Study Guide

Intro to Malware Analysis and Reverse Engineering

Created By: Pratyay Milind, Teaching Assistant

Module 1: Introduction

Lesson 1.1: Intro Part 1

Skills Learned From This Lesson: Malware, Types of Malware, Analysis

- About the Instructor:
  - Sean Pierce
    - Certifications: CISSP
    - Twitter: @secure_sean
    - He is a Malware Analyst
    - Employer: iSIGHT Rep
- What is Malware Analysis and why is it useful?
  - Anti-Virus can't be relied on
  - 50% to 97% of Breaches involve malware
  - Breach happens – Now what?
    - Typical
      - Reimage the machine
    - Advanced: Incident Response
      - Analyze Logs, network traffic, strange processes etc.
      - Is it any where else?
      - How did it get there?
    - Mature: Gather Intelligence
      - What is the Impact?
      - What is the Risk?
      - Financially Motivated? Hacktivism? Opportunistic? Advanced Persistent Threat (APT)?
- Scope
  - Beginner's intro to:

        - Windows Malware Analysis
        - Basic Forensics / Incident Response / Malware Discovery
        - Basic Reverse Engineering
    - Recommended Background:
        - Networking – TCP / IP
        - Operating System Internals
        - Programming (C, C++)
        - Software Vulnerabilities
        - Hacking
- What is Malware
    - **Ma**licious Soft**ware**
    - Executes without permission or Knowledge
    - Software Problems like every other product:
        - Compatibility Issues
        - Bugs
        - Customer service
        - Versions / Updating Issues
        - Team Development / Source Code Control
- Malware Types / Functionality
    - General:
        - Virus (File Infector Rare)
        - Trojan (Common)
        - Worm (Rare)
        - Bot (Very Common)
        - Rootkits (Uncommon)
        - RAT (Very Common)
    - More Specialized:
        - Scareware
        - Spyware
        - Adware
        - Backdoors
        - Credential Stealers
        - Anti-Analysis
        - Defenses

        - Stealth
        - Loader / Downloader
- Other (Malicious) Software
    - Builders
    - Exploit Kit
    - Packer / Crypter
- Types of Analysis
    - Dynamic Analysis
        - Executing the Malware. Simple, Fast. Easy to miss things.
    - Static Analysis
        - Reverse Engineering. Slow, Deep technical knowledge. With enough time anything can be reversed.
    - Hybrid Static / Dynamic
        - Most Analysis is a mixture: You can find something in the disassembly then you confirm / investigate while the malware is executing.
        - Memory Forensics. Can be very useful, but is not the end-all-be-all

Lesson 1.2: Intro Part 2

Skills Learned From This Lesson: Tools, Malware, Analysis

- Basic Tools
    - SysInternals - https://docs.microsoft.com/en-us/sysinternals/
    - MAP Pack – http://sandsprite.com/CodeStuff/map_setup.exe
    - 010 – http://www.sweetscape.com/010editor/
    - PE viewer: CFF Explorer, PE Explorer, PE View, PE Studio
    - Disassembler: IDA Pro, x64_dbg, Hopper, etc.
    - Other:
        - Cygwin – md5sum, gcc, xxd, file, strings, python https://cygwin.com/install.html
        - Notepad++ - https://notepad-plus-plus.org/downloads/
        - 7zip

Lesson 1.3: Intro Part 3

Skills Learned From This Lesson: Malware Samples, Malware, Analysis

- Get Samples

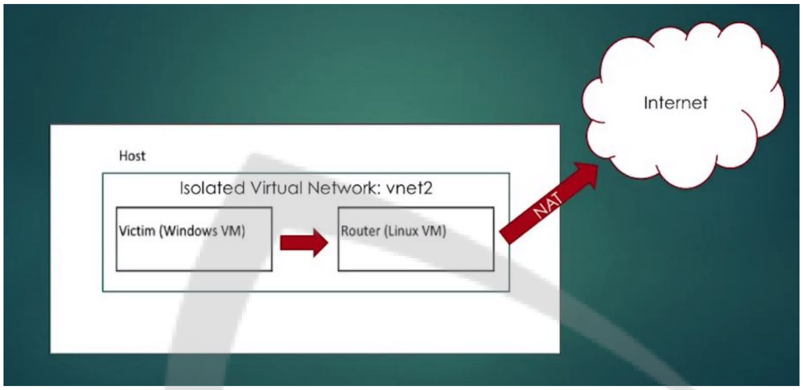
    - Contagio Malware Dump: Free; password required
        - http://contagiodump.blogspot.com/
    - KernelMode.info: Free; registration required
        - https://www.kernelmode.info/
    - Malshare: Free
        - https://malshare.com/
    - Malwares.lu's AVCaesar: Free; registration required
        - https://avcaesar.malware.lu/
    - MalwareBlacklist: Free; registration required
    - Malware DB: Free
        - https://thezoo.morirt.com/
    - Malwr: Free; regisatration required
        - https://malwr.com/
    - Open Malware: Free
    - SecuBox Labs: Free
    - VirusShare: Free
        - https://virusshare.com/
    - Catch your own: Honey Pot
    - Make your own:
        - Program Based on Description
        - Download a 'Builder'
- Note for the Paranoid:
    - Some Malware can Execute upon:
        - Being Scanned
        - Viewing Icon
            - Word
            - PDF
            - System Icon
        - Extracting the file from an Archive
    - MD5 vs. SHA256

Module 2: Lab Setup

Lesson 2.1: Lab Setup Part 1

Skills Learned From This Lesson: Lab, Analysis, Malware

- Industry Standard Setup

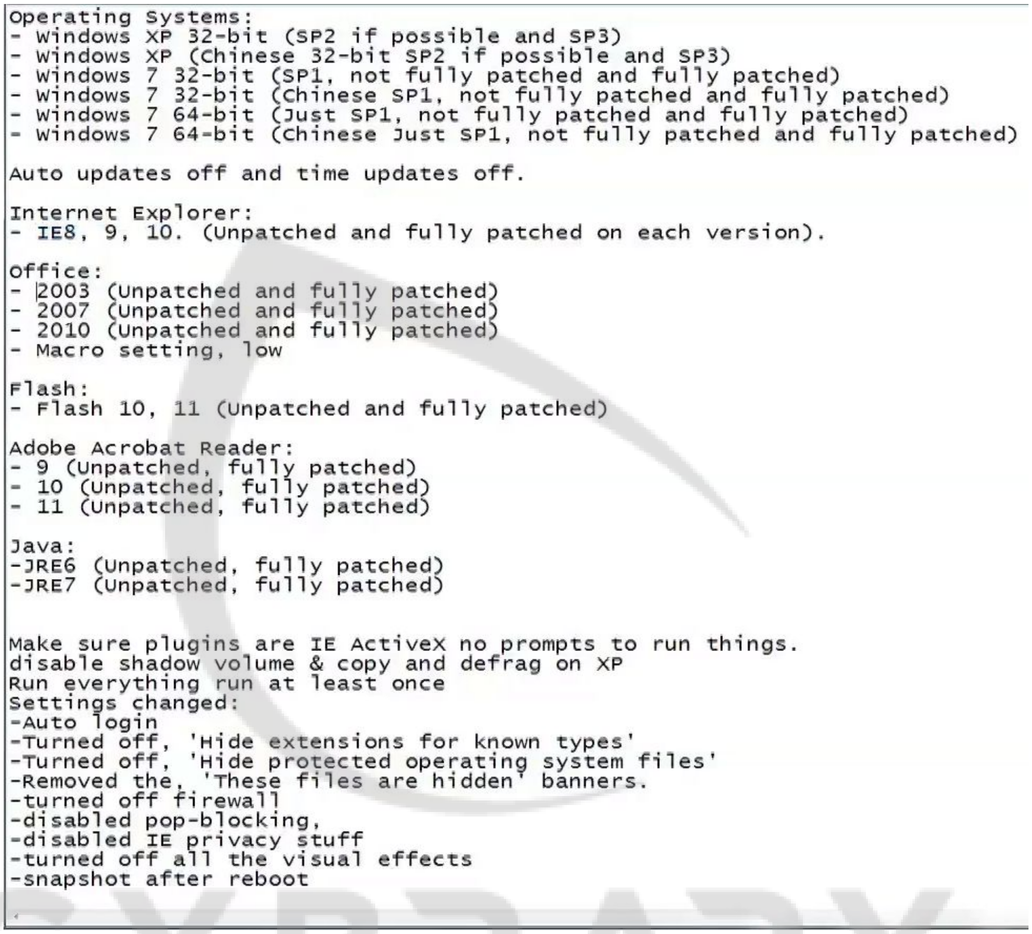


  - Basic Setup
    - Install VMWare
    - Install Windows XP
      - Install VMware tools
      - Install Analysis tools
    - Setup Kali
      - Install VMWare tools
      - Setup Network

Lesson 2.2: Lab Setup Part 2

Skills Learned From This Lesson: Downloading, Malware, Setup

- Steps:
  - Download and Install VMWare WorkStation
    - https://www.vmware.com/
  - VM Notes provided by the instructor



```
Operating Systems:
- Windows XP 32-bit (SP2 if possible and SP3)
- Windows XP (Chinese 32-bit SP2 if possible and SP3)
- Windows 7 32-bit (SP1, not fully patched and fully patched)
- Windows 7 32-bit (Chinese SP1, not fully patched and fully patched)
- Windows 7 64-bit (Just SP1, not fully patched and fully patched)
- Windows 7 64-bit (Chinese Just SP1, not fully patched and fully patched)

Auto updates off and time updates off.

Internet Explorer:
- IE8, 9, 10. (Unpatched and fully patched on each version).

Office:
- 2003 (Unpatched and fully patched)
- 2007 (Unpatched and fully patched)
- 2010 (Unpatched and fully patched)
- Macro setting, low

Flash:
- Flash 10, 11 (Unpatched and fully patched)

Adobe Acrobat Reader:
- 9 (Unpatched, fully patched)
- 10 (Unpatched, fully patched)
- 11 (Unpatched, fully patched)

Java:
-JRE6 (Unpatched, fully patched)
-JRE7 (Unpatched, fully patched)


Make sure plugins are IE ActiveX no prompts to run things.
disable shadow volume & copy and defrag on XP
Run everything run at least once
Settings changed:
-Auto login
-Turned off, 'Hide extensions for known types'
-Turned off, 'Hide protected operating system files'
-Removed the, 'These files are hidden' banners.
-turned off firewall
-disabled pop-blocking,
-disabled IE privacy stuff
-turned off all the visual effects
-snapshot after reboot
```

Lesson 2.3: Lab Setup Part 3

Skills Learned From This Lesson: Dynamic, Analysis, Tools,

- Dynamic Analysis Tools for Virtual Machine
    - For Dynamic Analysis
        - Capture BAT
        - RegShot
        - PEid
        - LordPE
        - Import Reconstructor

        - OllyDbg 2.0
- Levels of Automating / Outsourcing
    - Local VM
    - Scripting Local VM
    - Automating ESX (i), Zen, Hyper-V
    - Cuckoo Sandbox
    - Malware Farm
    - Virus Total, Anubis
    - Joe Sandbox, Hybrid Analysis, ThreatGrid
    - FireEye
- Notes for the Paranoid
    - Vulnerabilities in VMWare
    - Some malware will detect it's in a VM and act differently
        - VMWare tools
        - Easy: MAC address, timings. Advanced: v-instructions. Very Advanced: bluepill
    - Some malware will check / rely on correct DNS resolutions
    - Some malware will do an external IP check
    - Checks name for 'malware', or 'sample' or username of 'user'
    - You can route the connections through a logless VPN

Module 3: Dynamic Analysis Part 1

Lesson 3.1: Dynamic Analysis Part 1.1

Skills Learned From This Lesson: Dynamic, Analysis, Malware

- What is Dynamic Malware Analysis
    - Execute the Malware
    - First Response / Triage
    - Virtual Machine vs. Native Hardware
    - Characteristics:
        - Easy
        - Fast
        - Code may not execute
    - Goals:
        - Generate Indicators of Compromise (IoC's)

        - Determine Malware Type / Family
        - Assess Risk and Impact
        - Attribution

Lesson 3.2: Dynamic Analysis Part 1.2

Skills Learned From This Lesson: Snapshot, Dynamic Analysis, Malware Samples

- Get Samples
    - theZoo aka Malware DB: https://github.com/ytisf/theZoo
        - Dyre: https://github.com/ytisf/theZoo/blob/master/malwares/Binaries/Dyre/Dyre.zip

Lesson 3.3: Dynamic Analysis Part 1.3

Skills Learned From This Lesson: Dynamic, Analysis, Malware Samples

- Demo
    - Download Malware from https://malshare.com/
        - Snapshot
            - CaptureBAT
            - RegShot
            - Autoruns
        - More Advanced:
            - SysAnalyzer
            - ProcMon
            - OllyDbg
- Note for the Paranoid:
    - Some Malware will detect:
        - Executing / Installed Analysis Tools
        - Virtual Machine Containment
        - Sandbox Containment
        - Security Products
    - Other Reasons why it might not run correctly:
        - Incorrect environment:
            - Software Versions

- Installed OS Language
- Multiple components
- Disabled networking

- Bugs in the Malware
- Dependencies not met
- ‘Kill dates’
- Specific Target

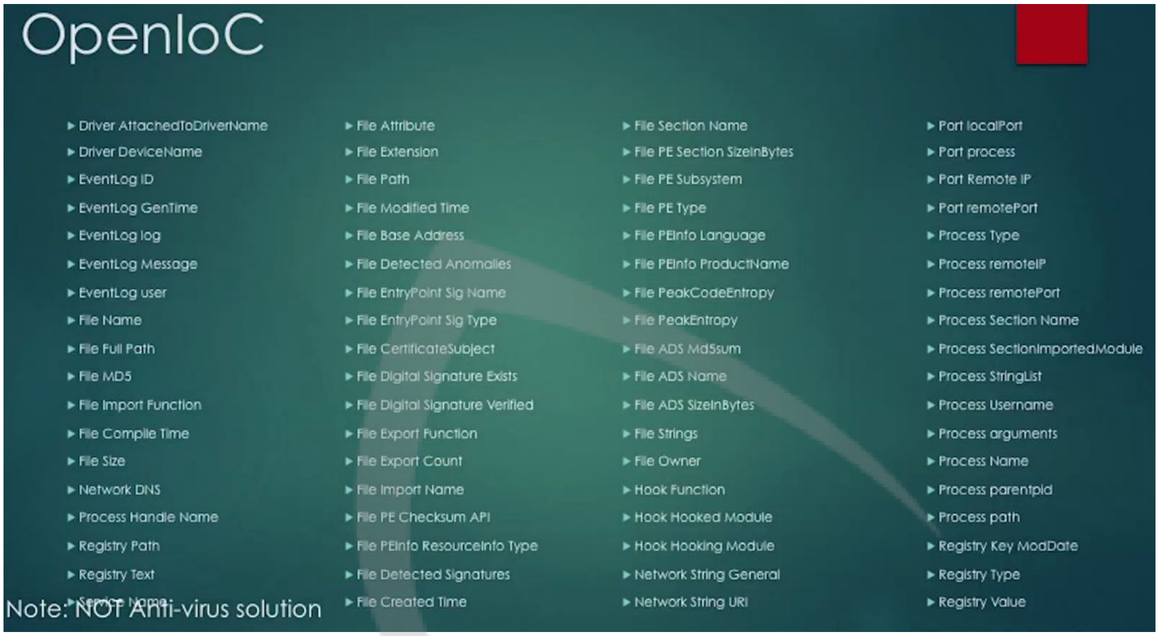
- Note for the Paranoid
    - Malware Repo should be non-execute:
        - Windows Host:
            - icacls C:\malware /deny “Everyone: (OI)(IO)(X)”
        - Linux Host:
            - chmod 600 /malware <file_name>
    - User interaction

Module 4: Dynamic Analysis Part 2

<u>Lesson 4.1</u>: Dynamic Analysis Part 2.1

Skills Learned From This Lesson: Dynamic Analysis, Malware, Indicators of Compromise

- Dynamic Malware Analysis
        - Indicators of Compromise (IoC’s)
            - File Hashes
            - Strings
            - Registry Keys
            - File Names
            - File Paths
            - Process Names
            - IP Addresses
            - Domains
            - URLs
            - Network Traffic
- OpenIoC



OpenIoC

- Driver AttachedToDriverName
- Driver DeviceName
- EventLog ID
- EventLog GenTime
- EventLog log
- EventLog Message
- EventLog user
- File Name
- File Full Path
- File MD5
- File Import Function
- File Compile Time
- File Size
- Network DNS
- Process Handle Name
- Registry Path
- Registry Text
- Service Name
- File Attribute
- File Extension
- File Path
- File Modified Time
- File Base Address
- File Detected Anomalies
- File EntryPoint Sig Name
- File EntryPoint Sig Type
- File CertificateSubject
- File Digital Signature Exists
- File Digital Signature Verified
- File Export Function
- File Export Count
- File Import Name
- File PE Checksum API
- File PEInfo ResourceInfo Type
- File Detected Signatures
- File Created Time
- File Section Name
- File PE Section SizeInBytes
- File PE Subsystem
- File PE Type
- File PEInfo Language
- File PEInfo ProductName
- File PeakCodeEntropy
- File PeakEntropy
- File ADS Md5sum
- File ADS Name
- File ADS SizeInBytes
- File Strings
- File Owner
- Hook Function
- Hook Hooked Module
- Hook Hooking Module
- Network String General
- Network String URI
- Port localPort
- Port process
- Port Remote IP
- Port remotePort
- Process Type
- Process remoteIP
- Process remotePort
- Process Section Name
- Process SectionImportedModule
- Process StringList
- Process Username
- Process arguments
- Process Name
- Process parentpid
- Process path
- Registry Key ModDate
- Registry Type
- Registry Value

Note: NOT Anti-virus solution

- Good Resources:
  - https://github.com/rshipp/awesome-malware-analysis
  - https://www.malware-analyzer.com/
  - http://opensecuritytraining.info/MalwareDynamicAnalysis.html

Lesson 4.2: Dynamic Analysis Part 2.2

Skills Learned From This Lesson: Dynamic Analysis, Demo, Analyzing IllusionBot

- Demo
  - Download:
    - https://github.com/ytisf/theZoo/blob/master/malwares/Binaries/IllusionBot_May2007/IllusionBot_May2007.zip
  - Network Traffic
    - Wireshark
    - Strings -> YARA sigs

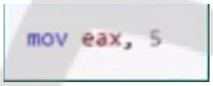
Module 5: Basic Static Analysis

Lesson 5.1: Basic Static Analysis Part 1

Skills Learned From This Lesson: Static Analysis, Malware, Assembly Code

- What is Static Analysis?
    - Reading the assembly code
    - Use tools such as
        - Debuggers
        - Disassemblers
    - Characteristics:
        - Slow
        - Detail oriented
        - Technical Knowledge Required
    - Goals:
        - Confirm Dynamic Analysis
        - Understand Behavior
        - Find more Indicators of Compromise
            - Encrypted Strings / Payloads
            - Domain Generation Algorithms (DGA's)
            - Network Traffic Encryption Algorithms
        - Determines Defenses
            - Anti-Debugging
            - Anti-VM
        - Determine Capabilities for Assess Risk and Impact
        - Determine Sophistication
        - Attribution
- What is Assembly?
    - Human readable machine code for a particular chip
        - Intel invented the 8086 chips in 1978
            - Used in the IBM PC
            - Originally 16-bit
        - Focus on x86 code (aka 'i486' architecture or '32-bit')
        - Examples of other Architectures:
            - AMD x64 – common in PC's also known as 'x64' or '64-bit' most x64 chips also have the circuitry to execute x86 code

- ARM – common in phones and tablets
- MIPS – common in printers

- More Details about x86 Assembly
    - 14 Instructions make up 90% of cache
    - Syntax
        - Intel
            - mov eax, 5
            - 

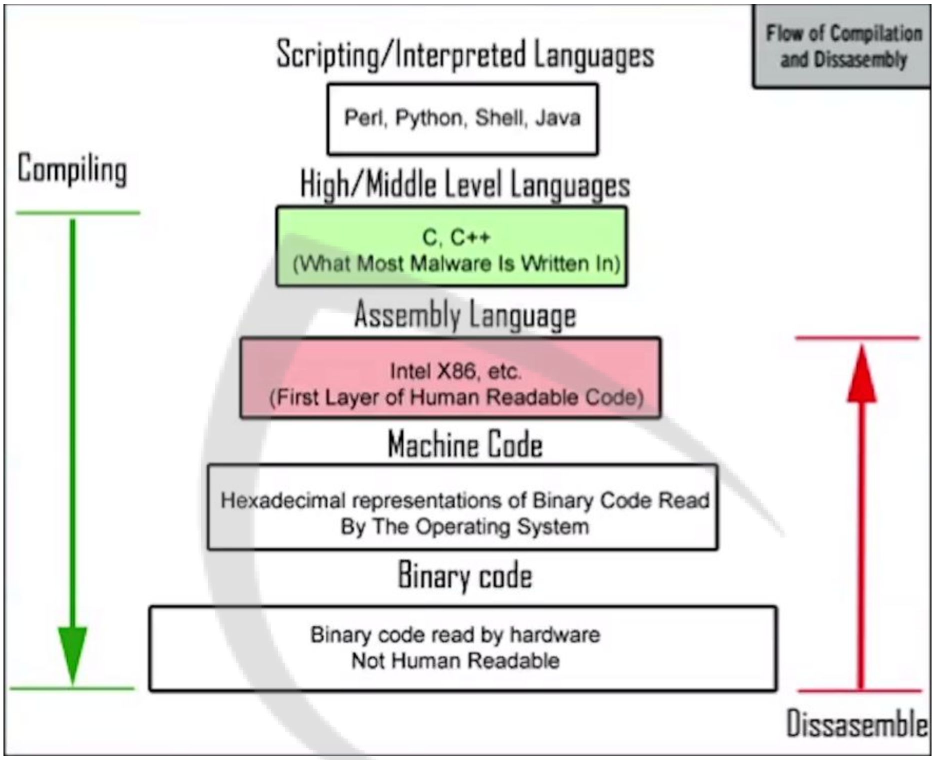
              ```
              mov eax, 5
              ```
        - AT&T
            - mov $5, %eax
            -

              ```
              mov $5, %eax
              ```
    - Programming Knowledge is needed
        - Functions
        - Local Variables
        - Application Programming Interfaces (API's)
    - Math
        - Binary
        - Hex
        - Decimal

1. MOV
2. PUSH
3. CALL
4. CMP
5. ADD
6. POP
7. LEA
8. TEST
9. JE
10. JMP
11. RET

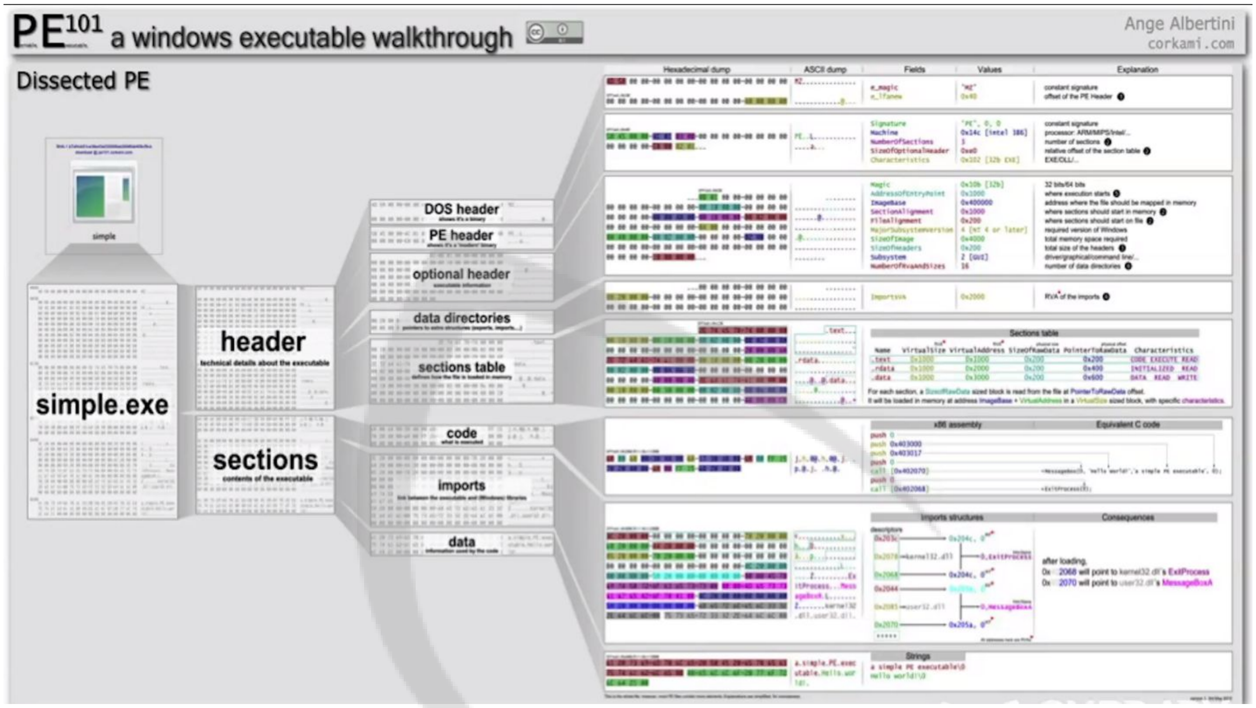
12. INC

- Demo: Compiling ‘C’ Code
  - C is a lower level language
  - Demo:
    - gcc -S hello.c
    - cl /FA hello.c
    - Visual Studio
      - Project Settings -> C / C++ -> Output Files -> ASM List Location
      - Change “Assembly Output” to “Assembly With Source Code”
    - Place a break point in the debugger right click and find “Go to Assembly”
    - OllyDbg


Flow of Compilation and Dissasembly
Scripting/Interpreted Languages
Perl, Python, Shell, Java
Compiling
High/Middle Level Languages
C, C++
(What Most Malware Is Written In)
Assembly Language
Intel X86, etc.
(First Layer of Human Readable Code)
Machine Code
Hexadecimal representations of Binary Code Read By The Operating System
Binary code
Binary code read by hardware
Not Human Readable
Dissasemble

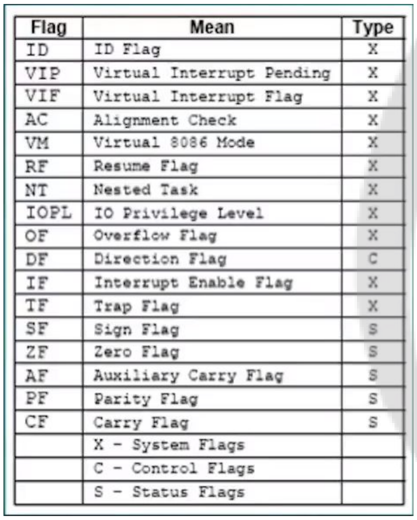
Lesson 5.2: Basic Static Analysis Part 2

Skills Learned From This Lesson: Static Analysis, PE, Malware, Assembly Code

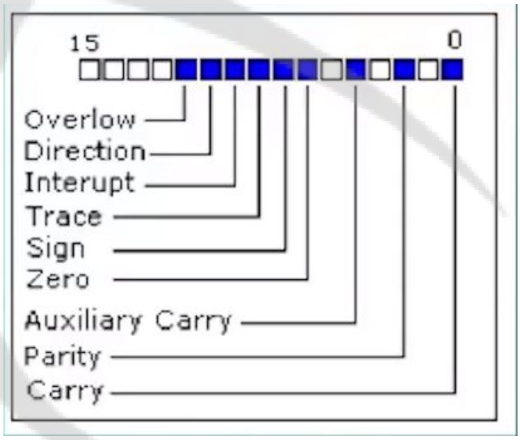


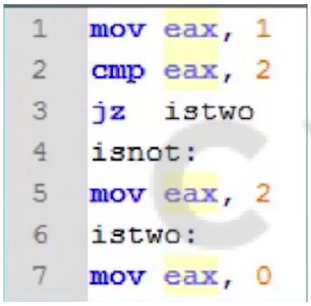
- PE file Parsers
    - PE Explorer
    - COFF Explorer
    - PEiD
    - PE Studio
    - 010 Hexeditor with the PE Binary Templates
    - Make your own:
        - Malware Analysis Cookbook
- Portable Executables
    - Most modern Windows executables use the ‘PE’ format
        - .exe
        - .dll
        - .src
        - .cpl
        - .ocx

  - .sys
  - .drv
  - .efi
  - .fon
- EFLAGs Register



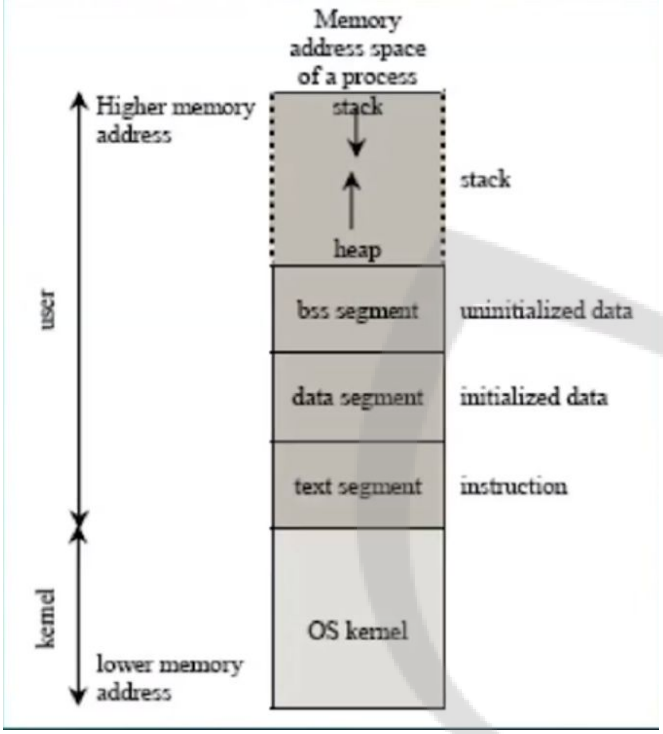
Flag	Mean	Type
ID	ID Flag	X
VIP	Virtual Interrupt Pending	X
VIF	Virtual Interrupt Flag	X
AC	Alignment Check	X
VM	Virtual 8086 Mode	X
RF	Resume Flag	X
NT	Nested Task	X
IOPL	IO Privilege Level	X
OF	Overflow Flag	X
DF	Direction Flag	C
IF	Interrupt Enable Flag	X
TF	Trap Flag	X
SF	Sign Flag	S
ZF	Zero Flag	S
AF	Auxiliary Carry Flag	S
PF	Parity Flag	S
CF	Carry Flag	S
	X - System Flags	
	C - Control Flags	
	S - Status Flags	

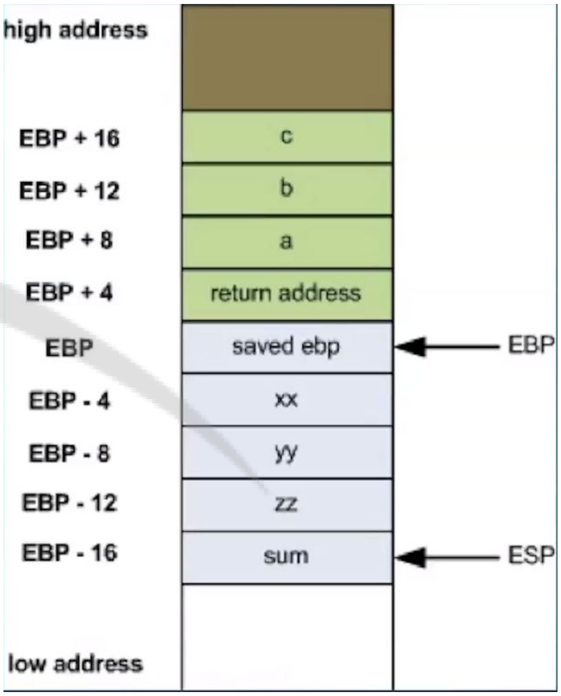




```
mov eax, 1
cmp eax, 2
jz  istwo
isnot:
mov eax, 2
istwo:
mov eax, 0
```

- The Stack

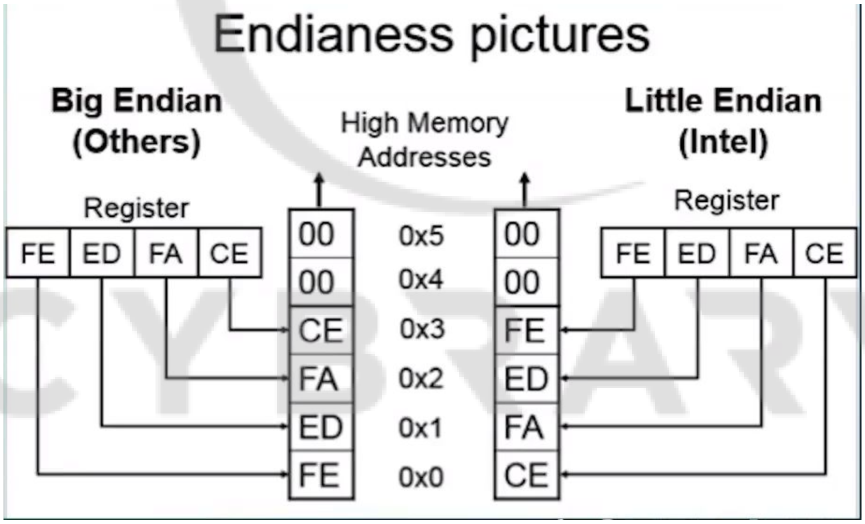




- At the top of the memory
- Grows downward
- Normally holds local variables
- ESP Points the top of the stack (The lowest memory address)
- EBP – Extended Base Pointer, always points to the bottom of the stack (The highest memory address)
- PUSH Instruction – DECrements ESP (stack pointer) by 4, and MOV'es 4 bytes at that location.
- POP – MOV'es the ESP value and increments the stack by 4.
- CALL – PUSH'es EIP, and JMP's to the function address.
- RET – JMP's to the return address which was pushed on to the stack during the CALL instruction just before the

- Misc.

    - NOP Instruction
    - Flags
        - Example: Zero Flag
    - Bit masks:
        - Example:
            - 0010 AND 1110 = 0010
            - 0x0000FF00 AND 0xA0AB2AA01 = 0X0000AA00
    - Endianness
        - Big Endian in Intel Registers. Little Endian in storage
    - Size of datatypes such as WORD, DWORD, QWORD
    - One’s Complement – flip all bits
    - Two’s Complement – flip all bits + 1
    - Negative numbers are the Two’s Complement of the positive number
- Endian



    - Little Endian – 0x12345678 stored in RAM “little end” first. The least significant byte of a word or larger is stored in the lowest address.
        - E.g. 0x78563412
            - Intel is Little Endian

- Big Endian – 0x12345678 stored as is
    - Network traffic is Big Endian
    - Most everyone else you've heard of (PowerPC, ARM, SPARC, MIPS) is either Big Endian by default or can be configured as either (Bi-Endian)
- Visual Representation

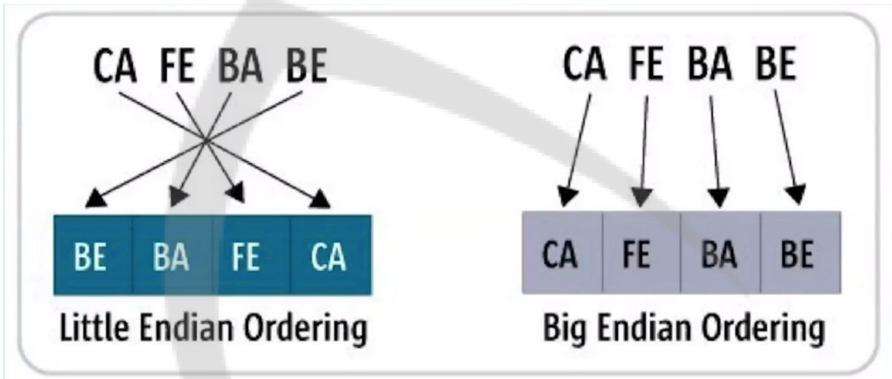


Figure 2 Byte ordering

- Little Endian Example
    - 11AB44FFAADD1221
    - 0x11AB44FF 0xAADD1221
    - 0xFF44AB11 0x2112DDAA
    - 0xFF44AB112112DDAA
- Notes for the Paranoid
    - Disassemblers can be wrong
        - Without running the code it's impossible to know what instructions will actually be executed
        - Malware will use code that tricks / breaks disassemblers / debuggers such as switching from x86 to x64 code. And JMP'ing into the middle of other instructions.
        - Malware will sometimes modify its own code while executing
    - Some malware will statically compile library's in to itself. This will make the malware much larger and difficult to analyze. IDA Pro automatically tries to identify statically compiled libraries.

  - Malware will have ‘junk code’ which does nothing or as no functional impact
  - Malware could not follow conventions. Such as using MOV’s and SUB’s instead of PUSH for API parameters
  - Malware sometimes corrupts its own stack to mess up disassemblers
- Recap & List of Good Resources
  - Goals of Static Analysis
  - Assembly
  - The IDA Pro Book: The Unofficial Guide
    - Chris Edge
  - Professional Assembly Language
    - Richard Blum
  - Reversing: Secrets of Reverse Engineering
    - Eldad Eilam
  - Corkami
    - https://github.com/corkami
  - http://opensecuritytraining.info/IntroX86.html
    - https://www.youtube.com/playlist?list=PL038BE01D3BAEFDB0
  - https://en.wikipedia.org/wiki/X86_assembly_language
  - https://en.wikipedia.org/wiki/X86_calling_conventions

Lesson 5.3: Basic Static Analysis Part 3

Skills Learned From This Lesson: Static Analysis, PE, Malware, Assembly Code

- Calling Conventions
  - cdecl
    - “C declaration”
    - Most common
    - Push reverse order parameters
    - Caller is responsible for cleaning up the stack
  - stdcall
    - Microsoft API
    - Push reverse order parameters
    - Callee is responsible for cleaning up the stack
- Demo

- stdcall vs. cdecl
- Different ways to put data on the stack
    - gcc
    - Visual studio (36:47)

Lesson 5.5: Basic Static Analysis Part 4B

Skills Learned From This Lesson: Static Analysis, Tricks, Malware

- Tricks
    - Why are we doing this?
        - Understand the Malware
        - Discover Indicators of Compromise
        - Confirm Dynamic analysis
        - Discover Anti-Debugging code
    - Tricks used by Malware so far:
        - Stack Corruption
        - Import Hiding
            - Dynamic Function Resolving
        - String Obfuscation

Lesson 5.6: Basic Static Analysis Part 5

Skills Learned From This Lesson: Static Analysis, Basics, Tips

- Tips and Tricks
    - Dealing with Obfuscated Strings
        - The Hard way: Fully reverse engineer the code, re-implement it, then apply the same process to the strings.
        - The Easy way: Use the native code.

Lesson 5.7: Basic Static Analysis Part 6

Skills Learned From This Lesson: Static Analysis, Basics, Tips

- Enumerating Capabilities
    - Find Command Processing Subroutines
    - Configuration Processing Subroutines
    - Document everything!
    - Test afterwards via Dynamic Analysis

  - *Note: There is no 'Undo' in IDA!*

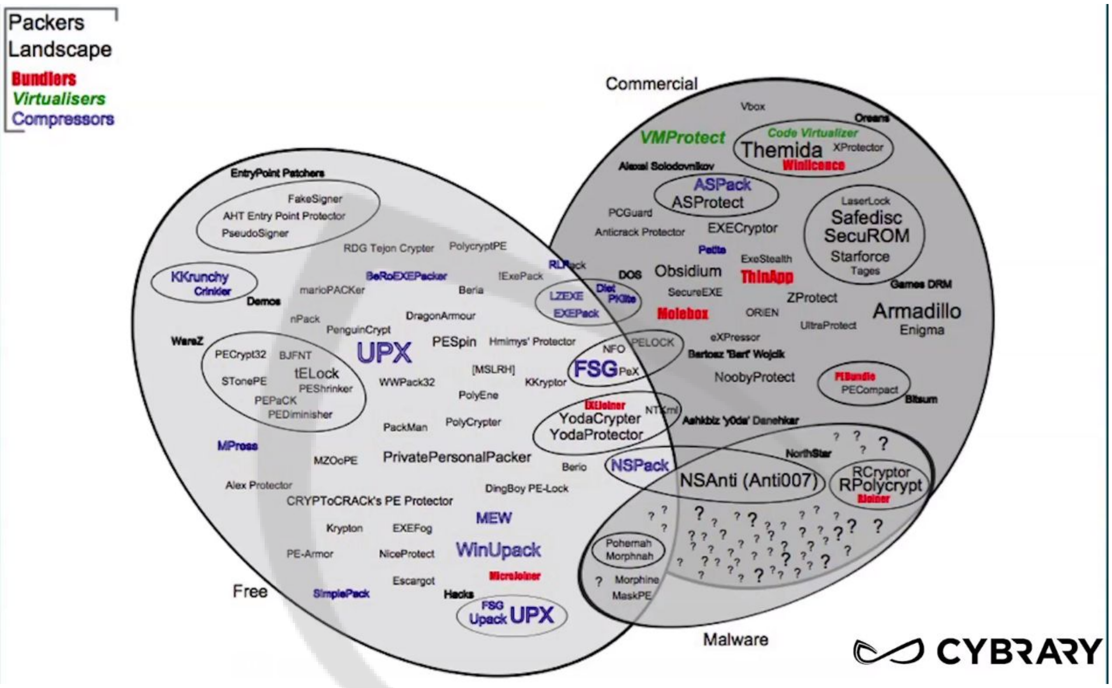
Module 6: Packers

Lesson 6.1: Packers Part 1

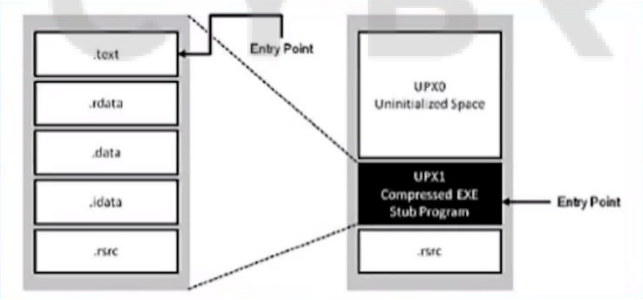
Skills Learned From This Lesson: Packers, Introduction, Theory

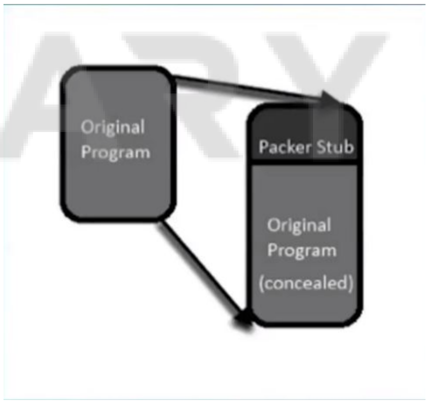
- What are Packers?
  - Self-decrypting executables
  - Originally made for compressing code size
  - Use tools such as
    - Debuggers
    - Memory Dumpers
  - Characteristics:
    - Packer strings/advertisements
    - Few strings
    - Few imports
    - High entropy data
    - Large virtual sections with small raw disk size
  - Goals:
    - Hide strings
    - Change the hash
    - Mask binary signatures
- Legitimate Users
  - Code Compression
  - Intellectual Property Protection
  - Anti-Reverse Engineering
  - Anti-Cheat
  - Digital Rights Management (DRM)
    - Licensing
- Common Packers
  - UPX
  - Armadillo
  - ASPack
  - VMProtect

  - ○ Themida

- 

- Changes in Code

- 



Lesson 6.3: Packers Part 3

Skills Learned From This Lesson: Packers, Packing, Demo

- Packing Example
  - Before Packing
    - Illusion Bot Strings
    - Illusion Bot AV Detections
    - Illusion Bot PE Sections
  - After Packing
    - No Strings
    - Fewer AV Detections
    - Different PE Sections

Lesson 6.4: Packers Part 4

Skills Learned From This Lesson: Packers, Unpacking, Demo

- Unpacking Demo
  - UPX Packed Regshot
    - Strings
    - PEiD
  - Pack Regshot (run)
    - Strings after
    - PeiD -> deep. Data base from SANS
  - Unpack Regshot “upx -d”
    - OllyDbg 1.10 (OllyDbg 2 will auto unpack)
    - Find OEP
  - Dump
    - OllyDump or
    - OllyDbg PE Dumper 3.03 or
    - LordPE
  - Reconstruct IAT
    - ImpREC 1.7e

Lesson 6.5: Packers Part 5

Skills Learned From This Lesson: Packers, Advanced, Theory

- More Advanced Packers
    - Multiple Layers
    - Adds Junk Code
    - Built in defenses
        - Anti-Analysis Code
        - Anti-Debugging Code
    - Custom Encryption
    - Create Small Virtual Machine

Module 7: Malware Defenses

Lesson 7.1: Malware Defenses Part 1

Skills Learned From This Lesson: Malware, Defenses, Introduction

- Defense Categories
    - Anti-Debugging
        - API
        - Process and Thread
        - Hardware and Register Based
        - Exception Based
        - Modified Code Based
        - Timing Based
    - Anti-Virtual Machine
        - API
        - Memory Constants
        - File/Process Names
    - Anti-Disassembly
        - Tricky Assembly
        - Dynamic Code Generation/Calling
    - Misc.
        - Anti-Analysis Tools
    - Malware Goals:
        - Stop Automated Analysis
        - Slow down Malware Analysts
- Basic Anti-Debugging Example

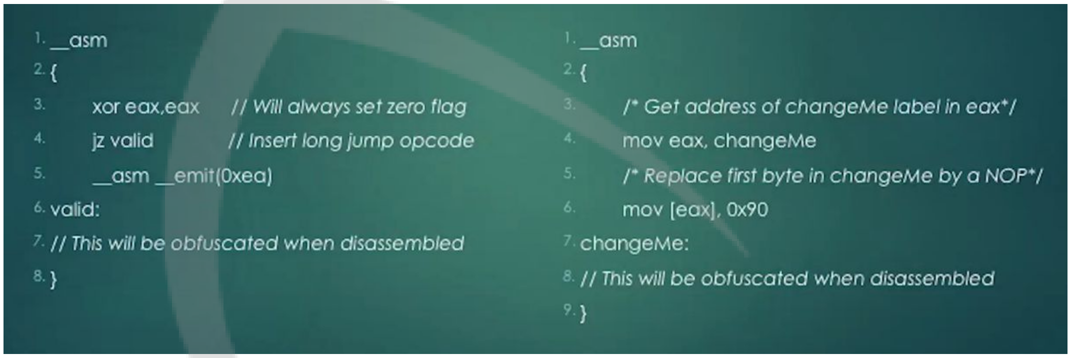
    - if(IsDebuggerPresent() == TRUE) {
        - exit(0); // Debugger is detected
    - }

Lesson 7.3: Malware Defenses Part 3

Skills Learned From This Lesson: Malware, Anti-Debugging, Techniques

- Anti-Debugging Techniques
    - IsBeingDebugged()
    - CheckRemoteDebuggerPresent()
    - FindWindow()
    - OutputDebugString()
    - NtQueryInformationProcess(ProcessDebugFlags)
    - NtQueryInformationProcess(ProcessDebugObjectHandle)
    - NtQueryInformationProcess(ProcessDebugPort)
    - NtSetInformationThreadDebuggerDetaching
    - SeDebugPrivilege OpenProcess
    - DebugActiveProcess()
    - NtGlobalFlag
    - PEB ProcessHeap Flag Debugger
    - LDR_Module Flags
    - Vista TEB System DLL Pointer
    - GetTickCount and TimeGetTime
    - Process names check
    - int 0Xcc scanning
    - and many more
- Anti-Virtual Machine Techniques
    - Process name check
    - LDR_Module
    - VMWare LDT Register Detection
    - VMWare STR Register Detection
    - VMWare special I/O instruction
    - Checks special VT-x or VMM instructions
    - Timing checks
    - Registry Checks

    - Virtual MAC address
    - Virtual hardware names
    - Anti-Cuckoo
    - Checks common VM drive ID’s
    - And many more
- Anti-Disassembly Examples
    - From: https://blog.sevagas.com/?Fun-combining-anti-debugging-and



```
__asm
{
    xor eax,eax     // Will always set zero flag
    jz valid        // Insert long jump opcode
    __asm __emit(0xea)
valid:
// This will be obfuscated when disassembled
}
```

```
__asm
{
    /* Get address of changeMe label in eax*/
    mov eax, changeMe
    /* Replace first byte in changeMe by a NOP*/
    mov [eax], 0x90
changeMe:
// This will be obfuscated when disassembled
}
```

- Anti-Anti-Debugging Techniques
    - Modify (Patch) the Malware
        - Patch the memory
        - Anti-Anti-Anti-Debugging: Integrity Checking Malware
    - Hook Function Calls
    - Run without a Debugger
        - Log API calls
        - Dumps memory