FOR610 – Reverse-Engineering Malware

Topics

= Concatenate join cells / chars together . . . 3–129

/usr/share/remnux/objects.js . . . 4–8

=¿ Operand
- LEA R9=¿local$_6$28 $- loadingtheaddressoflocal_628intoR9$ . . . 2 − −156
- Indirectly memory addressing . . . 2–156

Registry Key
- Persistence Presence . . . 1–43

0x values end of ShellCode . . . 4–93

0x30 hex PEB FS:[0x30] list of loaded modules . . . 3–113

0x40 VirtualAllocEx PAGE$_E XECUTE_R EADWRITE$ . . . 4 − −1381768$.py$ . . . 3 − −95

Function
- AllocateVirtualMemory . . . 5–26, 4–109
- BlockInput . . . 5–76, 5–92
- CheckRemoteDebuggerPresent . . . 5–12
- CloseClipboard . . . 2–144
- CreateMutex . . . 2–143
- CreateProcess . . . 5–51, 5–54, 2–85
- CreateRemoteThread . . . 4–108, 4–113, 4–123
- CreateThread . . . 4–109
- CreateToolHelp32Snapshot . . . 2–61, 5–85
- CreateUserThread . . . 4–109
- CryptDecrypt . . . 4–90-92, 1–99
- CryptEncrypt . . . 1–97
- EnumProcesses . . . 4–108
- FindResource . . . 2–140
- FindWindow . . . 5–79
- FindWindows . . . 5–92
- GetAsyncKeyState . . . 2–144, 2–175
- GetClipboardData . . . 2–144
- GetCursorPos . . . 5–66
- GetForeGroundWindow . . . 5–66
- GetKeyState . . . 2–144
- GetLocalTime . . . 5–13
- GetModuleHandle . . . 5–77, 5–92
- GetProcAddress . . . 5–53
- GetSystemTime . . . 5–13
- GetTempFileName . . . 2–61
- GetTempPath . . . 2–121
- GetTickCount . . . 5–13, 5–66
- GetWindowText . . . 2–144
- HttpAddRequestHeaders . . . 2–46
- HttpOpenRequest . . . 2–46
- HttpSendRequest . . . 2–46
- InternetConnect . . . 2–46
- InternetOpen . . . 2–46
- InternetOpenA . . . 2–47
- InternetReadFile . . . 2–46
- IsDebuggerPresent . . . 5–6
- KdDebuggerEnabled . . . 5–92
- LoadLibrary . . . 5–53, 4–123, 5–146
- LoadResource . . . 2–140
- LockResource . . . 2–140
- OpenClipBoard . . . 2–144
- OpenKey . . . 4–109

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

OpenProcess 4–108, 4–114
OutputDebugString 5–12
PINK TAGS 1–1
Process32First 5–85, 4–108
Process32Next 5–85, 4–108
ProtectVirtualMemory 5–26
QueryInformationProcess 5–12, 5–38
QuerySystemInformation 4–109
QuerySystemTime 5–13
ReadFile 4–87, 1–91
ReadProcessMemory 4–127, 4–129
RegOpenKey 2–18, 4–87, 5–89, 4–109
ResumeThread 5–54
RtlDecompressBuffer 5–26, 5–29
SetWindowsHookEx 5–67, 5–67-70, 5–70, 4–126
ShellExecute 2–154
SizeofResource 2–140
UnmapViewOfSection 5–53, 54
URLDownloadToFileA 3–72
VirtualAlloc 5–52, 5–54, 4–92, 4–108
VirtualProtect 4–127
wcsicmp 2–106
WinExec 3–72, 3–156
WriteProcessMemory 5–52, 5–54, 4–108, 4–116, 4–127
WriteVirtualMemory 5–26, 4–109

Instruction
ORANGE TAGS 1–1

Register
PURPLE TAGS 1–1

Structure
EXCEPTION_REGISTRATION 2–179
KBDLLHOOKSTRUCT 2–179
PROCESS_INFORMATION 2–94
STARTUPINFOA 2–94

Instruction
ADD 2–38
AND 2–38, 2–40
CALL 2–21, 2–39, 40
CMP 2–40, 41
IMUL 2–38
JA 2–41
JB 2–41, 2–106
JC 2–106
JCC 2–41
JE 2–41
JG 2–41
JL 2–41
JMP 2–23, 2–40, 41
JZ 2–41
LEAVE 2–83
LOOP 2–39
LOOPcc 2–109
MOV 2–23, 2–30, 2–38
NOP 5–9
OR 2–38
PUSH 2–22
RDTSC 5–13

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

RET 2–39, 2–83
SCASB 2–23
SETZ 2–185
SHL 2–38
SHR 2–38
SUB 2–38, 2–40
TEST 2–40

Register
CS code segment 2–26
DS data segment 2–26
EAX add/multi/return value 2–25
EBP 2–67
EBP reference arguments passed in func 2–25
EBX generic registers 2–25
ECX counter looping 2–25
EDI memory transfer instructions 2–25
EDX generic registers 2–25
EIP 2–93
EIP points next instruction to exec 2–26
ES 2–26
ESI memory transfer instructions 2–25
ESP point last item on stack 2–25
FS 2–26
GS 2–26, 5–100, 5–127
RIP 1–93
SS stack segment 2–26

64-bit malware
Two types, Browser Helper Objects (BHO) Device Drivers (rootkits) 2–148

64bit Assembly Registers
Whats the difference 2–150

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

A

ActiveXObject interact with system components 4–9

Addressing mode 2–23, 2–32

Administrative rights
- Admin Privileges always run malware as this. 1–40

AMSI Event logging on always Event viewer¿ASL¿Mic¿Anti¿Scan$_Int$ .... $3--141 AMSI Example$ .... $4--10-11$

AMSI Windows Antimalware Scan Interface examine scripts in Windows VBA XLM PS JS VBS 3–139

AMSIScriptContentRetrieval 3–140

AMSITrace logman start monitoring and save log file 3–140

AMSITrace logman stop monitoring 3–140

AMSITrace.etl log file 3–140

Analysis detection 5–91

Anti-Debugger 5–6
- Cloaking Capabilities 5–15
- Countermeasure 5–15
- ScyllaHide 5–15

Any.run 1–13

API 2–19

API Call
- GetLocalTime 5–14
- GetSystemTime 5–14
- GetTickCount 5–14
- NtQuerySystemTime 5–14

API Hooking Definition 4–146

API Monitor
- Examine API Calls made by processes 1–101

Arguments
- 32bits PUSH above CALL 2–156
- 64bits RCX RDX R8 R9 2–156

Arguments passed in registers on the stack 2–156

arguments.callee.toString() 4–18

ASLR 4–35
- disable Dynamic base 5–54
- Disable per file basis 4–36
- Disable prior unpacking analyzing code level analysis 4–35
- DllCharacteristics PE Header 4–36
- DynamicBase Flag 4–36
- ImageBase Field Virtual Memory Address executable perfer loaded 4–35
- PE Header file relocation section translate mem addresses 4–35
- rebasing 4–36
- setdllcharacteristics -d disable DynamicBase 4–36

Assembly Instructions
- (64bit) SUB
  - SUB RSP, 0x658, start of prologue, makes room on stack for local variables 2–156
- ADD (MSB)
  - MSB, Most significant bit, unsigned/signed binary math 2–182
- ADD interpretations
  - How it works in detail 2–183
- Branching and Conditionals
  - JMP, CALL, CMP, TEST 2–38
- Data Manipulation

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Arithmetic (ADD,SUB,IMUL) Boolean (AND,OR) Bit Manipulation (SHR,SHL) 2–38
Data Transfer
PUSH,/POP, MOV, XCHG, LEA 2–38
Example
PUSH, MOVE, TEST Column2 13
Three Categories, data manipulation, data transfer, branching and conditional 2–38

Automated analysis 1–13

Avast Antivirus 5–80

AVCaesar 1–13

avghookx.dll 5–76

B

Balbuzard toolkit 5–18

base64dump.py 3–88
-a view contents of string 3–88
-d parameter dump decoded string 3–90
-n 10 only consider strings 10 bytes long 3–90
-s parameter select string 3–90
-s select string 3–88
-t utf16 convert to UTF-16 3–89
. right side output Unicode UTF-16 char two bytes 3–88
gunzip process 3–90
scans code for base64 3–90

base64dump.py -s select ID -d Dump contents to file 4–90

bbcrack.py auto identify patterns decode alg XOR ROL ADD crackguess 5–19

Beaconing
Brief periodic messages to the adversary 1–17

Behavioral Analysis
ProcMon, ProcHack, RegShot, fakeDNS 1–34

BeingDebugged (flag) 5–12

Binary Ninja 2–6
Disassembler strong automated analysis 1–80

64-bits 2–150

BlockInput 5–74

Botnet C2 - Large number of infected hosts connecting to webserver 1–117

box.js automatically decode 4–18

BP 1–92

BPX 1–92

Branch Instructions
Direct code execution to another location 2–39

Branching
Conditional and Unconditional Jumps
CALL, RET and Looping 2–39

Break statements 2–130

Breakpoint
Hardware 5–106
hardware 5–121
SingleShoot 5–159
VirtualProtect 5–152

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Browser Helper Objects (BHOs)
  Internet Explorer (64bit) wont load 32bit vice verse 2–148

Browser-specific scripts 4–6

brutexor.py 5–18

brxor.py 5–18

Burp
  Network Sniffer and Web Proxy 3–19

Byte 8bits 2–28

C

CALL E8 Instruction 3–112

Calling convention
  cdecl
    Args passed to function pushed onto stack right to left. Return value is in EAX register, code block calls function to clean up ie printf() 2–72
    how it works 2–74
  fastcall
    Args stored in registers, extra Args placed in stack, callee responsible cleaning stack, return value is EAX. ECX and EDX pass args 2–73
    how it works 2–75
  how data gets in and out of functions, code block calls function and performing housekeeping 2–72
  stdcall
    Calling functions in WIN32 API, function itself responsible removing args pass in stack, smaller code size. 2–72
    how it works 2–74
  thiscall
    how it works 2–75
    includes reference this pointer, Microsoft ECX holds this pointer and callee cleans args on stack, GNU this pointer pushed to stack last and caller cleans 2–73

Capa
  Focuses on automatic identify capabilities - uses ATTCK and MBC 1–86
  report 1–87

capa
  -vv parameter verbose 5–46
  CreateProcess 5–46

CAPE 1–13

Capstone 2–6

cat examine contents of files 3–118

cat read contents of file 4–88

Censys 1–13

CFF Explorer 4–36
  Nt Headers 4–36

CFF Explorer - Static analyzer extract files
  extract file and code analysis 2–145

Cobalt Strike 3–95

Code analysis
  Dynamic 1–81
  Ghidra, windbg OllyDumb, Scylla 1–34
  Static 1–81

Code Analysis - Imports, Strings, Functions 2–162

Code Injection Techniques and Why 4–126

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Code lifecycle
- Compiler 2–5
- Executable File 2–5
- Linker 2–5
- Loader 2–5
- Object Code 2–5
- Source, Compiler, ObjCode, Linker, Libraries, File, Loader 2–5

Command and control (C2) 1–17

Compiler 2–5

Compound (complex) Expressions
- multiple conditions 2–123

Concatenate join cells together 3–129

Core Malware Analysis Techniques 1–6

CREATE$_SUSPENDED - dwCreationFlags$ .... $5 -$
$-49CreateThreadcreatesCPUthreadandlaunchesvariableshellcode$ .... $4 - -92$

CScript advantage auto supply script OS objects 4–12

CScript natively supports eval 4–12

CScript uses WScript.Echo to display 4–12

CScript Windows script interpreter 4–7

curl
- /.curlrc
  - curl config files to spoof HTTP headers 3–19
- same as wget, used to download files using CLI 3–19

CurrentVersion/Run 1–44

Cutter 2–6
- Code Analysis Toolkit 1–80

CyberChef
- Decoding, deobfuscating, decrypting data 1–104
- Output 1–104
- Recipe 1–104
- XOR
  - How to use XOR key to decode 1–106

D

Data Sizes
- byte/word/dword/qword
  - 8bit/16bit/32bit/64bit - Assembly Instructions (dword) 2–28

Data Structure
- representation of information, access and manipulate that representation 2–34

Data Structure - Common Examples
- Strings, linked lists, network sockets, file handles, arrays 2–34

Debugger 1–81
- Detection 5–78

Debugger Process 4–46

Decompiler vs Dissassembly 2–159

Decompiling
- Generate approx original program source code 1–81

Deep Freeze
- Physical Snapshot/backup 1–33

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Dereferencing
  Pointer Access
    Address the pointer points is accessed ... 2–29

destination 00000000 offset jump 0 bytes ... 3–112

Detect It Easy (DIE) ... 1–48

Device Drivers (rootkits)
  Digitally signed trusted cert, TDL bypass ... 2–148

Digital Fingerprints
  hash values example ... 1–46

Direct Addressing
  Address is specified in a Operand ... 2–31

Disassemblers and Debuggers
  Ghidra, IDA, Windbg, Cutter, Binary Ninja, Hopper ... 1–80

Disassembling
  Translating binary machine instructions to human language code ... 1–81

Disk Cloning ... 1–33
  Clonezilla ... 1–33
  dd ... 1–33
  Fog ... 1–33

DLL ... 1–14, 2–137
  how it works
    library to help share code with multiple programs, export functions, need EXE to load, etc ... 2–137

DLL needs EXE to load it into memory exports functions share code with multiple programs ... 2–137

DllRegisterServer run a Windows DLL ... 3–138

DNS ... 2–45, 1–66

DNS leakage
  Observing source DNS queries to malicious domain ... 1–19

document.body.appendChild ... 4–6

document.parentNode.insertBefore ... 4–6

document.write ... 4–6

DownloadFile inseng.dll exports funtions lesser alternative ... 3–138

Dropper ... 2–140

dwCreationFlags ... 5–49

Dword 32bits ... 2–28

Dyre ... 5–64

E

EA → Effective Address ...

EBP Frame Pointer -8 access local var +8 reference parameter ... 2–68

Effective Address (EA)
  Calculated Address or resides in register ... 2–32

EFLAGS ... 2–26

EIP 32bit ... 1–93

EIP Register stores address of instruction execute next ... 3–112

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Emulators
- binee 1–82
- Capa 1–82
- Imitates behavior CPU, OS, runtime components, preview actions 1–82
- Qiling 1–82
- Speakeasy run$_speakeasy.py - tfilename.exe - ospeakeasy.json2 > speakeasy.txt$ .... $1 - -82$
- Vivsect 1–82

Emulators - INetSim
- Emulate HTTPS, SMTP,FTP,POP3,TFTP,IRC 1–119

encoded.hex 1–75

Entry Point
- Original 5–104

Enviroment variable appdata
- users profile directory 1–38

Equation Editor 3.0 3–104

ESP Stack Pointer register top of stack Push Pop 2–68

eval deobfuscation process 4–6

evilclippy doc pass removal 3–72

exceptions error conditions code usually handle 5–29

Executable File 2–5

Exeinfo PE 5–95
- Tools used to generate exe 1–48

Exercise 2.3
- How many Arguments 2–80-81
- How many Arguments and Local Variables 2–89
- How many Local Variables 2–82
- How many Locations 2–80, 2–88
- Knowledge of Functions
  - Locations of functions, number of args, number of local vars, instructions comprise pro/epi, calling convention used 2–79
- What calling convention does FUN$_4064c8uses(cdecl)$ .... $2 - -85$
- Which instruction comprises the epilogue(LEAVE,RET) 2–84
- Which instruction comprises the epilogue(POP,POP,MOV,POP,LEAVE,RET) 2–91
- Which instruction comprises the prologue(PUSH,MOV,SUB) 2–83
- Which instruction comprises the prologue(PUSH,MOV,SUB,PUSH,PUSH,PUSH) 2–90

Exercise 2.5
- Loop analysis
  - assess stopping condition(CMP, JC) 2–116
  - control variable before stopping condition(ADD instruction) 2–117
  - control variable before the loop (XOR ESI,ESI 2–118
  - decompiler output 2–120
  - loop body by the stopping condition 2–119

Exercise 2.8
- 64bit DLL Ghidra
  - ShellExecuteW, SUB RSP, 0x658 start of code block, LEA places pointers 2–156

Exercise 3.4
- oledump 791+62
  - Size os compiled code and size of compressed code 3–45
- oledump Concealed,obfuscated URL inside Dim array1(12) 12 lines 3–48
- oledump grep
  - Extracts lines that start with a keyword in " 3–49
- oledump Macro decoded using strings
  - Function DU(yI) array1 = Xf book3 50
- oledump nh = Split(Replace(pn(array1), Fw, ''), zM)book3 50

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

oledump numbers-to-string.py -j join 3–57
oledump OLE2
examining structure and contents of OLE2 files 3–45
oledump particulars.doc -i
examine contents OLE2 in particulars.doc 3–45
oledump particulars.doc -s a -v pipe grep -v $'^{pipemore}$
-v to grep eliminate lines that match regex. $^{i}sstartoflineandmatches''$ .... 3 – –47
oledump particulars.doc -s a -v pipe more
-s a Extracts VBA Macros from ALL streams 3–46
oledump Public Const zM As String = –—book3 50
oledump re-search.py -n str-u
Extracts strings within the " 3–49
oledump regsvr32
Function U5 Macro 3–52
oledump sed ss/====//g!Remove ==== separators 3–49
oledump sets.py join "
Join lines into a single line 3–49
oledump StrReverse
Function U5 Macro 3–52
oledump Uppercase M
means Macro inside that object 3–45

Exfiltration
Stolen data keystroke logs 1–17

exiftool
extracts metadata and can include other details about the document 3–53

Exploit Kit
automate exploit of client side vulns 3–19

Exploit What Is 3–105

Exploring Network Interactions 1–111

Expressions
¡ less than
if (a ¡ 4) = if a is less than 4 (JNL jump if not less than) 2–124
¿ greater than
(a ¿ 1) if a is greater than 1 (JNG jump if greater than) 2–124
Compound Expression (AND)
if (a ¡ 4) (a ¿ 1) = if a is less than 4 AND greater than 1 2–124

extractscripts.py auto extract JS from webpages 4–7

F

FakeDNS 1–118

fakedns
listen and automatic respond all host DNS queries UDP 53 1–67

feh
image1.jpg 3–41

feh Image Viewer 3–14

Fiddler
Expect-CT Response
Common CloudFlare Websites minimize risk of misused SSL certificates 3–22
Extract Artifacts 3–23
HTTP Intercept, capture HTTPS, Decrypt HTTPS, AutoResponder, Enable rules. 1–123
Network Sniffer and Web Proxy 3–19
Request 3–23
Response raw tab

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

example 3–22
steel.saz file types
saz file types is a Fiddler capture file 3–20

file 3–39

File Sections
grouping of code and data, .text .data .rsrc .reloc 1–46

file see format of files 3–118

FindRes, LoadRes, SizeRes, LockRes, WriteFil, CreateProcess - Dropper 2–140

FindWindow function 5–77

FLOSS 5–25
-no-static-strings parameter regular non-obf strings 5–26
-x parameter address function handle deobf 5–26

FLOSS emulate execution deobf strings decode stack strings 5–25

fn callback function
nCode, wParam, lParam, WM-KEYDOWN,UP,SYSKEYDOWN,UP 2–179

format-bytes.py 3–107
-f parameter name-eqn1 3–107

Frame-Based Handling 5–98

FS Register address data structure TIB 3–113

FS:[0] 5–98

FS:[30h] 5–12

Function
Calling
branch (jump to another location), executing CPU processes instructions 2–65
Calling Control Transfers
Calling Function (Pass in parameters, save return pointer, transfer control function) 2–66
Epilogue
Epilogue (cleans up stack eg POP variables, restores registers) 2–67
Hot patching
2byte MOV EDI, EDI NOP instruction - without need to stop and restart the program 2–67
Hot Patching - ShortLong Jumping
2-byte relative jump transfers control to another jump anywhere in memory, 5-byte long jump is 5 NOP instructions 2–67
Prologue
Prologue (allocates space for variables, saves registers reused in function body) (MOV EDI, EDI - NOP instruction) 2–67
Returning Control Transfers
Returning Function (Set return value EAX, clean stack restore registers, transfer control saved return pointer) 2–66

function call interfere with debugging of script 4–11

Function Execution Top to Bottom2 13

Functions
group of instructions/code that performs a tasks, send data, log keystrokes, r/w files 2–65
Three components
Input passed into function from caller, body of function code perform task, value returned from function 2–65

G

GetEIP CALL POP 3–112

GetEIP CALL saves EIP value to stack, POP EAX places stack value in EAX 3–112

GetFileSize file handle scanning 3–114

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

- GetModuleHandle function to locate DLLs 5–76
- GetProcAddress
  - hModule, lpProcName 5–51
- Ghidra
  - =¿ operand
    - LEA R9=¿local$_6$28 $- loadingtheaddressoflocal_628intoR9$ .... 2 − −156
  - Apply Function Data Types 2–48
  - Bad instruction 5–118
  - Browser Field Formatter
    - Resizing fields, Field Categories, Field Controllers 2–15
  - Bytes to ASCII
    - 00 bytes between each character means Unicode String 2–63
  - Calls are wrapped in Libraries (DLLs) 2–19
  - Convert 5–22
  - Create Project 2–8
  - Data / String
    - lpszVerb-00413d8c pointer to a string right click, data, string 2–49
  - Data Convert 2–49
  - Data Structure
    - individual elements called members 2–94
  - Data Type 2–47
  - Defined Strings 2–103
    - locate reference to string 2–105
    - reveals filenames and directories of interest 2–104
  - Defined Strings - show references
    - Right Click on Label (Application$_0$0413$e34)References - ShowReferences$ .... 2 − −103
  - Disassembler and decompiler 1–80
  - et Equate
    - Text representation of a numerical value 2–36
  - Exports Symbol Tree
    - Exports Entry points into the program 2–16
  - File Structure .data
    - Contains data 2–12
  - File Structure .rdata
    - Contains read-only data 2–12
  - File Structure .reloc
    - Contains relocation data fix addresses 2–12
  - File Structure .text
    - Contains executable code 2–12
  - Function Call HTTP C2 APIs
    - Outgoing Call of the same Windows API 2–56
  - Function Graph
    - blue arrow code block ends uncondition jump 2–14
    - green arrows condition met 2–14
    - red arrows condition not met 2–14
    - Visualizing loops, complex conditionals within functions. 2–14
  - Function Graph - Red/Green Arrow
    - Red Arrow = Unsuccessful (FALSE), Green Arrow = Successful (TRUE) 2–53
  - Function Size
    - Number of bytes comprise the function, more work to reverse 2–58
  - Headers 2–12
  - Import Address Table
    - APIs (functions) used by the program contained in external DLLs 2–17
  - Imports Symbol Tree
    - Imports Program dependencies 2–16
  - Interpret Arguments
    - Double click lpPrefixString$_0$0413$d04 - rightclickDATlabel, data, TerminatedUnicode$ .... 2 − −63
  - JC Jump if carry - Table of Strings (Loop)

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

ADD ESI, 0x4 - 4 bytes, CMP ESI, 0x38 - 56 bytes (total number of strings in table 4 bytes per one) 2–106
Key binding 5–23
Local Variable Conflict 2–94
local$_6 28 decompiler$
Use decompiler if possible to assist 2–159
OneShot
Propagate External Parameters 2–48
Parse C Source
Resolving Symbolic Constants that Ghidra is unsure of 2–49
Process Information Structure
similar to Data Structure 2–94
Program Tree 2–12
Project Analysis 2–10
Propagate External Parameters
Data Type Manager - Apply Function Data Types, One Shot 2–48
Renaming Functions
Right click Function - Edit Label 2–57
Sections 2–12
Set Comments
Click line and press semicolon 2–44
Static Code Analysis Tool NSA 2–6
Strings - Group with a Loop
dword ptr [ESI + PTR$_{uW} indows_0 0418cd8]ESI0 = Windows, ESI4 = Boot, ESI8 = System.TableofStrings2-$ -106
Symbol References - IOC API calls
Example, Registry API Calls for persistence 2–18
Symbol References Menu
List of API (functions) 2–17
System Calls same as API Calls 2–19
TLS 5–116
Windows API (CreateMutexA) 2–143
Windows API (Find/Size/Lock Resource)
Nearby calls to WriteFile/CreateProcess, malware extracts resource writes to disk 2–140
Windows API (KeyState, WindowText,Clipboard)
malware collecting information and keylogging 2–144
XREF
Same as right click Reference and Show References 2–106

Ghidra - CALL instruction
Performs a function call 2–21

Global Descriptor Table (GDT) 5–126

grep
-v :DollarSign colon end of line 3–66
-v $^g otoremovelinesstartgoto$ $3 - -66$

H

Handle 1–83

Handle -0x1 Special handle to read current process 4–148

Hard-coded IP
set-static-ip ipaddress of malware 1–125

HashSets 1–13

Hex to ASCII Tool
xxd -r -p filename.hex 1–73

Hextostring 3–67

HKCUWindows execute with specific file extension 4–7

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Honeypot 1–20

Hooking
- API Calls ReadProcessMemory Read first bytes Trampoline 4–146
- API Calls VirtualProtect modify permissions make writable 4–146
- API Calls WriteProcessMemory JMP 0xE9 PUSH 0x68 RET 0xC3 JMP 4–146
- Call table hooks 4–143
- Call table hooks IDT Interrupt Descriptor Table 4–143
- Call table hooks Import/Export Tables 4–143
- Call table hooks IRP I/O Request Packet 4–143
- Call table hooks SSDT System Service Descriptor Table 4–143
- Call table hooks vftable Virtual Function Table 4–143
- GUI hooks 4–143
- GUI hooks driver layering/filtering 4–143
- GUI hooks minidrivers 4–143
- GUI hooks SetWindowsHookEx API Call 4–143
- Inline User-mode Definition patch target function memory 4–143

Hooking Definition and Process - Intercepting Exec compiled 4–143

Hopper 2–6
- Disassembler and Decompiler 1–80

hProcess parameter handle points process program create new thread 4–132

HTTP Command and Control
- Most common - HTTP libraries transition between plaintext HTTP and encrypted HTTPS 2–45

HTTP Command and Control Windows API
- InternetOpen, InternetConnect, HttpOpenRequest, HttpAddRequestHeaders, HttpSendRequest, InternetReadFile. 2–46

httpd
- http request 1–72
- Web server NGINX 1–71

Hybrid Analysis 1–13, 1–15

I

ICP IP, Hostname, Encoded Hex 1–73

IDA 2–6
- Disassembler with built-in debugger 1–80

If-Else 2–99-100

If-ElseIf-Else 2–101-102

If-ElseIf-Else Statements (Switch)
- Nested. Switch statements are the same but different Syntax 2–101-102

Import Address Table (IAT) 5–163

Imports
- DLL libraries required for execution 1–46

Indicator of Compromise (IOC) 1–17, 1–44

Indirect Memory Addressing (square brackets)
- [EAX] access allo mem. [EBP + 0x10] access data on stack, [EAX + EBX *8] access and array 8bytes 2–32

INetSim
- /etc/inetsim/inetsim.conf
  - main conf, services emulate ports, HTTP response 1–119
- /var/lib/inetsim Data
  - data files provided by HTTP or SSL Cert 1–119
- /var/log/inetsim Logs
  - service.log details requests received, responses provided 1–119

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

/var/log/inetsim/service.log Output
shows connection details, IOCs, insights 1–119
DNS
inetsim.conf remove poundsign start-service dns, dns-default-ip to remnux. httpd stop to avoid conflicts ... 1–119
Emulate HTTPS, SMTP,FTP,POP3,TFTP,IRC 1–119
FTP 1–119
IRC 1–119
POP3 1–119
SMTP 1–119
TFTP 1–119

Inlined Functions
Body of function placed in block code that calls function. no extra overhead entering/exiting. String related function strlen strcmp 2–77

Inlined Functions strlen example 2–78

Instructions
Execution
fetch from mem, decode instruction perform action based on instruction, act on registers and mem locations 2–24
Operand
Square Brackets - Brackets mean fetch data at the specified address MOV EAX , [0x410230]. Classed as direct addressing as de-referencing immediate value 2–32
Operand addressing modes
Immediate value 2–23
memory address used as operand 2–23
Register 2–23
Operand Implied (Implicit) by Instructions
SCASB - Scans memory EDI register points at value equal to AL register 2–23
Stack Operations - move data to and from Extended Stack Pointer (ESP) 2–23
Operand Layout
Destination First - Source Second 2–23
Operands
Have 0 - 3 Operands, A register, Memory location, Immediate value 2–23

Instructions have two components
Operation and Operand 2–23

Intercepting IP Connections 1–127

Intermodular Calls - API Calls 4–51-52

Internet access Lab
Malicious Websites, exploit kits, C2, etc 1–18

Intezer Analyze 1–13

IP redirection 1–126

iptables
accept-all-ips 1–126
Linux Firewall, NAT intercept redirect network traffic 1–126
set-static-ip 1–126

IRC
Internet Relay Chat
Controlling malicious code 2–45

IsDebuggerPresent 5–6

J

Javascript Deobfuscation rely on objects, methods, attributes 4–7

JavaScript Detection if modified 4–18

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

JavaScript What are they . . . 4–4

Jcc instructions
  Conditional Jumps
    Perform Jump if condition is met - JZ, JB, JE, JG, etc . . . 2–42

JMP instruction
  directs execution to different code blocks depending on value of EAX . . . 2–184

JNE is equivlant to JNZ . . . 5–7

jq
  JSON Parser . . . 1–85

js-beautify . . . 4–5, 4–88

JScript Micsoroft version of CScript differences . . . 4–11

Jump Example . . . 2–43

Jump instruction (JMP)directs execution to different code blocks depending on value of EAX . . . 2–184

Jump table . . . 2–131, 2–184

Jump Tables - Optimization Technique
  list of address of each code block (an array) - . . . 2–131

Jumps
  Conditional
    Branch to a new location in memory only after a condition is met. jcc and loop instructions . . . 2–40
  Conditional - Multiple Instructions
    Disadvantage, they modify values of destination operand. . . . 2–41
  JNG Jump
    (a ¿ 1) if a is greater than 1 (JNG jump if greater than) . . . 2–124
  JNL Jump
    if (a ¡ 4) = if a is less than 4 (JNL jump if not less than) . . . 2–124
  Jump - If not below zero example(JZ / JNC)
    JNC = CMP EAX, EDI, if EAX is larger than EDI size of buffer, buffer is too small and will error . . . 2–128
  Jump - JNL
    if (a ¡ 4) = if a is less than 4 (JNL jump if not less than) . . . 2–124
  ump - JNG
    (a ¿ 1) if a is greater than 1 (JNG jump if greater than) . . . 2–124
  Unconditional
    branch to a new location in memory, JMP, CALL, RET Instructions . . . 2–40

K

KBDLLHOOKSTRUCT
  How this function works . . . 2–180

KdDebuggerFlag . . . 5–78

Keylogger Capabilities . . . 2–171

L

Linker . . . 2–5

Loader . . . 2–5

LoadLibrary Malicious Process . . . 4–142

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Loops
  Appear in Malware
    Encrypt/Decrypt, C2, Port Scan, DDOS, Log Keys 2–108
  Common types
    for loop (init, stop, update conditions) while loop (init is separate from stopping condition and update control variables in body of loop) do while loop (same as while but gets executed least once before condition) 2–112
  Do while (loop) 2–112
  five major components
    Control Variable, Loop initialization, Loop body, Loop update, Stopping conditions 2–110
  For (loop) 2–112
  For (loop) example 2–113
  Loop Example - strlen 2–111
  While (loop) 2–112
  While (loop) example 2–114

Loops - Exercise 2.5 2–120

Loops / Loopcc
  Max jump range of 128 bytes, loopnz, ZF bit in flags register 2–109

Loops to execute single code JZ JNZ JE 2–109

lpStartAddress address of function target process run new thread 4–132

M

MAEC Malware Capabilities project 1–11

Malicious Documents and Scripts 1–7

Malicious VBA Macros 3–37

Malicious Windows Executables 1–7

Malware 1–5
  definition, design, explaination of malware 1–5
  malware is code used to perform malicious actions 1–5
  what is malware and malware analysis 1–5

Malware analysis 1–5
  Fully automated
    Detailed Reports, Registry, mutex values, network traffic, quickly assess 1–9
  Input
    Verbal reports, Sus files, file/mem images, network logs, anomaly obs 1–10
  Interactive Behavior
    Running program isolated lab to reveal behaviour 1–9
  Manual Code 1–9
  Output
    Malware Behaviour, Identification, Attacker Profile, IR Reccoms, Reports IOCs, Trends 1–10
  Results 1–10
  Stages
    Manual, Interactive, Static, Fully Automated 1–9
  Static Analysis
    Metadata, strings, structure, header data 1–9

Malware Analysis Techniques 1–6

Malware Analyst 1–8

Malware Behavior Analysis Configuration Change
  what change caused what new behavior 1–70

Malware Behavioral Analysis 1–52

Malware Hash Registry 1–13

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Malware lab 1–24-25
  Configuration 1–28
  Isolation
    Bugs, Weakness, Exploit, Escape VM 1–28
  Network Isolation
    Host-Only, or Stronger Separation Virtual Network - No Host 1–29

Malware report 1–11

MBC Malware Behavior Catalog 1–11

Memory Address or Offset Example
  00401007, 008, 009 first column2 13

Memory BreakPoints 5–156

Memory Protection 5–157

memset copies contents of shellcode to memory 4–92

Metadata importance of saving headers, URLs, Last-Modified etc 4–15

MetaDefender 1–13

Misdirection 5–95, 5–138

Most significant bit (MSB) 2–182

MSB
  Most significant bit 2–182

mshta.exe Microsoft HTML Applications HTA files execute JS outside browser 4–77

Mutant1 15

Mutex 1–15, 2–143

MZ
  Windows Executable 2–142

N

NanoLocker 5–140

Native API Calls 4–128
  NtAllocateVirtualMemory/ZwAllocateVirtualMemory 4–128
  NtCreateThreadEx/ZwCreateThreadEx 4–128
  NtopenProcess/ZwOpenProcess 4–128
  NtQuerySystemInformation 4–128
  NtWriteVirtualMemory/ZwWriteVirtualMemory 4–128
  RtlCreateUserThread 4–128

nCode
  CallNextHookEx - if zero 2–179

Netcat 3–60

netcat listener
  sudo nc -l -p 3127 1–126

Network Sniffer 3–19

NOP instruction does nothing 5–9-10

NOP Padding shellcode 909090 hex 3–109

nslookup 1–67

NtGlobalFlag 5–13
  FLG$_{H}EAP_{E}NABLE_{F}REE_{C}HECK(0x20)$ .... $5--13$
  FLG$_{H}EAP_{E}NABLE_{T}AIL_{C}HECK(0x10)$ .... $5--13$
  FLG$_{H}EAP_{V}ALIDATE_{P}ARAMETERS(0x40)$ .... $5--13NtQueryInformationProcess$ .... $5--40$

NTVDM (NT Virtual DOS Machine) 2–149

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

numbers-to-string.py -j join ... 3–57

O

Obfuscation Techniques concealing encoded encrypted obfuscated ... 4–6

Object Code ... 2–5

object.js
  document.write ... 4–8
  eval ... 4–8
  href location ... 4–8

objects.js ... 4–8

OEP Original Entry Point ... 4–39

Office Documents
  Macros have been disabled - Enable Content ... 3–40
  OLE2 - Compound Document File V2 CDFV2 ... 3–38
  OLE2 - Compound File Binary Format CFBF ... 3–38
  OLE2 - Object Linking and Embedding
    Considered to be Legacy format. ... 3–38
  OLE2 - Structure Storage
    defines how to treat a single file entity. ... 3–38
  Open Office XML OOXML ... 3–38
  VBA Macros
    embedded macros in a file ... 3–37
  XML ... 3–38

OLE2
  Macros ... 3–38

oledump
  -A parameter show Hex values and ASCII ... 3–79
  -S parameter extract Strings ... 3–87
  -d dump contents stream ... 3–88
  -i information data structure ... 3–78
  791+62
    Size os compiled code and size of compressed code ... 3–45
  c suffix compiled ... 3–79
  CDFV2 Compound Document Format ... 3–84
  Concealed,obfuscated URL inside Dim array1(12) 12 lines ... 3–48
  example VBA stomping ... 3–78
  grep
    Extracts lines that start with a keyword in ” ... 3–49
  Macro decoded using strings
    Function DU(yI) array1 = Xf book3 ... 50
  nh = Split(Replace(pn(array1), Fw, ”), zM)book3 ... 50
  numbers-to-string.py -j join ... 3–57
  OLE2
    examining structure and contents of OLE2 files ... 3–45
  oledump.py particulars.doc -i
    examine contents OLE2 in particulars.doc ... 3–45
  oledump.py particulars.doc -s a -v pipe grep -v $'^{pipemore}$
    -v to grep eliminate lines that match regex. $^{i}sstartoflineandmatches''$ ... $3--47$
  oledump.py particulars.doc -s a -v pipe more
    -s a Extracts VBA Macros from ALL streams ... 3–46
  plugin$_{h}ttp_{h}euristics$ ... $3--71$
  Public Const zM As String = ––—book3 ... 50
  re-search.py -n str-u
    Extracts strings within the ” ... 3–49

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

regsvr32
  Function U5 Macro 3–52
s suffix source 3–79
sed ss/====//g!Remove ==== separators 3–49
sets.py join ”
  Join lines into a single line 3–49
SRP
  contained cached compiled copy of the VBA Macro can reveal aftifacts from previous version 3–53
StrReverse
  Function U5 Macro 3–52
unable to decompress 3–79
Uppercase M
  means Macro inside that object 3–45
VBA Purging 3–85

oledump.py
  plugin$_biff-xlocateandexamineXLMMacro$ .... $3--142olevba$
  -decode automate 3–142
  AutoOpen function example 3–44
  locate decode extract tool good for auto scanning files 3–43
  olevba particulars.doc ¿ particulars.olevba 3–43
  risky capabilities
    AutoOpen - runs when document is opened 3–43
    Call - call a DLL using XLM/XLF 3–43
    exec - run an executable file or command using XLM/XLF 3–43
    Hex Strings - Hex-encoded strings 3–43
    StrReverse - obfuscate strings -deobf 3–43
    URLDownloadToFileA - download files from internet 3–43
  VBA Stomping 3–77

OllyDbg 5–77

OllyDumpEx 4–54
  Get EIP as OEP 32bit 4–54
  Get RIP as OEP 64bit 4–54
  MEM$_WRITECharacteristicsFlags$ .... $4--55$
  memory access violation error 4–55

OllyDumpEx API code strings 5–107

open file handle scanning 3–114

Open Office XML OOXML
  4 letter extension 3–38
  Macro follow OLE2 format 3–38

Open Threat Exchange 1–13, 1–16

OpenProcess API Call obtains handle existing process PID 4–133

Operand =¿
  LEA R9=¿local$_628-loadingtheaddressoflocal_628intoR9$ .... $2--156OROperator$ .... $2--37$

OS provide managed access for user space code
  allocating memory requires mgmt of system ram 2–19

OSINT 1–17

P

p-code 3–77

p-code compressed and compiled 3–38

P2P 2–45

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Packed Executable . . . 4–38
- dumping file PE Header tweaked . . . 4–39
- dumping process packed exe unpack in memory extraction . . . 4–39
- Entry Point . . . 4–39
- Import Address Table IAT . . . 4–39
- OEP Original Entry Point . . . 4–39

Packers
- AddressOfEntryPoint . . . 4–24
- Armadillo . . . 4–25
- Bytehist . . . 4–26
- Detect It Easy . . . 4–26
- Entry Point . . . 4–24
- Exeinfo PE . . . 4–26
- FSG . . . 4–25
- How it Works . . . 4–22
- PE Header Values . . . 4–24
- PeStudio . . . 4–26
- Sections . . . 4–24
- Signs and how to identify them . . . 4–26
- Themida . . . 4–25
- UPX . . . 4–25
- What is . . . 4–22

Packers Bytehist . . . 4–28
- bottom region order of occurance Y-axis . . . 4–28
- examine file byte value visual histogram . . . 4–28
- holes in them file is packed . . . 4–28
- Not Packed Less uniform byte values . . . 4–28
- Packed more uniform byte values book4 . . . 28
- top region bytes value 0 to FF X-axis . . . 4–28

Packers DetectItEasy diec commandline . . . 4–29

Packers Exeinfo PE GUI flagged Tampered File . . . 4–29

Packers PeStudio . . . 4–27
- Entropy higher value more random contents . . . 4–27
- Raw size space section occupy file disk . . . 4–27
- Section names different to nonpacked .txt .data .rdata . . . 4–27
- Virtual size Memory windows allocate section malware exec . . . 4–27

Page Memory Rights . . . 5–161

PAGE$_E XECUTE_R EADWRITEVirtualAlloc0x40$ . . . $4 - -92 PassiveTotal$ . . . $1 - -13$

Password Protected VBA Macro . . . 3–65

Patching . . . 5–9
- Change JNE with JE . . . 5–9
- NOP instruction does nothing . . . 5–9-10

Payload What Is . . . 3–105

pcode2code . . . 3–81

pcodedmp.py . . . 3–79

pcodedmp.py disassemble VBA p-code . . . 3–80

PDF
- /Filter /FlateDecode
  - Stream - encoded contents - endstream . . . 3–8
- AA
  - action automatically when pdf is open . . . 3–4
- AcroForm
  - Execute embedded javascript . . . 3–4
- Elements

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

header (version specs), objects (text, fonts, graphics, JS), xref (modify without rewriting file), trailer (metadata and file specs) 3–5
EmbeddedFiles
Launching external or embedded programs 3–4
Encode Streams
/FlateDecode stream encoded contents endstream 3–8
header
version specs 3–5
How it works
exploit vulns, ASCII keywords, JavaScript, Links, ... 3–4
Indirect Object References
/AA (AutoAction) /O (when file is opened) 43 0 R (Reference Object 43) 3–6
Launch
Launching external or embedded programs 3–4
Objects
reference each other and specify actions 3–6
objects
text, fonts, graphics, JS 3–5
OpenAction
action automatically when pdf is open 3–4
Risky Keywords
/JS, /JavaScript, /AcroForm, /XFA, /Launch, /EmbeddedFiles, /OpenAction, /AA, /URI, /SubmitForm 3–4
SubmitForm
interact with websites (URL) 3–4
trailer
metadata and file specs 3–5
URI
interact with websites (URL) 3–4
XFA
Execute embedded javascript 3–4
xref
modify without rewriting file 3–5

PDF Password Encrypted 3–35

PDF-Parser
-r Reference object to objects 3–15
/Annots 3–16
/Annots /Page
/Type /Page / Annots have interactive capabilities - popup notes, clickable links 3–16
/DCTDecode
Stream contains Image can be decoded using /DCTDecode, standard for JPEG 3–13
/Image within /XObject
/Type /XObject - /Subtype /Image (confirms contains image) 3–13
/Page 3–16
/URI
PDF type known as dictionary. key-value pairs 3–9
/XForm
Images embedded into the PDF 3–12
/XObject
Embedded Objects ie: image with clickable link 3–9
Images embedded into the PDF 3–12
/XObject embedded image 3–32
-f -w -d
dump stream to file 3–35
export PDFPARSER$_{O}PTIONS = -O$
similar to -O parameter 3–30
how it works 3–9
Javascript /JS /AcroForm /XFA 3–35

steel1.pdf -o 6
Examine specific object number 3–13
steel1.pdf -a
-a displays statistics about keywords in file 3–12
steel1.pdf -k /URI
-k parameter shows values with associated KEY /URI (multiple URIs) 3–12
steel1.pdf -o 6 -d object6.jpg
-o 6 select object 6 -d extract it and dump to file 3–14
steel1.pdf -r 6
Follow the trail of references 3–15
steel2.pdf -O -a
scans inside /ObjStm objects to find more objects 3–30
Streams
store data, scripts such as text, font definitions, images. encoded but decoded by applying /Filter 3–13

PDFid 3–8
-n
Only show non-zero keywords and outputs 3–29
ObjStm Object Stream
Object stream that contains other object - indirect objects , stream compression filters 3–29
OpenAction
automatically initiate risky actions 3–8
URI
clickable URLS 3–8

pdftk
PDF Password Encrypted (decrypt) 3–35

PE (Portable Executable) 1–48

PE Tools 4–39

pe_unmapper 5–162
/base parameter 5–162
/in parameter 5–162
/out parameter 5–162

PEB NtGlobalFlag 5–13

PEB Process Environment Block list of DLLs 3–113

PECompact 5–95

peframe
Static Analysis Display Attributes, hashes, indicators 1–47

Persistence 1–44

pestr
extracting strings, encoding -l unicode -a scan whole file 1–42

PEStudio
Imports risky DLLs LoadLibrary 5–145

PeStudio
(Resource executable)
dump MZ filebook2 142
DLL file (Exports)
entry point (address of first instruction to execute), (Malware use) Ordinal alt method export/import functions, setting function 2–138
Exports 2–138
Static Analysis Display Attributes, hashes, indicators 1–45-46
TLS 5–115

Pev Toolkit
Tools examining Windows exe 1–43

PF File Extension
Prefetched allow OS process start faster 1–60

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Pinpoint
  wget on steroids 3–19

Pointer
  variable contains the address of location in memory - points to addresses in memory 2–29

Position-independent code (PIC) 2–150

PowerShell 3–61
  -encoded -EncodedCommand -NoProfile 3–86
  -nop -NoProfile 3–86
  -w hidden -WindowsStyle hidden 3–86
  DollarSign s Variable saves command 3–89
  GzipStream decompress contents 3–89
  IEX Invoke-Expression execute decompressed commands 3–89

PowerShell Integrated Scripting Environment ISE 4–93

PowerShell ISE 4–93
  .ps1 file extension 4–95
  decode and save BIN file 4–96
  Toggle Breakpoint F9 4–95

ProcDOT
  Correlate, Filter Visualize Procmon Logs PCAP 1–63-65
  Visualizes Process Monitor Logsbook1 52

Process Environment Block (PEB) 5–12

Process Hacker
  Detailed Process Explorer Tool 1–52
  handle
    x64dbg, ProcHacker 1–95
  Process list, services, network connections 1–53

Process hollowing 5–44, 5–59

Process hollowing explaination 5–52

Process Monitor 1–54
  Filter
    Process Name, Process Identifier (PID) 1–61
  Process Interaction, Reg, File Sys, Network 1–52
  Process Tree 4–76

PXE booting 1–33

Q

qpdf
  PDF Password Encrypted (decrypt) 3–35

Quttera 1–13

Qword 64bits 2–28

R

Radare 2–6

rar x extract contents of compressed file 3–118

RCX RDX R8 R9 order 2–156

RDTSC Read Time-Stamp Counter 5–14

reg$_e$*xportcommandlinetool* .... $4 - -86$*Register*
  Edit 5–9

Register-based CPU 2–24

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Registers
  (32bit Architecture)
    EAX32/AX16/AH8/AL8 - ECX32/CX16/CH8/CL8 - EBX32/BX16/BH8/BL8 2–27
  (32bit)(General Purpose)
    EAX, ECX, EBP, ESP, ESI/EDI 2–25
  (32bit)(General Purpose) EAX
    EAX, ECX, EBP, ESP, ESI/EDI 2–25
  (32bit)(General Purpose) EBP
    Reference arguments passed into a function and local variables within a function 2–25
  (32bit)(General Purpose) ECX
    Used as a counter (Loops) 2–25
  (32bit)(General Purpose) ESI/EDI
    Memory transfer instructions 2–25
  (32bit)(General Purpose) ESP
    Point to the last item on a stack and affected by stack-related functions 2–25
  (32bit)(Special Use)
    hold flags and track program execution 2–26
  (32bit)(Special Use) EFLAGS
    Outcome of computations and control operation of CPU 2–26
  (32bit)(Special Use) EIP/RIP
    points to next instruction to execute - EIP 32bit / RIP 64bit 2–26
  (32bit)(Special Use) Segment
    CS, DS, ES, FS, GS, SS 2–26
  (32bit)Types
    General purpose ECX, particular use, or both 2–24
  (64bit Architecture)
    RAX-EAX-AX-AH-AL 2–151
  (64bit Architecture) R9 Register
    R9 - R9D - R9W - R9B 2–152
  (64bit) (General Purpose)
    RAX, RBX, RCX, RDX and R8 through R15 2–150
  (64bit) (Special Use) - RIP
    RIP reference memory locations 2–150
  (64bit) - Calling convention (fastcall)
    First four parameters passed RCX, RDX, R8, R9. Additional parameters stored on stack 2–150
  (64bit) RSP
    SUB RSP, 0x658 used to make room on stack for local variables. RBP is not used 2–155
  32bit compared to 64bit
    InternetOpenUrlA - Windows API 2–160
  AH register 8 bit 2–27
  AL register 8 bit 2–27
  AX register 16 bit 2–27
  CPU on-chip memory locations 2–24
  EAX register 32 bit 2–27

Registry 1–43

Registry Export Powershell Command 4–86

Regshot
  Changes in File Sys and Registry 1–52
  Names of files to be renamed sys restart 1–59-60
  Snapshot Registry before after infection 1–55

Resource Starvation
  depriving malware from resources. 1–70

Reversing Malicious Code 2–4

RIP 64bit 1–93

RIP-relative addressing 2–150

RiskIQ 1–16

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

ROL
  Algorithms rotate right, rotate left ... 1–103
RollBack Rx
  Physical Snapshot/backup ... 1–33
RootKits
  Digitally signed trusted cert, TDL bypass ... 2–148
ROR
  Algorithms rotate right, rotate left ... 1–103
ROT
  Rotates alphabet chars by a certain number of positions (ROT13) ... 1–103
RSP stack pointer instead of EBP ... 2–150
RTF ... 3–100
  Equation Editor Exploit ... 3–104
  braces and define groups ... 3–101
  control word character ... 3–101
  objects, binary contents, hexdec chars, ... 3–101
RTF Rich Text File ... 3–100
RTF Risks ... 3–100
  allow arbitrary files ... 3–100
  dont support macros ... 3–100
  extract embedded object percent temp percent execute ... 3–100
  Javascript / VBScript ... 3–100
  OLE1 Packager Object Server ... 3–100
  Vulnerability Exploit execute shellcode assembly instructions ... 3–100
rtfdump.py ... 3–102
  -H parameter convert its contents from hex ... 3–109
  -O parameter show objects only ... 3–103
  -d parameter dump object ... 3–103
  -s parameter selects index ... 3–103
  ¿ output to file ... 3–103
  90 Hex NOP Padding shellcode ... 3–109
  embedded objects ... 3–102
  h= tag hex chars ... 3–109
  magic number ... 3–102
  nested groups ... 3–102
  Remainder Overlay content exist outside file ... 3–108
  u= tag unknown chars ... 3–108
rtfobj similar to rtfdump.py ... 3–103
runsc32 -f run file ... 4–101
runsc32 attach file ... 4–102
runsc32 execute shellcode ... 3–116
runsc64 execute shellcode ... 3–116

S

Sandbox detection ... 5–62, 5–64, 65, 5–91
Sandboxie ... 5–80
sc32 long series hexdec shellcode ... 4–92
scdbg Debug Shell ... 4–99
  .poke specify place bytes worth of data ... 4–100
  g command emulate exec of shellcode ... 4–100
  k command show current state of stack ... 4–100

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

scdbg shellcode emulator 3–94

scdbgc 3–94
- /f parameter select file 3–94
- /foff 3B specify where shellcode starts 3–114
- /fopen open handle to original RTF document 3–115
- /norw hide Read/Write events 3–115
- /s 1 no limits on steps 3–94

Scout
- wget on steroids 3–19

Script Debuggers in Web Browser 4–18

Script Deobfuscation specific value Objects.js file 4–15

Scylla
- IAT 5–163

Scylla How to use 4–40

ScyllaHide 5–15, 5–28

selector 5–126

Self-defense
- Detect VM, monitor, tools, confuse code, terminate, sleep 1–30-31

self-modifying code
- code-data duality - modify themselves during runtime 2–35

self-referencing code
- code-data duality - reference themselves during runtime 2–35

setdllcharacteristics -d 4–37

Shell.Application ShellExecute 4–9

shellcode 3–93
- debug 3–93
- disassemble 3–93
- emulate execution 3–93
- emulation example 3–111
- extract strings 3–93
- Invoke API Calls DLLs 3–113

ShellCode end of values 0x 4–93

Shellcode sc32 4–92

Shellcode What Is Machine CPU Code 3–105

Short circuit evaluation
- skips a code block Boolean and fails or succeeds 2–123

Signed binary math 2–182

Simple expressions
- single condition 2–123

Small Code Base
- Benefits
  - Increasing modularity, drawback additional overhead 2–77

Snapshot
- Save and restore state of VM 1–32

Speakeasy 1–83
- API Call 1–83
- Handle 1–83
- Unsupported API
  - Unfimilar API Call CryptDeriveKey 1–84

Special Use Registers 2–26

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

SpiderMonkey js -f command 4–88

SpiderMonkey js Linux script interpreter 4–7

Spoof HTTP Headers 3–19

Square Brackets [EBP + 0x10]
  Indirect Memory Addressing, [EBP + 0x10] access data on stack, ... 2–33

SRE → Software Reversing Engineering . . . .

SRP
  Strings or oledump
    look for ASCII and Unicode strings or oledump to extract 3–53

SSH Daemon sshd start transfer files between VMs 4–87

Stack
  example how it works 2–69-70
  ADD ESP 0x1c after CALL cleaning stack 2–92
  Cleanup 5–104
  How it works
    LIFO, PUSH, POP, ESP Stack Point, EBP Frame Point (EBP -8 accessing local Variable, EBP +8 referencing parameter) Compilers pop value off stack 2–68-69
  Last In, First Out 2–68-69

Stack Optimisation
  PUSH subtracts 4 from ESP value at location - MOV place value on stack but does not subtract from ESP . 2–76

Stack String 5–21

Stack String → String,stack . . . .

Standalone Script Interpreter 4–7
  CScript 4–7
  SpiderMonkey js 4–7

Static Analysis Tools
  PeStudio,strings, CFF, peframe, DIE 1–34

Static properties
  Analysis 1–34
  file/sect hash, packer, embedded, imp/exp, strings 1–42

strace (Strace-for-NT) Tool
  monitors and analyses system calls 2–19

strdeob.pl 5–23

String 2–34
  Stack 5–21

String References 4–51-52

Strings 1–42-43

strlen
  analysis example 2–111

strlen example 2–78

Structured Exception Handling (SEH) 5–97
  Frame-Based 5–98
  SEH Chain 5–98

Switch Statement - C to assembly 2–132

Switch Statements - How it works
  enables one of many code blocks to be executed based on integer value of a variable. code blocks terminate break statement 2–130

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Symbolic Constant
  hex representation of the value PUSH 0x80000001 Set Equate HKEY$_Current_User$ . . $2--36 SymbolicConstants$
  Visual Studio SDK
    Load WinInet.h header file 2–49

SysAnalyzer 5–80

Systeminternals tool System Monitor
  sysmon monitors system call register and files 2–19

T

TcpLogView what is 1–52

TDL RootKit
  Digitally signed trusted cert, TDL bypass 2–148

Thread Information Block (TIB) 5–98

Thread Local Storage (TLS) 5–114

ThreatAnalyzer 5–80

Thug
  client-side JavaScript 3–19
  Emulating browser 3–19
  Honeyclient 3–19
  honeyclient , mimic a web browser, executing javascript, emulating web browser 3–19

TIB Thread Information Block currently running thread 3–113

Tinba 5–64

TOR
  Adversaries can track exit nodes 1–19

TOR Socks
  utility for tunneling SOCKs Traffic through TOR 1–19

tr -d delete null characters -d delete specific characters 4–88

Transfer Files between Virtual Machines 4–87

translate.py used to decode XOR Strings 3–93

TrickBot 5–164

trid 3–39

U

Unpacker Scripts x64dbg Repo specific versions of packers 4–34

Unpacker Tools 4–39
  Imports Fixer 4–39
  PE Tools 4–39
  Scylla fixes PE Header rebuilding IAT 4–39-40

Unpacker Tools Automatic 4–34
  Ether 4–34
  UnPacMe 4–34

Unpacking 5–144

Unsigned binary math 2–182

Upatre 5–64

UPX Packer 4–33
  -d decompress 4–33
  modified/hacked/protected mangled file prevent decompress 4–33

UPX Unpacker Debugger OEP lots of zeros JMP is end 4–48

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

URLDownloadtoFileA Windows API call download file urlmon 3–138
urlscana.io 1–13
UTF-16 3–89

V

Variable
  Access Restrictions
    enforced by the compiler not the CPU 2–54
  Global
    Used anywhere and referenced using an address 2–54
  Local
    Variable accessible by function that allocates it. Function exits, code free to be reused 2–54
  Represents some piece of information (code or data) without knowing actual contents (local$_a$$c0$) .... $2--54$
  Static
    Used within function allocates it, cannot be reused, not referenced in Stack 2–54
Variable Types
  Local, Global, Static 2–54
Variables and Arguments Local Tab 5–151
Varible pr saving pointer to new memory segment 4–92
VBA Debugging 3–72
  Debug.Assert False 3–73
VBA Emulation 3–55
VBA Macro Password Protected 3–37
VBA Purging removes p-code 3–85
VBA Stomping 3–77
VBA Stomping removes source code 3–85
VBA Tampering 3–78
ViperMonkey 3–55
VirSCAN 1–13
virtual-key codes 2–179
VirtualAlloc 4–111
  Malware unpacking calling function extract memory 4–111
  memory allocated auto init zero 4–111
VirtualAlloc 0x40 PAGE$_EXECUREREADWRITE$ $4--92 VirtualAllocallocatenewmemoryforshellcode$ $4--92$
VirtualAlloc Definition reserves commits changes state 4–111
VirtualAllocEx + WriteProcessMemory = Code Injection Pattern 5–50
VirtualAllocEx allocate mem inside process inject code or data 4–111
Virtualization
  Emulates hardware to VM and software to network within host 1–26
Virtualization detection 5–63
VirusTotal 1–13, 14
vmonkey 3–55
VPN
  OpenVPN, WireGuard, AlgoVPN 1–19
Vulnerability What Is 3–105
vURL 1–13

W

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Web Proxy 3–19

Websites - Ways to investigate 3–19

wget
  /.wgetrc
    wget config files to spoof HTTP headers 3–19
  on steriods 3–19
  same as curl, used to download files using CLI 3–19

WH_MOUSE_LL 0x0E 5–68

WH$_C BT$ . . . . . $2--175WH_K EYBOARD$ . . . . . $2--175$

WH$_K EYBOARD_L LSymbolicConstant0D - Dec13$ . . . . . $2--175WinDbg$ . . . . . $5--80$
  ScyllaHide 5–15
  Windows Debugger 1–80

windbg 4–48
  API Calls interactions with its environment DLLs 4–68
  Breakpoint 4–49
  Call Stack 4–108
  cleardb command clear all breakpoints 4–64
  database files c-progfiles-x64dbg-x32-db 4–64
  database files c-progfiles-x64dbg-x64-db 4–64
  Disable DynamicBase before Debugging 4–47
  Dump 5–31
  Dump lpBuffer 5–55
  EAX register contain VirtualAlloc Unpacked Code Dump 4–114
  Execute till return 4–113
  Follow in Memory Map 5–56
  Follow To 4–108
  hardware breakpoint likely to remain during runtime 4–69
  Ignore Exception 5–101
  Memory Map 4–65
  Memory Map E Flag - Execute code 4–65
  Save Patch 5–11
  Scylla Fix IAT Table AFTER OllyDumpEX Process - Fix Dump 4–58
  SEH 5–99
  Set breakpoints in API Calls 4–63
  SetBPX advapi32.RegOpenKeyExA 4–106
  SetBPX RtlDecompressBuffer conceal the string 5–30
  software breakpoint F2 Toggle can be ignored, modifying program 4–69
  Step into 4–50
  Step over 4–50
  TLS 5–119
  Unpacked Code Dump - CFF Explorer edit Entry Point PE Header 4–53
  Unpacked Code Dump - Scylla 4–53
  Unpacked Code Dump Plugin - OllyDumpEx 4–53
  Unpacked Code Intermodular Calls - API Calls 4–51-52
  Unpacked Code String References 4–51-52
  xAnalyzer 5–151

Windows API accept arguments using flags
  combined symbolic constants, binary or operator 2–37

Windows API Calls 5–12, 4–127
  (GetKeyState)(GetAsyncKeyState)(SetWindowsHookEx)
    Two Common Methods for collecting keystrokes 2–171
  CheckRemoteDebuggerPresent 5–12
  CloseClipboard
    closes the clipboard 2–144
  CreateProcess 5–52
  CreateRemoteThread 4–127

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

CreateToolHelp32Snapshot . . . 5–83
CreateToolHelp32Snapshot get listing of running processes . . . 4–127
CryptDecrypt
  places decrypted data into buffer whose address passed to function name pbData . . . 1–99
CryptDeriveKey
  Windows Cryptographic capabilities . . . 1–46
CryptEncrypt
  API Call Decrypts or Encrypts. . . . 1–97
EnumProcess . . . 4–127
GetAsyncKeyState
  How the function works - KBDLLHOOKSTRUCT . . . 2–180
  key is currently down or up, was pressed, last call to GetAsyncKeyState . . . 2–144
GetClipboardData
  gathers data from clipboard, usernames,passwords and other data . . . 2–144
GetKeyState
  retrieves status of specified key . . . 2–144
GetProcAddress
  hModule, lpProcName . . . 5–51
GetThreadContext . . . 5–52
GetWindowText
  obtains text of windows title bar combined with keystate APIs, keys logged and in what window . . . 2–144
InternetOpenUrlA
  64bit RCX, RDX, R8, R9 detailed . . . 2–167
IsDebuggerPresent . . . 5–6
LoadLibrary . . . 4–127
LoadLibraryW loads designated DLL . . . 5–51
NtGetContextThread . . . 5–52
NtQueryInformationProcess . . . 5–12
NtSetThreadContext . . . 5–52
NtUnmapViewofSection/ZwUnmapViewofSection . . . 5–52
OpenClipboard
  access to clipboard and locks access . . . 2–144
OpenProcess . . . 4–127
OutputDebugPresent . . . 5–12
OutputDebugString . . . 5–12
Process32First . . . 5–83
Process32Next . . . 5–83
ProcessFirst . . . 4–127
ProcessNext . . . 4–127
RegSetValueExA
  Capability to set registry values . . . 1–46
ResumeThread . . . 5–52
SetThreadContext . . . 5–52
SetWindowsHookEx
  used to install hook procedure, args to function type messages, mouse events, keyboard . . . 2–173
SetWindowsHookEx Arguments
  idHook, lpfn, hMod, dwThreadId . . . 2–175
VirtualAllocEx . . . 5–52, 4–127
VirtualAllocEx 0x40 flProtect P$_{ER}W$ . . . $5 - -50$
VirtualProtect change permissions Injection/unpacking . . . 5–152
WriteProcessMemory . . . 5–52, 4–127
WriteProcessMemory lpBuffer parameter . . . 5–53
ZwQueryInformationProcess . . . 5–12

Windows API Calls -ReadFile
  Kernel32.dll read date from I/O device . . . 1–91

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

Windows API Documentation 2–20
  A ANSI 8 bit character 2–20
  Ex Extended update function 2–20
  Inputs - Parameters 2–20
  Outputs - Return Value 2–20
  Purpose 2–20
  W Wide 16 bit charactor 2 bytes UTF-16 2–20

Windows Hooks
  Hooking Procedures
    used to monitor events - mouse movements, key presses, window interactions 2–173

Windows Loader fixup 2–150

Windows Virtual PC 5–80

WinSCP Windows Tools transfer files between VMs 4–87

WINWORD.EXE 3–117

Wireshark
  DNS Noise
    microsoft.com 1–66
  DNS query
    Log example 1–66
  GET Request
    Request server to provide website or file 1–73
  HTTP Connection Details
    HTTP GET Request via httpd and accept-all-ips 1–129
  RST ACK
    Packet indicating connection not established 1–66
  Sniffing Network Traffic and how to start capture 1–56
  Sniffs Captures Network Traffic 1–52
  TCP Stream
    Shows entire traffic capture 1–73

WM_LBUTTONDOWN 0x201 5–70

WM_LBUTTONUP 0x202 5–70

WM_MOUSEMOVE 0x200 5–69

wmiprvse.exe Windows Management Instrumentation WMI Commands 4–77

Word 16bits 2–28

WOW64 Subsystem
  Windows on Windows, NTVDM, 32bit DLLs 2–149

WPE Pro 5–80

WScript.Shell implements Methods CreateShortcut 4–9

X

x32dbg
  EIP pointer
    32bit Address of the current instruction 1–93

x64dbg
  Breakpoints
    Set Breakpoints interrupt execution of program 1–90
  Call Operand
    Functions invoked in assembly 1–97
  Call Stack
    path of nested function in reverse chrono order program executed before pausing 1–96
  functions
    hexdec, relative addr, offsets. 1–89

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

RIP pointer
64bit Address of the current instruction 1–93
Run to user code
Finished executing the breakpoint to see what happens. 1–96
SetBPX, bpx, bp
Set Breakpoint command 1–92
Single Step
One instruction at a time 1–90
Static and Dynamic Code Analysis 1–89

xAnalyzer Locals Tab 5–151

XLM Function FORMULA.FILL places result into cell comma 3–129

XLM Macros 3–125
macro execution xmldump.py 3–126
macro execution zipdump.py 3–126
obfuscation techniques Value 3–131
xl/$_rels/workbook.xml.relsmappingIDandxmlfilesheetsstartingpoint$ ..... $3--127$
xl/sharedStrings.xml String Table Obfuscation 3–132
xl/workbook.xml sheets starting point 3–126

XLMMacroDeobfuscator automated analysis 3–142

XLSM Forumla 3–130

XLSM Reference 3–130

XLSM Tags c - Cell 3–128

XLSM Tags f - Formula 3–128

XML 3–38

xmldump pretty 3–54

xmldump.py
¡v¿17¡/v¿ 17th string xl/sharedStrings.xml 3–132
celltext 3–129
grep -A2 two lines before and after 3–132
r=cell number reference cell number 3–132
t=s string stored in table of shared strings 3–132

XOR
Change each char with 1-byte key value (5b) 1–103
Exclusive OR
Only true (nonzero) if two operands differ. XOR EDI EDI = 0 2–49

XOR = $^{i}nbashcommand$ ..... $3--93XORObfuscation$ ..... $5--117$

xor-kpa.py3 ..... 69, 70

XORI - XOR3 67

XORSearch 5–16-17
-i parameter case insensitive search 5–17
-s parameter generate file decode bytes with key and alg 5–17
How to use 5–17

xorsearch 3–111
-W parameter shellcode patterns 3–111
-d 3 parameter avoid applying ROT transformation FPs 3–111

xorsearch search file string encoded 3–111

xxd 1–75

xxd decode hex to ASCII
xxd -r -p filename.hex 1–73

xxd hex editor tool 4–88

Y

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template

yara-rules . . . 3–95
Yotalhash . . . 1–13

Z

zipdump
  Extracting individual files . . . 3–41
  particulars.doc -s 9 -d pip xmldump.py pretty pipe more . . . 3–54
  xmldump pretty . . . 3–54
  zipdump.py particulars.doc . . . 3–41
  zipdump.py particulars.doc -s 5 -d ¿ image1.jpg
    -s specify object with file -d dump the file ¿ filename . . . 3–41
zipdump.py -j parameter xmldump.py -j JSON format automatic analysis . . . 3–134
theindex

This work is licensed under a Creative Commons Attribution 4.0 International License, by A. Cailliau
Template by A. D'hondt, https://github.com/dhondta/tex-course-index-template