Demystifying Debugging and Disassembling Applications

James Lyne
Global Head of Security Research
Sophos & SANS
@JamesLyne

Stephen Sims
Security Researcher
SANS Institute
@Steph3nSims
Part One: Introduction

Disassembly, Disassemblers, and Debuggers
Disassembly and Debugging is used by Application Developers, Security Researchers, Attackers, Malware experts, etc...

- Disassembly allows you to interpret machine code and map it to its mnemonic representation to perform static analysis.
- Debugging allows you to monitor application behavior in a controlled manner, offering the ability to pause, patch, and examine.
- Decompilation goes even further, converting disassembly back to source code.

Expertise in this area can offer new opportunities:
- Security experts who are adept in reverse engineering are highly sought after.
- Exploit sales can quickly yield into the six figures.
Profiting

- **Exploit Sales**
  - Remote browser or document-based exploits can go for >$10K USD
  - Remote Windows Kernel bugs can go for >$100K USD
  - Zerodium paid $1M USD to a group who disclosed a iOS remote jailbreak exploit - [https://www.zerodium.com/ios9.html](https://www.zerodium.com/ios9.html)

- **Bug Bounty Examples:**
  - United Airlines – Will pay up to 1 million award miles for disclosures  
    — [https://www.united.com/web/en-US/content/Contact/bugbounty.aspx](https://www.united.com/web/en-US/content/Contact/bugbounty.aspx)
  - Google – Will pay various amounts depending on the severity of the bug  
    — [https://www.google.com/about/appsecurity/reward-program/](https://www.google.com/about/appsecurity/reward-program/)
  - Microsoft – Will pay up to $100K USD for exploitable bugs and exploit mitigation bypass techniques  
  - CanSecWest Pwn2Own – Annual conference and challenge in Vancouver, CA offering high-priced bounties  
    — [https://www.cansecwest.com/](https://www.cansecwest.com/)
Let’s get right to it! What is disassembly?

![Disassembly Diagram]

Machine Code:
- 10010100111
- 90
- EB 10
- 50
- 6A FF
- 55
- 90
- 90

Disassembler:
- NOP
- JMP SHORT 0x10
- PUSH EAX
- PUSH -1
- PUSH EBP
- NOP
- NOP

Disassembly:
- Black: Instruction
- Red: Operand
Tools for Disassembly

- IDA (Interactive Disassembler)
  - Available from Hex-Rays at [http://www.hex-rays.com](http://www.hex-rays.com)
  - Commercial product with different pricing options
  - Seen as the de facto tool for disassembly

- radare2
  - Available at [http://www.radare.org/](http://www.radare.org/)
  - Free open source reverse engineering framework
  - Offers disassembly, debugging, and many other features

- Many other tools available such as hopper and vivisect
What is Debugging

- Debugging allows us to:
  - Validate and confirm findings made during reverse engineering and static analysis
  - Modify program flow and behavior
  - Set breakpoints at various locations within a program for analysis
  - Determine the exploitability of a potential vulnerability
  - Weaponize and validate the working order of an exploit
  - Learn about application and OS changes made in relation to exploit mitigations

Common Tools for Debugging

- **WinDbg** – An x86/x64 ring0 and ring3 debugger offered by Microsoft at [https://developer.microsoft.com/en-us/windows/hardware/windows-driver-kit](https://developer.microsoft.com/en-us/windows/hardware/windows-driver-kit)


- **OllyDbg** – An x86 debugger for Windows maintained at [http://www.ollydbg.de](http://www.ollydbg.de)

- **GDB** – An open source debugger for UNIX systems available at [https://www.sourceware.org/gdb/](https://www.sourceware.org/gdb/)
Interactive Disassembler (IDA)

- Ilfak Guilfanov – Founder/CEO, Chief Architect, Lead Developer
- Currently maintained by Hex-Rays in Belgium
- [http://www.hex-rays.com](http://www.hex-rays.com)
- Hex-Rays Decompiler also available to convert compiled C & C++ code back to source

**Recommended: The IDA Pro Book**

- *The Unofficial Guide to the World’s Most Popular Disassembler* by Chris Eagle
IDA Basics

- Recursive Descent Disassembler
  - Much more complex and effective approach
  - Can tell instructions from data
  - Handles branches such as jumps and calls
  - Defers branch target instructions based on a condition

- Supports multiple debuggers and techniques, including WinDbg, GDB, Bochs emulator, etc.
  - Disassembles many processor architectures including ARM, x86, AMD, Motorola, etc.
  - Provides many different graphical and structural views of disassembled code
  - Reads symbol libraries
Conditions

● Jump on Zero (JZ) and similar instructions

Green Arrow
Jump

Red Arrow
Don’t Jump

loc_3F3DB:
    mov     rcx, [r14+1C0h]
    and     sil, 0FDh
    mov     byte ptr [rsp+1E8h+var_198], sil
    test    rcx, rcx
    jz      short loc_3F408

loc_3F408:
    xor     r12d, r12d
    mov     edx, r12d
    jmp     short loc_3F40E
Introduction to IDA Demonstration

- Understanding the basic features of IDA
Scripting with IDA

- **IDA Scripting Language (IDC)**
  - Proprietary C-like language to interact with the IDA SDK

- **IDAPython Plugin allowing Python scripting**
  - IDAPython is led by Gergely Erdelyi and available at http://code.google.com/p/idapython/
  - Powerful interface to the IDA SDK and easy to use!

- **IDA Plugins**
  - IDA plugins are compiled C++ programs that perform actions using the IDA APIs and allow you to greatly expand IDA’s capabilities
Using IDAPython to locate banned functions in a program
Part Two: Patch Reversing

Discovering 1-Day Exploits
What are 1-day exploits?

- Many researchers and attackers download security patches as soon as they become available
- Quick bug discovery can lead to exploit development and a large return on investment

Microsoft started “Patch Tuesday” in the early 2000’s

- It serves as a way for Windows administrators to prepare for patching
- ...but, Microsoft seems to be heading towards mandatory updates
- Cumulative patches began in October, 2016
- Windows as a Service (WaaS) for Windows 10
In the past...

- To obtain security patches you could simply visit Microsoft TechNet
  - Choose the desired patch, download, and grab the prior update

---

**Microsoft Security Bulletin MS16-106 - Critical**

**Security Update for Microsoft Graphics Component (3185848)**

Published: September 13, 2016

Version: 1.0

**Executive Summary**

This security update resolves vulnerabilities in Microsoft Windows. The most severe of the vulnerabilities could allow remote code execution if a user either visits a specially crafted website or opens a specially crafted document. Users whose accounts are configured to have fewer user rights on the system could be less impacted than users who operate with administrative user rights.

**Microsoft Windows**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows 10 for 32-bit Systems [2] (3185611)</td>
<td>Important Elevation of Privilege</td>
<td>Important Elevation of Privilege</td>
<td>Important Information Disclosure</td>
<td>Important Elevation of Privilege</td>
<td>Not applicable</td>
<td>3176492</td>
</tr>
</tbody>
</table>

---

Download the new patch here

Download the old patch here
Cumulative Updates

- Now patches are rolled up for the entire year and the files are very large
Making sense of the way Microsoft is forcing cumulative updates can be a challenge.

Greg Linares (@Laughing_Mantis) wrote tools to help make sense of the cumulative patches:
- **PatchExtract** extracts all updates from a cumulative update and organizes them.
- **PatchClean** moves any file older than a month into a subdirectory to allow for focus on recently changed files.
- Mapping updated files to their Knowledge Base (KB) number is still a manual process.
PatchExtract Demonstration

- Extracting the patches from the February, 2017 Patch Tuesday Update
Reversing Patches to Find Vulnerabilities

- Reversing patches can help a research in various ways
  - Quick discovery and weaponization of the patched vulnerability to a driver or DLL can be extremely lucrative as many organizations fail to patch in a timely manner
  - An understanding of how fixes are made to vulnerabilities can help with 0-day vulnerability discovery

- Using tools available to identify changes between two versions of the same file can greatly decrease analysis time

- Many tools are available to help
The following is a list of well-known binary diffing tools:

- **Zynamics/Google’s BinDiff**: Free as of March 18, 2016!
- **Core Security’s turbodiff**: Free
- **DarunGrim 4 by Jeongwook Oh**: Free
- **patchdiff2 by Nicolas Pouvesle**: Free
- **Diaphora** by Joxean Koret
- There are more

Each use different techniques and heuristics to identify changes
Reverse engineering a Microsoft security update to locate a vulnerability
Part Three: Ring 0 Debugging

Debugging Drivers and the Windows Kernel
The Windows Kernel

- The modern Windows Kernel is very complex, requires intermediate to advanced debugging experience, and preferably low level programming experience.

- The majority of the native services and underlying functionality is undocumented:
  - You know you’re onto something when you Google a symbol and get 0 hits!
  - Much of the Kernel is documented on the underground.

- Most operating systems have a two-ring processor access mode architecture: Ring 0 (Kernel) and Ring 3 (User).
To gain proficiency of the Windows Kernel and OS internals, you would need to spend countless months studying and reversing.

A quick list of some key items you’d need to ramp up quickly on:

- Kernel Executive, SRM, Subsystems, System Calls, Kernel Objects
- Kernel Structures such as EPROCESS, KPROCESS, ETHREAD, KTHREAD, TLS, KPRCB, KPCR
- The Hardware Abstraction Layer (HAL)
- Mutexes and SpinLocks
- Driver behavior (IOCTL, IRP, Bus)
Methods for Windows Kernel Debugging

- VirtualKD by SysProgs – A powerful tool to improve and simplify kernel debugging on Windows
  - Available at http://virtualkd.sysprogs.org/

- Serial ports through virtualization applications such as VMware

- Cable-based kernel debugging
  - Ethernet, Null modem, IEEE 1394 (FireWire), USB

- Local Debugging
Attacking the Kernel

Physical Hardware

Hardware Abstraction Layer (HAL)

Kernel-Land Drivers  Kernel

Windows Executive Layer

Less Exploit Mitigation Controls Historically

ntdll.dll

kernelbase.dll  gdi32.dll  user32.dll

kernel32.dll  comctl32.dll

Windows API | DLLs

System Processes (LSASS, etc.)  GDI  User-Mode Services  User-Mode Applications

Application Opt-In Controls

OS Controls

Compile-Time Controls
Exploit Mitigations

- Historically, user mode has seen more advances in exploit mitigations; however, the Kernel has become much more hardened.

- Examples of modern mitigations:
  - Control Flow Guard (CFG)
    - Aimed at stopping Return Oriented Programming (ROP)
  - Browser Specific Controls: MemGC and Isolated Heaps
    - Aimed at stopping Use After Free (UAF) exploitation
  - Kernel Specific Controls: Guard Pages, Kernel Pool Cookies, Null Ptr Deref Prot
  - Proposed Mitigations: Shadow Stacks and Control Flow Integrity (CFI)
  - Oldies but Goodies: ASLR, DEP, Canaries, Safe Unlink, LFH, EMET**

**Microsoft to retire in 2018 ☝️
Kernel Debugging Demonstration

- Connecting to the Windows Kernel using VMware and VirtualKD
When returning to work:

- Audit the patch management program in your organization and ensure critical patches are quickly and safely applied
  - This applies to non-Microsoft products as well, which are often more difficult to manage in relation to patch awareness

- Identify members of your security staff who have skills in reverse engineering and debugging
  - Understand how they are currently using these skills and look for opportunities for improvement

- Inform others as to the risks of delaying security updates
  - Demonstrations can greatly help to gain support
Questions?

Thank You!

James Lyne
@JamesLyne

Stephen Sims
@Steph3nSims