requires more in-depth techniques. Tools such as CAPA Scanner and KANAL become invaluable in this context. CAPA Scanner excels in identifying and classifying code patterns, while KANAL specializes in recognizing and mapping out the various functions and their relationships within the malware. These tools automate the detection of common tactics, techniques, and procedures (TTPs), which can be crucial for revealing complex behaviors and hidden functionalities. They help by providing comprehensive analyses of code and identifying key indicators that manual inspection might overlook. By integrating these tools into your analysis workflow, you can achieve a more nuanced understanding of the malware, streamline the identification of its capabilities, and enhance the overall effectiveness of your investigation. This approach not only improves accuracy but also accelerates the process of developing effective countermeasures.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

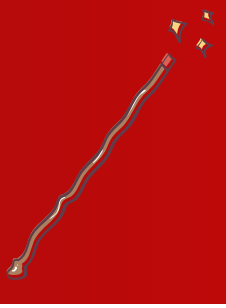
Source : https://fareedfauzi.github.io

Ransomware : Encryption

- AES (Advanced Encryption Standard): A widely used encryption algorithm known for its security and efficiency, operating on block sizes of 128, 192, or 256 bits.
- RC4 (Rivest Cipher 4): A stream cipher known for its simplicity and speed, though less commonly used today due to security vulnerabilities.
- Serpent: A block cipher designed to be highly secure, using a 128-bit block size and key sizes of 128, 192, or 256 bits, known for its strong cryptographic properties.
- Blowfish: A fast and flexible block cipher with a variable key length, offering 64-bit block size encryption, and known for its efficiency and security.

RSA

CryptoAPI Function	Description
CryptAcquireContext	Acquires a handle to a Cryptographic Service Provider (CSP) for cryptographic operations. Required to use CryptoAPI
CryptEncrypt	Encrypts data using the specified cryptographic key and algorithm obtained from CryptImportKey and CryptAcquireContext.
CryptDeriveKey	Derives a key from a specified hash value or password. Parameter Algid is crucial.
CryptDecrypt	Decrypts data using the specified cryptographic key and algorithm obtained from CryptImportKey and CryptAcquireContext.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Ransomware : Hashing

- MD5 (Message Digest Algorithm 5): Produces a 128-bit hash value, often used for data integrity checks, though it's considered weak against collision attacks.
- SHA (Secure Hash Algorithm): A family of cryptographic hash functions, including SHA-1, SHA-256, and SHA-3, providing varying levels of security and hash lengths, with SHA-256 and SHA-3 being commonly used for secure hashing.
- CRC (Cyclic Redundancy Check): A non-cryptographic hash function designed for detecting accidental changes to raw data, commonly used in error-checking scenarios.

API Function	Description
CryptAcquireContext	Acquires a handle to a Cryptographic Service Provider (CSP) for cryptographic operations. Required to use CryptoAPI
CryptCreateHash	Initiates the hashing of a stream of data. Parameter Algid is crucial.

All in One Malware Analysis Cheat Sheat & Roadmap

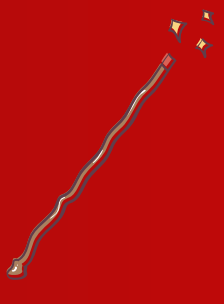
Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Ransomware : Compression

- APLib (Advanced Packer Library): A lossless compression algorithm known for its efficiency in compressing executable files, often used in software packing and protection.
- LZNT (Lempel-Ziv NT): A variant of Lempel-Ziv compression used in Windows NT-based operating systems, effective for compressing data with moderate compression ratios.
- LZMA (Lempel-Ziv-Markov chain algorithm): A high-compression, lossless algorithm used in formats like 7z and xz, known for its high compression ratios and slower compression and decompression speeds compared to other algorithms.

API Function	Description
RtlCompressBuffer	Compresses a given buffer of data
RtlDecompressBuffer	Decompresses a given buffer of compressed data



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

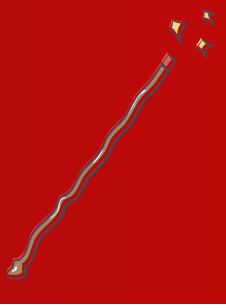
Shellcode:

Shellcode is a small piece of code used as the payload in an exploit, typically designed to execute a specific function or command within the context of a compromised system. It often operates as a series of instructions intended to be injected into a running process or memory space, allowing an attacker to execute arbitrary commands or gain control over the system. Shellcode can be written in various assembly languages and is frequently used in exploits to bypass security mechanisms, escalate privileges, or achieve persistence.

Due to its compact size and specific functionality, shellcode is often employed in buffer overflow attacks and other vulnerabilities where it needs to be both small and efficient. Its primary role is to initiate further malicious activities, such as creating backdoors, downloading additional malware, or establishing a command and control channel. Understanding shellcode is crucial for malware analysis and reverse engineering, as it reveals how attackers exploit vulnerabilities and what actions they aim to perform once they gain access.

Shellcode Flow:

- Shellcode: A sequence of bytes representing assembly instructions designed for execution within a compromised system.
- Allocation: Often uses VirtualAlloc to allocate memory for execution.
- NOP Sled: Look for the NOP (0x90) sled at the start and 00 byte values at the end, indicating its probable boundaries.
- DLL Loading: Frequently seeks kernel32.dll for functions like LoadLibrary and GetProcAddress to load DLLs and resolve API function names.
- PEB Lookup: Uses the Process Environment Block (PEB) to locate kernel32.dll in the memory space of the exploited application.



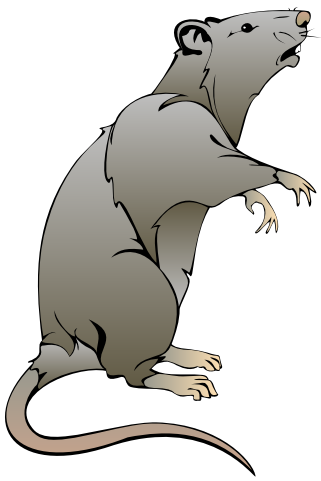
All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Shellcode: Shellcode common opcodes

Opcode	Description
FC	This translates to the instruction CLD (clear direction flag).
EB	This is the opcode for a relative jump instruction.
E8	This is the opcode for a CALL instruction.
55 8B EC	This translates to the instructions push ebp and mov ebp,esp, commonly seen at the beginning of a function (i.e., the function prologue) in x86.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

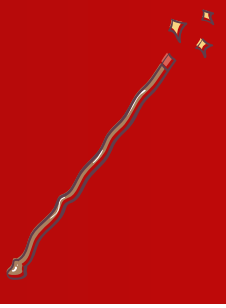
Shellcode: Tips

- To rebase shellcode addresses, use the formula: Base Address of Image - Entry Point of Shellcode.

- Tools and Methods:
- Execution: Use shellcode2exe.py to convert and analyze the shellcode. Follow up with debugging or behavior analysis using tools like jmp2it or shellcode_launcher.

- Disassembly: Utilize tools like IDA Pro (press C to convert undefined data to code), Ghidra (select the appropriate compiler for language), and x64Dbg (step through code in the debugger).

- Emulation: Employ tools like scdbg and speakeasy for emulation.
- Pattern Search: Use xorsearch with parameters -W -d 3 to identify shellcode patterns within binary files.



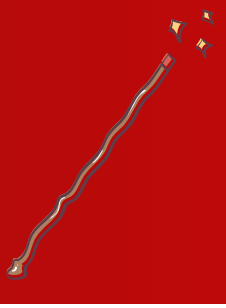
All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : malfav.gitbook.io

Toolkit for Malicious APK Analysis:

- APKTool: Useful for decompiling and reassembling APK files. It can decode resources to nearly original form and rebuild them after modification.
- JD-GUI: A Java decompiler that can be used to decompile the .dex files inside an APK to view the source code.
- Dex2Jar: Converts Android .dex (Dalvik Executable) files to Java .class files, which can then be analyzed using Java decompilers like JD-GUI.
- Androguard: A comprehensive tool for analyzing Android applications. It supports various static analysis tasks, including decompiling .dex files and analyzing the APK's structure.
- APKToolBox: A tool for analyzing and decompiling APK files, which includes various modules for different analysis needs.
- VirusTotal: Although not a static analysis tool per se, VirusTotal can be used to quickly check the reputation of APK files and see if they are flagged by antivirus engines.
- JADX: A decompiler for Android APK files that generates Java source code from .dex files.
- Static Analysis Toolkit (SAT): This tool provides a set of utilities for static analysis of Android applications, including functionalities for APK extraction and analysis.



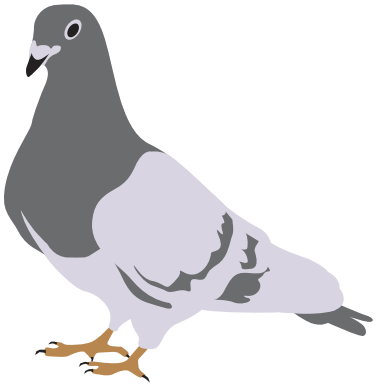
All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : malfav.gitbook.io

Toolkit for Rootkit Detection

- TDSSKiller: Detects and removes TDSS rootkits and other types of malware.
- GMER: Provides comprehensive rootkit detection and removal, including hidden processes and files.
- ASWMbR: A tool from Avast that detects and removes rootkits and bootkits.
- Sanity: Offers a range of rootkit detection features, including hidden files and registry entries.
- Rootkit Revealer: A utility from Microsoft that identifies rootkits and hidden files.
- Rootkit Buster: Detects and removes rootkits and advanced malware.
- Tuluka: Provides detection capabilities for rootkits and suspicious system behavior.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

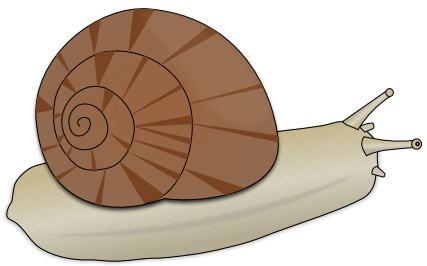
Cheat Sheet: Document Malicious Analysis:

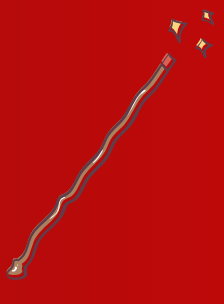
What to look for in Maldoc analysis:?

- URLs: Links used to download second-stage payloads, such as fileless commands or executables.
- Commands: Includes PowerShell, JavaScript, and wscript commands used for further actions.
- Filenames: Names and download paths of files retrieved during the attack.
- Embedded File Signatures: PE headers with MZ magic bytes indicating executable files.
- Encoded Files/Commands: Data or commands encoded to evade detection and analysis.

Interesting VBA Functions/Code:

- AutoOpen()
- AutoExec()
- AutoClose()
- Chr()
- Shell()
- Private Declare Function WINAPIFUNC Lib DLLNAME





All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://fareedfauzi.github.io

Cheat Sheet: PDF Malware Analysis

Interesting PDF keywords:

- /OpenAction: Action triggered when a PDF or document is opened.
- /AA: Adobe Acrobat-specific action.
- /Javascript: Indicates the use of JavaScript within a document.
- /JS: Abbreviation for JavaScript.
- /Names: Refers to a dictionary of names or identifiers.
- /EmbeddedFile: File embedded within a document.
- /URI: Uniform Resource Identifier, used to reference URLs.
- /SubmitForm: Action to submit form data.
- /Launch: Command to execute or open a file.
- /ASCIIHexDecode: Decoding method for ASCII hex-encoded data.
- /LZWDecode: Lempel-Ziv-Welch algorithm for decoding compressed data.
- /FlateDecode: Flate (zlib) compression/decompression.
- /ASCII85Decode: Decoding method for ASCII85-encoded data.
- /Crypt: Indicates encryption or cryptographic functions.

Common tools to analysis malicious PDF:

- PDFiD: Identifies suspicious elements in a PDF.
- pdf-parser: Analyzes and parses PDF files to reveal hidden objects and data.
- PDFtk: Manipulates and inspects PDF files.
- peepdf: Analyzes PDF files, focusing on security aspects.
- pdf stream dumper : is a tool used to extract and analyze streams of data embedded within a PDF file.

All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://malfav.gitbook.io

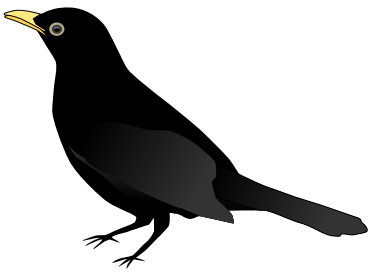
Windows Internal:

"Windows Internals" refers to the detailed workings and architecture of the Windows operating system, including its core components, subsystems, and underlying mechanisms. Understanding Windows internals is crucial for advanced troubleshooting, system optimization, and security analysis.

Why Windows Internal:?

Understanding Windows Internals is crucial for several reasons, especially for professionals involved in system administration, software development, and cybersecurity. Here's why a deep knowledge of Windows Internals is important:

- Troubleshooting: Helps diagnose and resolve complex system issues and crashes.
- Performance Optimization: Enables better resource management and system tuning.
- Security and Forensics: Assists in malware analysis, incident response, and understanding attack vectors.
- Software Development: Improves API usage and driver development.
- System Architecture: Clarifies component interactions and system design.
- Configuration and Management: Supports advanced configurations, scripting, and automation.
- Cybersecurity: Aids in vulnerability assessment and exploit development.
- System Recovery: Facilitates effective use of restore points and backups.
- Compliance: Ensures systems meet regulatory requirements.
- Continuous Learning: Keeps professionals updated with evolving technologies.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://malfav.gitbook.io

Windows Internal : Process

In Windows Internals, a process is a fundamental concept representing an instance of a running application. It includes all the necessary components required for executing a program, such as code, data, and system resources. Understanding processes in Windows involves knowing how they are managed, their lifecycle, and their interaction with other system components.

Key Concepts of Windows Processes
Process Basics:

Definition: A process is an executing instance of an application, including its code, data, and system resources. It represents a single running application or task.

Process ID (PID): A unique identifier assigned to each process by the Windows operating system. It allows the system and applications to refer to a specific process.

Process Structure

Process Control Block (PCB): A data structure used by the operating system to store information about a process. It includes details like process ID, state, priority, and resource usage.

Process Environment Block (PEB): A data structure that holds information about the process's environment, including the process's configuration, loaded modules, and initialization parameters.

Process Components

Executable Code: The actual code of the application that gets executed by the processor.

Memory Space: The virtual memory allocated for the process, including code, data, stack, and heap.

Handles: References to system resources like files, registry keys, and synchronization objects that the process uses.

All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://malfav.gitbook.io

Windows Internal : Thread

In Windows Internals, a thread is the smallest unit of execution within a process. It represents a single sequence of instructions that the operating system's scheduler can execute independently. Threads are fundamental to multitasking in modern operating systems, allowing multiple operations to be performed concurrently within a single process.

Key Concepts of Threads in Windows
Thread Basics:

Definition: A thread is an execution context within a process. It has its own stack, registers, and execution state, but shares the process's memory and resources with other threads within the same process.

Thread ID (TID): A unique identifier assigned to each thread by the operating system. It allows the system and applications to reference and manage specific threads.

Thread Components

Stack: Each thread has its own stack for storing function call information, local variables, and return addresses.

Thread Context: Includes the thread's state, such as register values, program counter, and other execution-related data.

Thread State: Represents the current status of the thread, such as running, ready, or waiting.

All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://malfav.gitbook.io

Windows Internal : Handle

In Windows Internals, a handle is an abstract reference used by the operating system to manage and interact with system resources. Handles provide a way for applications and system components to access and manipulate resources such as files, processes, threads, registry keys, and other objects without directly dealing with their underlying implementations.

Key Concepts of Handles in Windows
Definition:

Handle: A handle is a numeric identifier or pointer provided by the Windows operating system that represents an object or resource. Applications use handles to perform operations on these resources.

Types of Handles

File Handles: Used to access and manipulate files and directories.

Process Handles: Used to manage and interact with processes.

Thread Handles: Used to control and manage threads.

Registry Handles: Used to access and modify Windows Registry keys and values.

Synchronization Handles: Used for synchronization objects like mutexes, semaphores, and events.

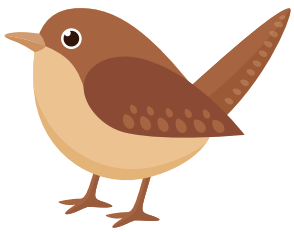
Handle Creation

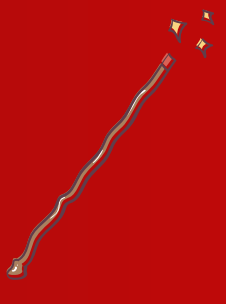
Handles are created by system functions when resources are opened or created. For example:

CreateFile creates a handle to a file or device.

OpenProcess creates a handle to an existing process.

CreateSemaphore creates a handle to a semaphore object.





All in One Malware Analysis Cheat Sheat & Roadmap

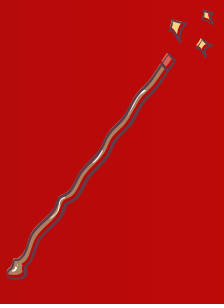
Cheatsheet: Windows Malware Analysis and Reversing

Source : https://malfav.gitbook.io

Strings : Suspicious String Lists

In malware analysis, strings are sequences of readable text embedded within binary files or memory. Analyzing these strings can reveal valuable information about the malware's behavior, capabilities, and goals. Strings often include filenames, URLs, error messages, command-line arguments, and other data that can provide insights into how the malware operates or communicates.

Suspicious Strings	Category
C:\Windows\System32; C:\Users\Public; temp; setup.exe	Filepaths and Filenames
http://; https://;www.;api/;cmd.exe	Command and Control
	(C2)
GET / POST /; socket; IP	Network Activity
; proxy	
VirtualAlloc; CreateRemoteThread; LoadLibrary; GetProcAddress;	Malicious Function Names
WriteProcessMemory	
base64; xor; decrypt; encode; crypt	Decryption and Encoding
RunOnce; Startup; Registry; ScheduledTask; AutoRun	Persistence Mechanisms
eval(; exec(; decodeURIComponent; document.write	Obfuscation



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://malfav.gitbook.io

Guide from Experts in different countries:

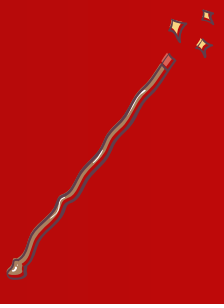
Initial Analysis and Tools:

- String and API Search: Begin by searching for strings and APIs within the malware sample.
- Injection Detection: Look for VirtualAlloc and VirtualProtect APIs to detect injection techniques.
- Process Manipulation: Identify CreateProcess for process-related actions.
- Practical Malware Analysis: Study resources like the "Practical Malware Analysis" book for comprehensive learning and exercises.

- 2. Using IDA and Sandboxing
- In IDA: Depending on your goal, search for network-related APIs to understand communications. Trace the subroutines from the API calls back to the main function to understand their role and responses.
- Sandboxing: Run the malware in a sandbox to observe its behavior and interactions with the system. This approach helps identify how it operates and what APIs it uses.

- 3. Analysis Approach
- Methodology: Depending on your goal, whether it's triage, deep analysis, or extracting C2s, tailor your approach. Use tools like Python for automation and create behavioral rules based on your findings.
- Static vs. Dynamic Analysis: Use static analysis for disassembly (e.g., IDA Pro) and dynamic analysis for observing real-time behavior in a controlled environment (e.g., using a sandbox or VM).

- 4. System Changes and Observations
- System Changes: Monitor the registry, file system, and network activity for any modifications or new entries.
- Key Areas to Check: Look at processes, services, file system changes, and registry modifications to identify persistence mechanisms.



All in One Malware Analysis Cheat Sheat & Roadmap

Cheatsheet: Windows Malware Analysis and Reversing

Source : https://malfav.gitbook.io

Guide from Experts in different countries:

- 5. Understanding Code and Behavior
- Code Analysis: In static analysis, focus on strings, functions, and APIs. Detect obfuscation or encryption routines and analyze the control flow.
- Behavioral Observation: In dynamic analysis, observe network traffic, file system changes, and new processes or services.

- 6. Advanced Techniques and Tools
- Tools for System Changes: Use tools like Procmon and Regshot to monitor changes in the registry and file system.
- API Monitoring: Tools like API Monitor can help track API calls made by the malware.
- Network Traffic: Use Wireshark to analyze network activity generated by the malware.

- 7. Overall Strategy
- Initial Steps: Gain as much information as possible without executing the malware. Check fo strings, hashes, section entropy, and APIs.
- Behavior Analysis: Observe specific behaviors, check for process spawning, and analyze execution paths.

- 8. Function Analysis and Renaming
- Function Analysis: When dealing with code, focus on understanding suspicious functions. Rename them if needed to clarify their purpose and facilitate easier analysis.

Note : This structured approach ensures a comprehensive analysis of malware, from initial discovery to in-depth understanding and behavior observation.

All in One Malware Analysis Cheat Sheat & Roadmap

"Thanks to everyone who helped make it happen. Jazak Allah Khair."