Windows Reverse Engineer

Basic through Advanced Development Path

# Basic

Skillsets

Memory Management

Intro

Hardware dependencies

Types

Physical memory

Virtual memory

User-mode vs. Kernel-mode

The PAGE

Memory Manager

MM very complicated s/w

Processes / Threads

Intro

Definitions

Structure

Private virtual address space

An executable program

A list of open handles to various system resources

A security context called an *access token*

A process ID

At least one thread of execution

Memory Structure

What you need to know

Executive Process (EPROCESS) structure

Process Environment Block (PEB)

CSR\_PROCESS structure

W32PROCESS structure

Processes are not named objects

A handle to a process provides access to **some** of the data in the EPROCESS structure

Assembly Language

Intro

What you need to know

Lowest level language

Not as complicated as some people fear it will be

As a reverse engineer, you read assembly language code

Compiler optimizations and obfuscations can be difficult to RE

Hardware vendors (chip manufacturers) define their languages

Tools

Debuggers

Intro

What you need to know

Provides you with an active view of a currently running program

Types of Debuggers

User

Variety of debuggers in user-mode

Kernel

In kernel-mode, there is only one choice for Windows, WinDbg

Workspaces

Symbol Files

Symbol file locations

Commands

Trace command and the step command

Various dump memory and registers commands

Dbghelp.dll – latest version

where.exe

findstr.exe

Task Manager

Debugging Tools for Windows

tlist.exe

Exercises

where.exe

where.exe plus findstr.exe

Task Manager

Find tlist.exe.

Find all dbghelp.dll’s on local drives. Which one is the one used?

Start your favorite debugger. Read/skim its help file. Read/study yearly.

# Intermediate

Skillsets

Memory Management

Hardware Dependencies

CR3 (physical address) → Page Directory

Virtual Address (Page Directory Index – 10 bits) + Page Directory → Page Directory Entry (PDE)

Page directory Entry (PDE) → Page Table beginning

Virtual Address (Page Table Index – 10 bits) + Page Table → Page Table Entry (PTE)

Virtual Memory hardware protection → page granularity, no segmentation protection

Windows does not use LDTR / LDT(s) only GDTR / GDT

Operating System Dependencies

X86 default – user(0x00000000 – 0x7ffeffff)

X86 default – kernel(0x80000000 – 0xffffffff)

64Kb gap – why?

X64 default – user(0x00000000’00000000 – 0x000007ff’fffeffff)

X64 default – kernel(0xffff0800’00000000 – 0xffffffff’ffffffff)

Much, much larger gap (0xFFFF0000’001001) – what does this accomplish?

IRQL

Processes / Threads

Important structures

A thread is defined by two structures – ETHREAD & KTHREAD

ETHREAD contains housekeeping information for the thread

KTHREAD stores scheduling information for the thread dispatcher

A process is defined by two structures – EPROCESS & KPROCESS

EPROCESS contains basic information about the process

KPROCESS

stores scheduling information for the process

Contents not shown – always changed by MS

Use WinDbg to display

System Calls (implementation undocumented)

Two data structures

service table descriptor (KSERVICE\_TABLE\_DESCRIPTOR)

array of function pointers/offsets

KeServiceTableDescriptor

native syscall table

KeServiceTableDescriptorShadow

native syscall table plus syscall table for GUI threads

Assembly Language

Opcode and zero or one or two operands

Program flow control dependent up the EFLAGS register

Basic Instructions

Moving Data

Arithmetic

Comparing Operands

Conditional Branches

Function Calls

Nop (No-Op) slide

DLL’s (Dynamic Link Libraries)

What is a DLL?

Map code into calling process’ address space

Implicit load-time linking, or

Explicit run-time linking

DLL never owns anything process owns it

Global/static variables not shared

copy-on-write mechanism

Implicit load-time linking

Explicit run-time linking

Exporting C++ classes restriction

extern "C" modifier

Name mangling

Tools

Debuggers

Evaluating Expressions

MASM vs C++ under WinDbg

Commands under WinDbg

~~pc – Step over until a CALL instruction is encountered~~

~~ph – Step over until a branching instruction is encountered~~

~~pt – Step over until a RET instruction is encountered~~

~~pct – Step over until a CALL or RET instruction is encountered~~

~~tc – Step into until a CALL instruction is encountered~~

~~th – Step into until a branching instruction is encountered~~

~~tt – Step into until a RET instruction is encountered~~

~~tct – Step into until a CALL or RET instruction is encountered~~

Debugger Extensions

IDA Pro

Intro

Popular disassembler / decompiler

**I**nteractive **D**is**A**ssembler

Licensing

~~Licensing is very strict. Named, Computer, Floating (network) licenses.~~

~~IDA Starter – 20 processors. No 64-bit files.~~

~~IDA Pro – 50 processors plus 64-bit files. 32-bit application. Soon switching to 64-bit.~~

~~Windows, Linux & Mac OS X~~

~~$$$~~

~~IDA Pro Floating License (MS Windows) $2534~~

~~x64 Decompiler License (MS Windows) $3525~~

~~Discount if ordering new IDA+Decompiler -$ 230~~

Interface

GUI is not good. Keyboard interface is.

Comments – Fantastic feature

Alternatives

Hopper – OS X and Linux

X64Dbg

Visual Duxdebugger

Immunity Debugger

Hiew

Radare2

ODA (**O**nline **D**is**A**ssembler)

Panopticon

Bokken

Binary Ninja

Depends

Dependency Walker

Free

32-bit or 64-bit modules

Builds hierarchical tree diagram of all dependent modules

Mainly DLL dependencies

Hex Editor

Several free options

Mainly used to examine binary code or ASCII strings

First tool for an unknown blob of code

Dumpbin

Dump embedded information on EXEs and DLLs

Quick way to see what APIs the file exports and imports

Exercises

WinDbg

Dumpbin

Hex Editor

Depends

IDA Pro

# Advanced

Skillsets

Memory Management

Exceptions

Page Faults

Working Sets

Kernel Memory and User Memory

Section Objects

VAD Trees

Objects & Handles

Named Objects

Processes / Threads

Context Switching

Synchronization Objects

Process Initialization Sequence

Application Programming Interfaces (API)

System Calling Mechanism

Assembly Language

DLL’s (Dynamic Link Libraries)

Windows Internals

Portable Executable (PE) format

Basic Concepts

Image Sections

Section Alignment

Headers

DOS header

PE header

Imports

Exports

Structured Exception Handling

C / C++

Compiler generates code the microprocessor can execute

C++ generates virtual address tables

Java / C#

Interpreted code

Requires JIT or interpreter

Tools

Debuggers

Breakpoints

Scripts

Debugger Extensions

Kernel-mode Debugging

Real machines

Virtual machines

IDA Pro

Converts opcodes to human-readable assembly language

Distinguishes code from data

~~Depends~~

~~Hex Editor~~

~~Dumpbin~~

SysInternals Suite

Process Explorer

Process Monitor

VMM Map

RAM Map

WinObj

AutoRuns

PE Browse

Channel 9

Defrag Tools

Exercises

WinDbg

IDA Pro

SysInternals Suite

Process Explorer

Process Monitor

VMM Map

RAM Map

WinObj

AutoRuns

PE Browse

Channel 9

Defrag Tools

# 1 – Building you USB thumbdrive (optional)

# 2 - Process Explorer

# 3 - Process Monitor

# 4 - Process Monitor – Examples

# 5 - Autoruns and MSConfig

# 6 – RAMMap

# 7 – VMMap

# 13 – WinDbg

# 14 - WinDbg – SOS

# 15 - WinDbg - Bugchecks (BSOD)

# 19 - WinDbg – OCA

# 20 - WinDbg - Basic Commands

# 21 - WinDbg - Memory User Mode

# 22 - WinDbg - Memory Kernel Mode

# 24 - WinDbg - Critical Sections

# 25 - WinDbg – Events

# 26 - WinDbg - Semaphores, Mutexes and Timers

# 27 - WinDbg - Configure Kernel Debugging

# 28 - WinDbg – Scheduling

# 29 - WinDbg - ETW Logging

# 30 - MCTS Windows Internals

# 89 - Symbol Folder Tools

# 94 - Sysinternals Strings, FindStr, !pde.ssz

# 96 - Writing a Debugger Extension Part 1

# 97 - Writing a Debugger Extension Part 2

# 98 - Writing a Debugger Extension Part 3

# 99 - Writing a Debugger Extension Part 4

#101 - Writing a Debugger Extension Part 5

#102 - Writing a Debugger Extension Part 6

#103 - Writing a Debugger Extension Part 7

#104 - Writing a Debugger Extension Part 8

#105 - Writing a Debugger Extension Part 9

#120 - Windows Management Instrumentation (optional)

#121 - DebugDiag Part 1

#122 - DebugDiag Part 2

#123 - DebugDiag Part 3

#124 - DebugDiag Part 4

#134 - Microsoft Symbol Proxy (SymProxy) (optional)

#135 - Debugging User Mode Crash Dumps Part 1

#136 - Debugging User Mode Crash Dumps Part 2

#137 - Debugging Kernel Mode Crashes and Hangs

#138 - Debugging - 'dx' Command Part 1

#139 - Debugging - 'dx' Command Part 2

#146 – WinDiff

#147 - Dependency Walker

#154 - Memory Footprint and Leaks

#161 - Troubleshooting a Slow PC (optional)

#167 - Debugging User Mode Crash Dumps Redux

#169 - Debugging Tools for Windows Team (optional)

#172 - Application Hangs

#173 - Troubleshooting with the Windows

#175 - Debugging the Network Stack