

Advanced Windows Exploitation

Morten Schenk
Alexandru Uifalvi



Copyright © 2024 OffSec Services Limited

All rights reserved. No part of this publication, in whole or in part, may be reproduced, copied, transferred or any other right reserved to its copyright owner, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, any broadcast for distant learning, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission from the author.

Table of Contents

1	Introduction.....	8
2	Microsoft Edge Type Confusion	9
2.1	Exploitation Introduction.....	9
2.1.1	64-bit Architecture	9
2.1.2	Vulnerability Classes.....	12
2.1.3	Basic Security Mitigations	13
2.2	Edge Internals	17
2.2.1	JavaScript Engine.....	18
2.2.2	Chakra Internals.....	19
2.2.3	JIT and Type Confusion	23
2.3	Type Confusion Case Study.....	24
2.3.1	Triggering the Vulnerability.....	25
2.3.2	Root Cause Analysis	26
2.4	Exploiting Type Confusion.....	31
2.4.1	Controlling the auxSlots Pointer	31
2.4.2	Abuse AuxSlots Pointer	34
2.4.3	Create Read and Write Primitive.....	39
2.5	Going for RIP.....	43
2.5.1	Vanilla Attack.....	44
2.5.2	CFG Internals.....	45
2.6	CFG Bypass.....	49
2.6.1	Return Address Overwrite.....	49
2.6.2	Intel CET	53
2.6.3	Out-of-Context Calls.....	54
2.7	Data Only Attack.....	57
2.7.1	Parallel DLL Loading	57
2.7.2	Injecting Fake Work.....	58
2.7.3	Faking the Work.....	63
2.7.4	Hot Patching DLLs	69
2.8	Arbitrary Code Guard (ACG)	72
2.8.1	ACG Theory.....	72
2.8.2	ACG Bypasses.....	74
2.9	Advanced Out-of-Context Calls	75
2.9.1	Faking it to Make it	75

2.9.2	Fixing the Crash	83
2.10	Remote Procedure Calls	88
2.10.1	RPC Theory	88
2.10.2	Is That My Structure	91
2.10.3	Analyzing the Buffers	95
2.10.4	Calling an API	106
2.10.5	Return of Mitigations	111
2.11	Perfecting Out-of-Context Calls	116
2.11.1	Come Back to JavaScript	116
2.11.2	Return Value Alignment	119
2.11.3	Call Me Again	125
2.12	Combining the Work	129
2.12.1	NOP'ing CFG	129
2.12.2	Call Arbitrary API	131
2.13	Browser Sandbox	133
2.13.1	Sandbox Theory Introduction	133
2.13.2	Sandbox Escape Theory	135
2.13.3	The Glue That Binds	136
2.14	Sandbox Escape Practice	139
2.14.1	Insecure Access	139
2.14.2	The Problem of Languages	142
2.15	The Great Escape	143
2.15.1	Activation Factory	143
2.15.2	GetTemplateContent	150
2.15.3	What Is As?	152
2.15.4	Loading the XML	155
2.15.5	Allowing Scripts	159
2.15.6	Pop That Notepad	161
2.15.7	Getting a Shell	163
2.16	Upping The Game - Making the Exploit Version Independent	165
2.16.1	Locating the Base	165
2.16.2	Locating Internal Functions and Imports	167
2.16.3	Locating Exported Functions	171
2.17	Wrapping Up	175
3	Kernel Exploitation and Payloads	176

3.1	The Windows Kernel.....	176
3.1.1	Privilege Levels.....	176
3.1.2	Interrupt Request Level (IRQL).....	177
3.1.3	Windows Kernel Driver Signing.....	179
3.2	Kernel-Mode Debugging on Windows.....	179
3.2.1	Remote Kernel Debugging Over TCP/IP	180
3.3	Communicating with the Kernel.....	183
3.3.1	Native System Calls	184
3.3.2	Device Drivers.....	193
3.4	Kernel Vulnerability Classes.....	209
3.5	Kernel-Mode Payloads	211
3.5.1	Token Stealing.....	212
3.5.2	ALC Editing	217
3.5.3	Kernel Mode Rootkits	221
3.6	Vulnerability Overview and Exploitation.....	222
3.6.1	Triggering the Vulnerability.....	222
3.6.2	Redirecting Execution to Usermode	234
3.7	ROP-Based Attack.....	236
3.7.1	Stack Pivoting.....	236
3.7.2	Kernel Read/Write Primitive	242
3.7.3	Restoring the I/O Ring Object.....	254
3.8	Elevate Privileges	255
3.8.1	Data Only Attack	255
3.9	Developing a Rootkit.....	258
3.9.1	Bypassing DSE.....	258
3.9.2	Elevating Permissions	263
3.9.3	Evading Detection.....	272
3.10	Version Independence.....	276
3.10.1	Dynamic Gadget Location	276
3.11	Extra Mile Exercise.....	278
3.12	Wrapping Up	278
4	Untrusted Pointer Dereference.....	280
4.1	Vulnerability Overview and Exploit Types.....	280
4.1.1	Identifying the Vulnerability through Patch-Diffing	281
4.2	Introduction to Memory Paging and Structures.....	291

4.2.2	Memory Descriptor Lists (MDLs)	299
4.2.3	The PML4 Self-Reference Entry	301
4.2.4	PML4 Self-Reference Entry Randomization	306
4.3	Virtualization-Based Security	307
4.3.1	Hyper-V - The Windows Hypervisor	308
4.3.2	Windows Hypervisor Debugging	312
4.4	Interacting With the Device Driver	317
4.5	Extra Mile Exercise	339
4.6	Reaching the Vulnerable Code Block	339
4.6.2	Joy: From Happiness to Insight	348
4.6.3	A Wild Blue Screen Appears	394
4.6.4	Contentment: Unveiling Inner Peace	407
4.6.5	Uncertainty: Navigating the Unknown	435
4.6.6	Doubt: Understanding Self-Doubt	442
4.6.7	Fear: Facing Our Deepest Anxieties	462
4.7	Despair: The Path to Hope	474
4.8	Mapping Physical Memory to User-Mode	530
4.9	Exploiting the Vulnerability	556
4.10	Wrapping Up	566
5	Unsanitized User-mode Callback	567
5.1	Windows Desktop Applications	567
5.1.1	Windows Kernel Pool Memory	567
5.1.2	Creating Windows Desktop Applications	580
5.1.3	Reversing the TagWND Object	591
5.1.4	Kernel User-mode Callbacks	598
5.1.5	Leaking pWND User-Mode Objects	611
5.2	Triggering the Vulnerability	619
5.2.1	Spraying the Desktop Heap	620
5.2.2	Hooking the Callback	625
5.2.3	Arbitrary WndExtra Overwrite	628
5.3	TagWND Write Primitive	638
5.3.1	Overwrite pWND[0].cbWndExtra	639
5.3.2	Overwrite pWND[1].WndExtra	647
5.4	TagWND Leak and Read Primitive	653
5.4.1	Changing pWND[1].dwStyle	654

5.4.2	Setting The TagWND[1].spmenu	657
5.4.3	Creating a fake TagWND[1].spmenu	664
5.4.4	GetMenuBarInfo Read Primitive	686
5.5	Privilege Escalation.....	688
5.5.1	Low integrity	688
5.5.2	Data Only Attack	692
5.5.3	Restoring The Execution Flow	701
5.6	Executing Code in Kernel-Mode	704
5.6.1	Leaking Nt and Win32k Base	706
5.6.2	NOP-ing kCFG	711
5.6.3	Hijacking a Kernel-Mode Routine	716
5.6.4	Wrapping Up.....	721