Detecting and mitigating elevation-of-privilege exploit for CVE-2017-0005 | Microsoft Security Blog

By Microsoft Defender Security Research Team

Published: 2017-03-27 · Archived: 2026-04-05 12:52:19 UTC

On March 14, 2017, Microsoft released security bulletin MS17-013 to address CVE-2017-0005, a vulnerability in the Windows *Win32k* component that could potentially allow elevation of privileges. A report from a trusted partner identified a zero-day exploit for this vulnerability. The exploit targeted older versions of Windows and allowed attackers to elevate process privileges on these platforms.

In this article, we walk through the technical details of the exploit and assess the performance of tactical mitigations in Windows 10 Anniversary Update—released in August, 2016—as well as strategic mitigations like Supervisor Mode Execution Prevention (SMEP) and virtualization-based security (VBS). We also show how upcoming Creators Update enhancements to Windows Defender Advanced Threat Protection (Windows Defender ATP) can detect attacker elevation-of-privilege (EoP) activity, including EoP activities associated with the exploit.

To test how Windows Defender ATP can help your organization detect, investigate, and respond to advanced attacks, **sign up for a free trial**.

Zero-day elevation-of-privilege exploit

Upon review of its code, we found that this zero-day EoP exploit targets computers running Windows 7 and Windows 8. The exploit has been created so that it avoids executing on newer platforms.

The exploit package unfolds in four stages:

Stages 1 and 2 – Decryptor and API resolver

To protect the main exploit code, attackers have encrypted the initial stage PE file using AES-256 algorithm. To load code for the next stage, a password must be passed as a parameter to the main entry function. Using the *CryptHashData* API, the password is used as a key to decrypt the loader for the next stage.

Stage 2 acts as an intermediate stage where API resolution is performed. API resolution routines in this stage resemble how shellcode or position-independent code works.

The following code shows part of the *GetProcAddress* API resolution routine. This code appears to obfuscate the succeeding payload and stifle analysis.

Stage 3 – Avoiding newer platforms

In stage 3, the exploit package performs environmental checks, specifically to identify the operating system platform and version number. The attacker ensures that the exploit code runs on vulnerable systems that have

fewer built-in mitigations, particularly Windows 7 and Windows 8 devices.

Analysis of the exploit code reveals targeting of systems running specific versions of Windows:

- Major release version 5
- Major release version 6 and minor version 0, 1, or 2

These versions map to Windows operating systems between Windows 2000 and Windows 8, notably excluding Windows 8.1 and Windows 10. Also, upon examination of its architecture-checking routine, we find that the exploit code targets 64-bit systems.

The next stage payload is loaded through DLL reflection.

Stage 4 – Exploit routine

After the environmental checks, the attacker code begins actual exploit of the Windows kernel vulnerability CVE-2017-0005, resulting in arbitrary memory corruption and privileged code execution.

PALETTE.pfnGetNearestFromPalentry corruption

Code execution in the kernel space is made possible by a corrupted pointer in the *PALETTE.pfnGetNearestFromPalentry* function. Microsoft security researchers have been closely tracking this exploitation technique, which is designed to execute code in the kernel courtesy of a malformed *PALETTE* object. Observed in an unrelated sample used during the *Duqu* incident, we have described this relatively old exploit technique in a Virus Bulletin 2015 presentation.

The exploit code calls the native API *NtGdiEngBitBlt* to trigger an *win32k!XLATEOBJ_iXlate* function call that uses the corrupted handler. This passes the control flow to a previously allocated shellcode. As a comparison, the exploit code in the *Duqu* 2.0 case used a *GetNearestPaletteIndex* call from *Gdi32.dll* to pass execution to the corrupt callback handler. This difference clearly indicates that these two exploits are unrelated, despite similarities in their code—similarities that can be attributed to the fact that these exploitation techniques are well-documented.

The exploit uses dynamically constructed *syscall* code snippets to call native Windows APIs.

Once the shellcode is executed, the exploit uses a common token-swapping technique to obtain elevated, SYSTEM privileges for the current process. This technique is often observed in similar EoP exploits.

Mitigation and detection

As previously mentioned, this zero-day exploit does not target modern systems like Windows 10. If environmental checks in the exploit code are bypassed and it is forced to execute on such systems, our tests indicate that the exploit would be unable to completely execute, mitigated by additional layers of defenses. Let's look at both the tactical mitigations—medium-term mitigations designed to break exploitation techniques—as well as the strategic mitigations—durable, long-term mitigations designed to eliminate entire classes of vulnerabilities—that stop the exploit.

Tactical mitigation – prevention of *pfnGetNearestFromPalentry* abuse

The use of *PALETTE.pfnGetNearestFromPalentry* as a control transfer point has been tracked by Microsoft security researchers for quite some time. In fact, this method is on the list tactical mitigations we have been pursuing. In August 2016, with the Windows 10 Anniversary Update, Microsoft released tactical mitigation designed to prevent the abuse of *pfnGetNearestFromPalentry*. The mitigation checks the validity of *PALETTE* function pointers when they are called, ensuring that only a predefined set of functions are called and preventing any abuse of the structure.

Strategic mitigations

Other than the described tactical mitigation, this exploit could also be stopped in Windows 10 by SMEP, ASLR improvements in Windows kernel 64-bit, and virtualization-based security (VBS).

Supervisor Mode Execution Prevention (SMEP)

SMEP is a strategic mitigation feature supported by newer Intel CPUs and adopted since Windows 8.

With SMEP, bits in the page table entry (PTE) serve as *User/Supervisor (U/S)* flags that designate the page to be either in user mode or kernel mode. If a user-mode page is called from kernel-mode code, SMEP generates an access violation and the system triggers a bug check that halts code execution and reports a security violation. This mechanism broadly stops attempts at using user-mode allocated executable pages to run shellcode in kernel mode, a common method used by EoP exploits.

Strategic mitigation like SMEP can effectively raise the bar for a large pool of attackers by instantly rendering hundreds of EoP exploits ineffective, including old-school exploitation methods that call user-mode shellcode directly from the kernel, such as the zero-day exploit for CVE-2017-0005.

To check whether a computer supports SMEP, one can use the Coreinfo tool. The tool uses CPUID instructions to show the sets of CPUs and platforms that should support the feature. The following screen shows that the tested CPU supports SMEP. SMEP is supported on Windows 8 and later.

Windows kernel 64-bit ASLR improvements

Although attackers are forced to work harder to create more sophisticated exploits with SMEP, we do know from studies shared in security conferences and documented incidents that there are ways to potentially bypass SMEP mitigation. These bypass mechanisms include the use of kernel ROP gadgets or direct PTE modifications through read-write (RW) primitives. To respond to these foreseeable developments in exploitation techniques, Microsoft has provided Windows kernel 64-bit ASLR improvements with the Windows 10 Anniversary Update and has made SMEP stronger with randomized kernel addresses, mitigating a bypass vector resulting from direct PTE corruption.

Virtualization-based security (VBS)

Virtualization-based security (VBS) enhancements provide another layer of protection against attempts to execute malicious code in the kernel. For example, Device Guard blocks code execution in a non-signed area in kernel memory, including kernel EoP code. Enhancements in Device Guard also protect key MSRs, control registers, and descriptor table registers. Unauthorized modifications of the CR4 control register bitfields, including the SMEP field, are blocked instantly.

Windows Defender ATP detections

With the upcoming Creators Update release, Windows Defender ATP will be able to detect attempts at a SMEP bypass through CR4 register modifications. Windows Defender ATP will monitor the status of the CR4.SMEP bit and will report inconsistencies. In addition to this, Windows Defender ATP will detect token-swapping attempts by monitoring the state of the token field of a process structure.

The following screenshot shows Windows Defender ATP catching exploit code performing the token-swapping technique to elevate privileges.

Conclusion: Resiliency with mitigation and behavioral detection

The zero-day exploit for CVE-2017-0005 shied away from newer systems because it would have simply been stopped and would have only managed to get unnecessary exposure. Attackers are not so much focusing on legacy systems but avoiding security enhancements present in modern hardware and current platforms like Windows 10 Anniversary Update. While patches continue to provide single-point fixes for specific vulnerabilities, this attacker behavior highlights how built-in exploit mitigations like SMEP, the ASLR improvements, and virtualization-based security (VBS) are providing resiliency.

Windows Defender ATP with Creators Update—now available for public preview—extends defenses further by detecting exploit behavior on endpoints. With the upcoming enhancements, Windows Defender ATP could raise alerts so that SecOps personnel are immediately made aware of EoP activity and can respond accordingly. Read our previous post about uncovering cross-process injection to learn more about how Windows Defender ATP detects sophisticated breach activity.

In addition to strengthening generic detection of EoP exploits, Microsoft security researchers are actively gathering threat intelligence and indicators attributable to ZIRCONIUM, the activity group using the CVE-2017-0005 exploit. Comprehensive threat intelligence about activity groups and their attack methods are available to Windows Defender ATP customers.

Windows Defender ATP is built into the core of Windows 10 Enterprise and can be evaluated free of charge.

Matt Oh
Windows Defender ATP Research Team

Talk to us

Questions, concerns, or insights on this story? Join discussions at the Microsoft community and Windows Defender Security Intelligence.

Follow us on Twitter @WDSecurity and Facebook Windows Defender Security Intelligence.
