Windows Kernel Exploitation: Becoming an Advanced Exploit Developer

1 Overview

This class is meant to show the approach an exploit developer or bug hunter should take in attacking a previously unknown component in the Windows kernel. The training is primarily focused around labs to teach the students what it takes to exploit a real-world vulnerability.

This class focuses on exploiting CVE-2018-8611 on Windows 10 x64 1809 (RS5), a complex race condition that leads to a use-after-free on the non-paged kernel pool. The vulnerability is in the Kernel Transaction Manager (KTM) driver (`tm.sys`), a component that has not received much public scrutiny.

Students will be able to put their new knowledge into practice by exploiting other vulnerabilities in KTM on Windows 11 x64 (CVE-2024-43570 and CVE-2024-43535).

Even though students will learn a lot about the KTM component, we focus on our approach for analyzing this component as a new kernel component that we had no prior knowledge about. The methodology can be reused for any other unknown kernel components a student may encounter in the future. We do not specifically focus on tricks or techniques for bypassing specific Windows versions mitigations. Instead, we teach you the thought process behind exploring functionality to find your own techniques to abuse the bug in ways that allow to build powerful primitives that would facilitate mitigation bypasses.

"Give a (wo)man a mitigation bypass and you feed them for an exploit. Teach a (wo)man to find their own bypasses and you feed them for a lifetime."

The tools/VM we provide during this training are generic and can be reused after the class to assist exploiting other Windows kernel vulnerabilities.

1.1 Key learning objectives

- Setup an efficient Windows kernel debugging environment
- Modern reverse engineering and binary patch diffing
- How to approach exploiting a vulnerability on a previously unknown target
- Step-by-step real-world Windows kernel exploitation

1.2 Prerequisite knowledge

- Comfortable with x86/x64 assembly and reversing it
- C knowledge (reading/writing)
- Comfortable with disassemblers/decompilers (IDA, Ghidra, etc) and debuggers (WinDbg, x64dbg, gdb, etc)
- Familiarity with memory corruption exploitation on any OS
- Windows kernel internals basic knowledge

1.3 Who should attend

- Want to become an exploit developer or bug hunter
- Reverse engineers
- Penetration testers
- Red teamers

1.4 Hardware/Software requirements

- Base OS: Windows recommended
- Hyper-V/VMware virtualisation software
- At least 80GB of free disk space
- At least 8GB of RAM

2 Course outline

2.1 Part 1: Debug environment

- Hyper-V/VMWare
- WinDbg
- Ghidra/IDA
- ret-sync
- Visual Studio
- **Lab: Debug environment setup**

2.2 Part 2: Binary diffing Microsoft updates

- Efficient use of the IDA/Ghidra decompiler to analyze the root cause
- **Lab: Basic binary diffing**

2.3 Part 3: Kernel Transaction Manager (KTM) basics

- KTM objects and APIs
- KTM internals
- Use of public tools for finding data
- **Lab: KTM experimentation**

2.4 Part 4: Understanding CVE-2018-8611

- Root cause vs effect
- Planning exploitation strategy
- **Lab: Better binary diffing**
- **Lab: Reaching vulnerable code**
- **Lab: Triggering CVE-2018-8611 in a debugger**

2.5 Part 5: Exploitation techniques

- Bypassing mitigations
- Windows non-paged pool manipulation
- **Lab: Bad vs good feng shui**
- **Lab: Getting controlled UAF in a debugger**

2.6 Part 6: More exploitation techniques

- Winning the race without a debugger
- Exploitation strategies
- **Lab: Debugging tricks and race win detection**
- **Lab: Discovering a kernel leak**
- **Lab: Restoring cleaned execution**

2.7 Part 7: How to escalate privileges

- Write primitive and privilege escalation strategy
- **Lab: Arbitrary read and write primitives with write 0 and PreviousMode primitive**
- **Lab: Privilege escalation**
- Arbitrary increment primitive and PreviousMode limitations
- **Lab: Arbitrary read and write primitives with increment primitive**

3 Comments

I love teaching vulnerability research and exploit development. The course I teach is all about improving your mental model.

Even the most experienced security researchers don't know everything.

Making vulnerability research and exploit development accessible is key in our evolving and challenging world.

"EZSecLab": Vulnerability research and exploit development made easy.

Checkout some of my previous students' comments on the course:

"The course is absolutely stellar and gave me the confidence to do independent security research."

"This was probably the best training I took (so far)."

"Showing his approach was invaluable to me."

4 Why attend an in-person training?

"There is enormous value in attending classes in person even if the full course is available online. The most important benefit is the ability to ask questions […] and be able to learn the materials much faster."

5 Why focusing on a particular CVE?

What matters is the methodology, not any particular vulnerability. CVE-2018-8611 might not be the latest vulnerability impacting Windows but it is a very informative vulnerability from a learning perspective. It is due to all the challenges we need to solve and all the thought processes involved in exploiting it. Also, we will see that there is not much difference between exploiting CVE-2018-8611 on Windows 10 and exploiting CVE-2024-43570 or CVE-2024-43535 on Windows 11.