

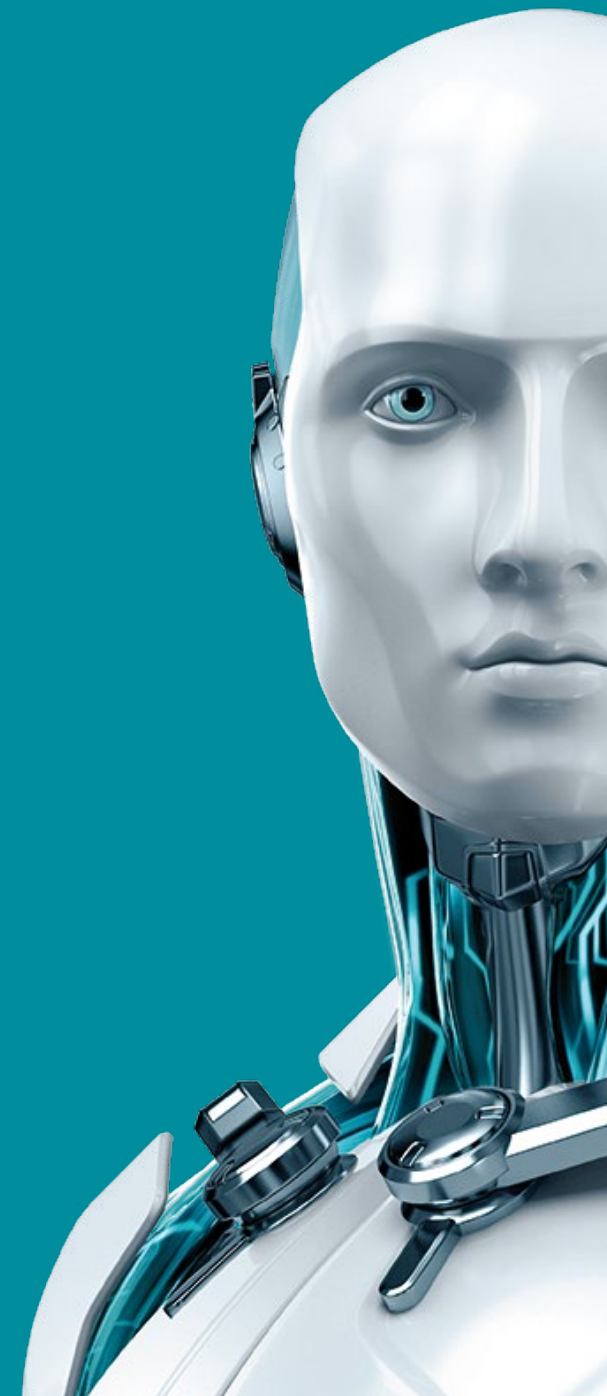


ENJOY SAFER TECHNOLOGY™

Windows exploitation in 2016

ARTEM BARANOV, ESET

Version 2016-DEC-19



Contents

Executive Summary	02
General information	03
Exploitations	07
Mitigations as the best approach for prevention of exploitation	08
A few words about ASLR	12
Web browser security	13
Third-party drivers as a real vector of exploitation	14
Everything updated, everything secured, everything... EMETed	16
Let's talk about firmware security	17
Equation group data breach	21
Conclusion	25

Executive Summary

In this edition of our annual "Windows exploitation" report, we will talk about Windows vulnerabilities and trends in their exploitation. We will show statistics relating to patched vulnerabilities and updates issued as well as additional information about security measures that have been introduced by Microsoft in Windows 10. A separate section is devoted to Edge and Google Chrome browser security; moreover, we'll discuss PC firmware security issues. Of course, this version of the report contains information about the latest version of Microsoft's Enhanced Mitigation Experience Toolkit (EMET).

General information

First of all, let's look at vulnerabilities in the web browsers Internet Explorer and Edge that have been fixed over the past 12 months. Vulnerabilities shown in red in *Table 1* are known to have been exploited in the wild.

Component	Bulletin	Type	Vulnerability
Internet Explorer	MS16-001, MS16-009, MS16-023, MS16-037, MS16-051, MS16-063, MS16-084, MS16-095, MS16-104, MS16-118, MS16-142, MS16-144	Remote Code Execution(12)	CVE-2016-0002, CVE-2016-0005, CVE-2016-0041, CVE-2016-0059, CVE-2016-0060, CVE-2016-0061, CVE-2016-0062, CVE-2016-0063, CVE-2016-0064, CVE-2016-0067, CVE-2016-0068, CVE-2016-0069, CVE-2016-0071, CVE-2016-0072, CVE-2016-0077, CVE-2016-0102, CVE-2016-0103, CVE-2016-0104, CVE-2016-0105, CVE-2016-0106, CVE-2016-0107, CVE-2016-0108, CVE-2016-0109, CVE-2016-0110, CVE-2016-0111, CVE-2016-0112, CVE-2016-0113, CVE-2016-0114, CVE-2016-0154, CVE-2016-0159, CVE-2016-0160, CVE-2016-0162, CVE-2016-0164, CVE-2016-0166, CVE-2016-0187, CVE-2016-0188, CVE-2016-0189 , CVE-2016-0192, CVE-2016-0194, CVE-2016-0199, CVE-2016-0200, CVE-2016-3202, CVE-2016-3205, CVE-2016-3206, CVE-2016-3207, CVE-2016-3210, CVE-2016-3211, CVE-2016-3212, CVE-2016-3213, CVE-2016-3204, CVE-2016-3240, CVE-2016-3241, CVE-2016-3242, CVE-2016-3243, CVE-2016-3245, CVE-2016-3248, CVE-2016-3259, CVE-2016-3260, CVE-2016-3261, CVE-2016-3264, CVE-2016-3273, CVE-2016-3274, CVE-2016-3276, CVE-2016-3277, CVE-2016-3288, CVE-2016-3289, CVE-2016-3290, CVE-2016-3293, CVE-2016-3321, CVE-2016-3322, CVE-2016-3326, CVE-2016-3327, CVE-2016-3329, CVE-2016-3247, CVE-2016-3291, CVE-2016-3292, CVE-2016-3295, CVE-2016-3297, CVE-2016-3324, CVE-2016-3325, CVE-2016-3351 , CVE-2016-3353, CVE-2016-3375, CVE-2016-3267, CVE-2016-3298 , CVE-2016-3331, CVE-2016-3382, CVE-2016-3383, CVE-2016-3384, CVE-2016-3385, CVE-2016-3387, CVE-2016-3388, CVE-2016-3390, CVE-2016-3391, CVE-2016-7239, CVE-2016-7227, CVE-2016-7198, CVE-2016-7199, CVE-2016-7195, CVE-2016-7196, CVE-2016-7241, CVE-2016-7202, CVE-2016-7278, CVE-2016-7279, CVE-2016-7281, CVE-2016-7282, CVE-2016-7283, CVE-2016-7284, CVE-2016-7287
edge	MS16-002, MS16-011, MS16-024, MS16-038, MS16-052, MS16-068, MS16-085, MS16-096, MS16-105, MS16-119, MS16-129, MS16-145	Remote Code Execution(12)	CVE-2016-0003, CVE-2016-0024, CVE-2016-0060, CVE-2016-0061, CVE-2016-0062, CVE-2016-0077, CVE-2016-0080, CVE-2016-0084, CVE-2016-0102, CVE-2016-0105, CVE-2016-0109, CVE-2016-0110, CVE-2016-0111, CVE-2016-0116, CVE-2016-0123, CVE-2016-0124, CVE-2016-0125, CVE-2016-0129, CVE-2016-0130, CVE-2016-0154, CVE-2016-0155, CVE-2016-0156, CVE-2016-0157, CVE-2016-0158, CVE-2016-0161, CVE-2016-0186, CVE-2016-0191, CVE-2016-0192, CVE-2016-0193, CVE-2016-3198, CVE-2016-3199, CVE-2016-3201, CVE-2016-3202, CVE-2016-3203, CVE-2016-3214, CVE-2016-3215, CVE-2016-3222, CVE-2016-3244, CVE-2016-3246, CVE-2016-3248, CVE-2016-3259, CVE-2016-3260, CVE-2016-3264, CVE-2016-3265, CVE-2016-3269, CVE-2016-3271, CVE-2016-3273, CVE-2016-3274, CVE-2016-3276, CVE-2016-3277, CVE-2016-3289, CVE-2016-3293, CVE-2016-3296, CVE-2016-3319, CVE-2016-3322, CVE-2016-3326, CVE-2016-3327, CVE-2016-3329, CVE-2016-3247, CVE-2016-3291, CVE-2016-3294, CVE-2016-3295, CVE-2016-3297, CVE-2016-3325, CVE-2016-3330, CVE-2016-3350, CVE-2016-3351, CVE-2016-3370, CVE-2016-3374, CVE-2016-3377, CVE-2016-3267, CVE-2016-3331, CVE-2016-3382, CVE-2016-3386, CVE-2016-3387, CVE-2016-3388, CVE-2016-3389, CVE-2016-3390, CVE-2016-3391, CVE-2016-3392, CVE-2016-7189, CVE-2016-7190, CVE-2016-7194, CVE-2016-7195, CVE-2016-7196, CVE-2016-7198, CVE-2016-7199, CVE-2016-7200, CVE-2016-7201, CVE-2016-7202, CVE-2016-7203, CVE-2016-7204, CVE-2016-7208, CVE-2016-7209, CVE-2016-7227, CVE-2016-7239, CVE-2016-7240, CVE-2016-7241, CVE-2016-7242, CVE-2016-7243, CVE-2016-7181, CVE-2016-7206, CVE-2016-7279, CVE-2016-7280, CVE-2016-7281, CVE-2016-7282, CVE-2016-7286, CVE-2016-7287, CVE-2016-7288, CVE-2016-7296, CVE-2016-7297

Table 1. Vulnerabilities fixed in Internet Explorer and Edge

Comparing this data with that in previous years, we see the situation is little changed. However, we can see that the number of vulnerabilities in Internet Explorer that were exploited in the wild before patches were available (0-day) has declined. It is worth noting that in the last year no vulnerabilities have been found for the Edge web browser that are known to have been exploited in the wild. From our point of view this situation with Edge was predictable, because, unlike IE11, Edge keeps modern security features turned on by default, including the *AppContainer* full sandbox and *64-bit processes for tabs*.

In "[Windows Exploitation in 2014](#)" we mentioned the EMET feature *Attack Surface Reduction (ASR)*; this was introduced by Microsoft for removing a range of interrelated vulnerabilities. ASR solves exploitation problems by disabling the use of specific vulnerable components in selected processes. Microsoft has adopted this technique from EMET and included it in Windows as a means of prohibiting the use of specific, known-vulnerable components in a system. For example, update [KB3161102](#) removes the *Windows Journal* component. Windows Journal has been patched regularly to remedy vulnerabilities as serious as remote code execution (RCE) that could be exploited with the help of specially crafted Journal (.JNT) files. In the table below you can see the vulnerabilities fixed for Windows user-mode components (UMC).

Component	Bulletin	Type	Vulnerability
Windows UMC (VBScript, JScript, gdi32.dll, Silverlight, Advapi32.dll, Qedit.dll, Ksuser.dll, Aepic.dll, Ntdll.dll, Csrsvr.dll, Session manager/ Smss.exe, Gicndfilter.dll, Windows Journal/ Jnwdrv.dll, Kernelbase.dll, Wow64.dll, Kerberos.dll, Rdpcorets.dll, Wab32.dll, Atmfd.dll, Mfds.dll, Oleaut32.dll, Rpcrt4.dll, Csrsvr.dll, Seclogon.dll, Gdiplus.dll, MSXML, ole32.dll, Vmsntfy.dll, CSRSS, Shell/ Windows.ui.dll, Ehshell.dll, DNS Server/ Dns.exe, Lsasrv.dll, Wdigest.dll, Mswsock.dll, Winhttp.dll, Ntdsai.dll, Ntprint.dll)	MS16-003, MS16-005, MS16-006, MS16-007, MS16-008, MS16-012, MS16-013, MS16-014, MS16-017, MS16-025, MS16-026, MS16-027, MS16-028, MS16-030, MS16-031, MS16-032, MS16-039, MS16-040, MS16-044, MS16-045, MS16-046, MS16-047, MS16-048, MS16-053, MS16-055, MS16-056, MS16-057, MS16-059, MS16-060, MS16-061, MS16-066, MS16-069, MS16-071, MS16-072, MS16-074, MS16-075, MS16-076, MS16-077, MS16-078, MS16-080, MS16-081, MS16-086, MS16-087, MS16-097, MS16-101, MS16-102, MS16-103, MS16-106, MS16-109, MS16-110, MS16-112, MS16-115, MS16-116, MS16-120, MS16-122, MS16-125, MS16-126, MS16-130, MS16-131, MS16-132, MS16-137, MS16-146, MS16-147, MS16-149	Remote Code Execution(38), Elevation of Privilege(17), Security Feature Bypass (2), Information Disclosure(6), Denial of Service(1)	CVE-2016-0002, CVE-2016-0008, CVE-2016-0034 , CVE-2016-0014, CVE-2016-0015, CVE-2016-0016, CVE-2016-0018, CVE-2016-0019, CVE-2016-0020, CVE-2016-0006, CVE-2016-0007, CVE-2016-0058, CVE-2016-0046, CVE-2016-0038, CVE-2016-0040, CVE-2016-0041, CVE-2016-0042, CVE-2016-0044, CVE-2016-0049, CVE-2016-0036, CVE-2016-0100, CVE-2016-0120, CVE-2016-0121, CVE-2016-0101, CVE-2016-0098, CVE-2016-0117, CVE-2016-0118, CVE-2016-0091, CVE-2016-0092, CVE-2016-0087, CVE-2016-0099, CVE-2016-0145, CVE-2016-0147, CVE-2016-0153, CVE-2016-0088, CVE-2016-0089, CVE-2016-0090, CVE-2016-0135, CVE-2016-0128, CVE-2016-0151, CVE-2016-0187, CVE-2016-0189 , CVE-2016-0168, CVE-2016-0169, CVE-2016-0170, CVE-2016-0184, CVE-2016-0195, CVE-2016-0182, CVE-2016-0179, CVE-2016-0185, CVE-2016-0180, CVE-2016-0178, CVE-2016-0181, CVE-2016-3205, CVE-2016-3206, CVE-2016-3207, CVE-2016-3227, CVE-2016-3223, CVE-2016-3216, CVE-2016-3220, CVE-2016-3225, CVE-2016-3228, CVE-2016-3213, CVE-2016-3236, CVE-2016-3231, CVE-2016-3201, CVE-2016-3203, CVE-2016-3215, CVE-2016-3226, CVE-2016-3204, CVE-2016-3238, CVE-2016-3239, CVE-2016-3301, CVE-2016-3303, CVE-2016-3304, CVE-2016-3300, CVE-2016-3237, CVE-2016-3319, CVE-2016-3312, CVE-2016-3354, CVE-2016-3355, CVE-2016-3356, CVE-2016-3367, CVE-2016-3346, CVE-2016-3352, CVE-2016-3368, CVE-2016-3369, CVE-2016-3302, CVE-2016-3370, CVE-2016-3374, CVE-2016-3375, CVE-2016-3209, CVE-2016-3262, CVE-2016-3263, CVE-2016-3393 , CVE-2016-3396, CVE-2016-7182, CVE-2016-0142, CVE-2016-7188, CVE-2016-3298, CVE-2016-7212, CVE-2016-7221, CVE-2016-7222, CVE-2016-7248, CVE-2016-7205, CVE-2016-7210, CVE-2016-7217, CVE-2016-7256 , CVE-2016-7220, CVE-2016-7237, CVE-2016-7238, CVE-2016-7257, CVE-2016-7272, CVE-2016-7273, CVE-2016-7274, CVE-2016-7292

Table 2. Vulnerabilities fixed in Windows user-mode components

In *Figure 1* the information from Table 1 and Table 2 are presented in an easier-to-read format.

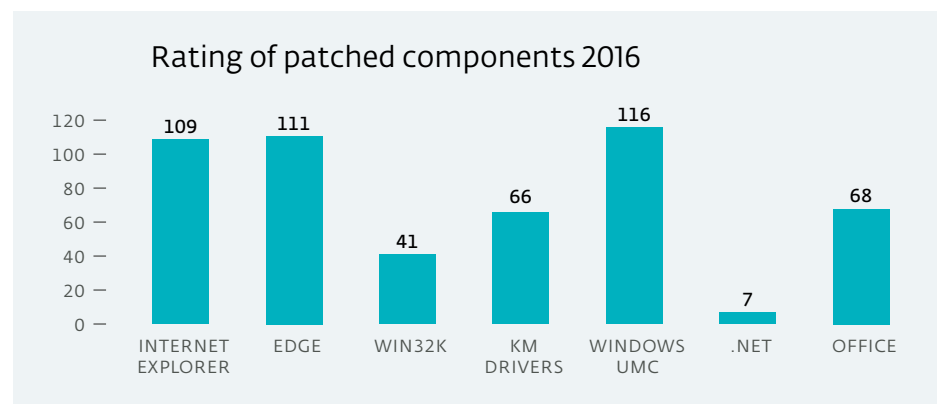


Figure 1. Proportions of patched components 2016

In *Figure 2* are presented statistics about issued security updates and types of attacks that they are intended to fix. As you can see, the largest number of updates, more than 60, were issued for Windows user mode components and most of them are fixed RCE and LPE vulnerabilities. Nor is it a surprise that most updates issued for Win32k.sys and kernel mode (KM) drivers are for fixing LPE vulnerabilities.

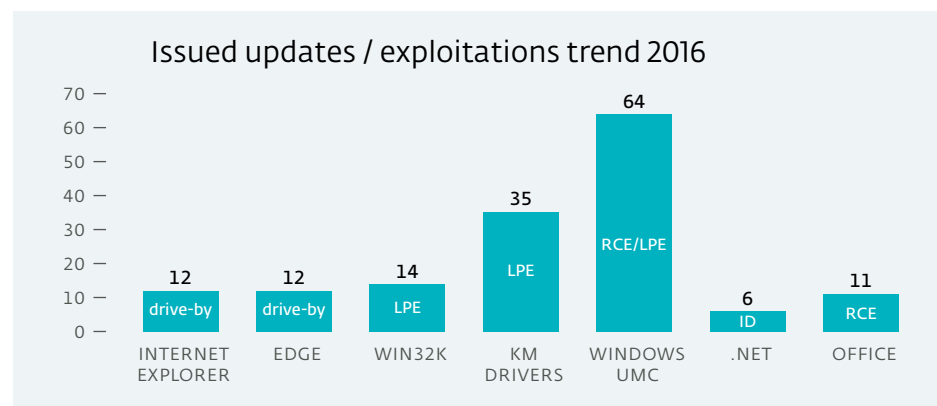


Figure 2. Updates and Exploitation Trends 2016

In May 2016, Microsoft [released](#) Service Pack 2 (SP2) for Windows 7 with the identifier [KB3125574](#) (Convenience rollup update for Windows 7 SP1 and Windows Server 2008 R2 SP1). Despite Microsoft's refusing to describe this update as a Service Pack, that is effectively what it is, since it contains all security and non-security fixes since the release of Windows 7 SP1. This cumulative update is very useful for IT specialists, who can integrate it into a Windows 7 SP1 image (WIM) file in order to deploy up-to-date copies of Windows at their workplaces.

In the figure below, you can see interesting statistics regarding patched vulnerabilities in both 2015 and 2016. There is an obvious trend in that in 2016 more vulnerabilities were fixed than in 2015, in almost all except the web browser Internet Explorer.

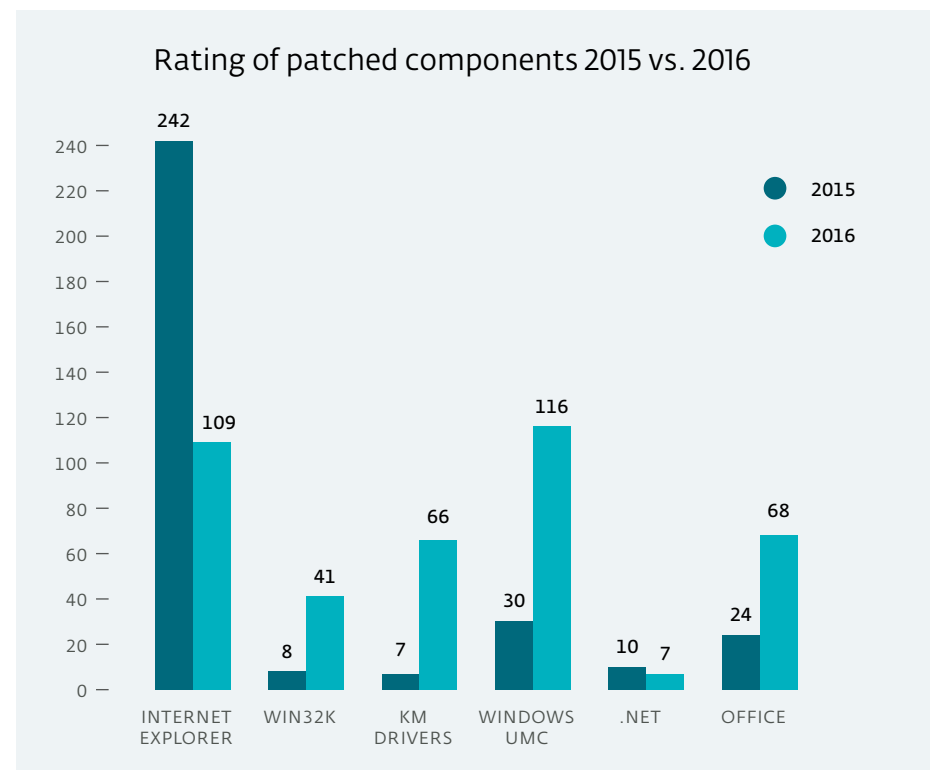


Figure 3. Comparison of components patched 2015-2016

Microsoft also [introduced](#) its cumulative update model for Windows 7 and 8.1. This scheme for distributing updates was originally used in Windows 10 by default and, unlike the previous model, it offers updates as a single (cumulative) monthly package. Previously, when users installed a clean copy of Windows 7 or 8.1, they needed to select many separate updates and download them. This method made the process of updating difficult, especially for system administrators who deal with many fresh installs of Windows. Cumulative updates mean users and IT specialists will update their copies of Windows without being required to take so many actions.

In the past year, Microsoft has released the Windows 10 Anniversary Update, which also brought in the [Linux subsystem](#) for users (Windows Subsystem for Linux, WSL). This means that users can work with the *bash* command interpreter as well as with other standard Linux tools and run their own Linux applications. WSL is implemented with help of two kernel mode drivers – *LXss.sys* and *LXCore.sys* – that are responsible for creating the semantics of Linux services based on the flexible NT kernel and providing support for the Linux VFS.

Component	Bulletin	Type	Vulnerability
Win32k	MS16-005, MS16-018, MS16-034, MS16-039, MS16-062, MS16-073, MS16-074, MS16-090, MS16-098, MS16-106, MS16-120, MS16-123, MS16-135, MS16-151	Remote Code Execution(1), Elevation of Privilege(13)	CVE-2016-0009, CVE-2016-0048, CVE-2016-0093, CVE-2016-0094, CVE-2016-0095, CVE-2016-0096, CVE-2016-0143, CVE-2016-0165 , CVE-2016-0167 , CVE-2016-0171, CVE-2016-0173, CVE-2016-0174, CVE-2016-0175, CVE-2016-0196, CVE-2016-3218, CVE-2016-3221, CVE-2016-3232, CVE-2016-3219, CVE-2016-3249, CVE-2016-3250, CVE-2016-3251, CVE-2016-3252, CVE-2016-3254, CVE-2016-3286, CVE-2016-3308, CVE-2016-3309, CVE-2016-3310, CVE-2016-3311, CVE-2016-3348, CVE-2016-3349, CVE-2016-3270, CVE-2016-3266, CVE-2016-3376, CVE-2016-7185, CVE-2016-7211, CVE-2016-7214, CVE-2016-7215, CVE-2016-7246, CVE-2016-7255 , CVE-2016-7259, CVE-2016-7260
KM drivers (Boot loader/Winload.efi, Winload.exe, Ntoskrnl.exe, Mrxdav.sys, Rdpvideominiport.sys, Usbstor.sys, Vmswitch.sys, Ksecdd, Mrxsmb10.sys, Mrxsmb20.sys, http.sys, Dxgkrnl.sys, Dxgmmms1.sys, Volmgr.sys, Ksecpkg.sys, Srv.sys, Netbt.sys, Cng.sys, Appid.sys, Clfs.sys, Bowser.sys)	MS16-008, MS16-014, MS16-016, MS16-017, MS16-031, MS16-033, MS16-044, MS16-045, MS16-047, MS16-048, MS16-049, MS16-060, MS16-062, MS16-067, MS16-075, MS16-077, MS16-082, MS16-089, MS16-092, MS16-094, MS16-100, MS16-101, MS16-111, MS16-113, MS16-114, MS16-123, MS16-124, MS16-134, MS16-135, MS16-138, MS16-139, MS16-140, MS16-149, MS16-150, MS16-152, MS16-153	Remote Code Execution(4), Elevation of Privilege(19), Denial of Service(2), Information Disclosure(7), Security Feature Bypass (5)	CVE-2016-0006, CVE-2016-0007, CVE-2016-0051, CVE-2016-0036, CVE-2016-0133, CVE-2016-0088, CVE-2016-0089, CVE-2016-0090, CVE-2016-0128, CVE-2016-0150, CVE-2016-0180, CVE-2016-0176, CVE-2016-0197, CVE-2016-0190, CVE-2016-3225, CVE-2016-3213, CVE-2016-3236, CVE-2016-3230, CVE-2016-3256, CVE-2016-3258, CVE-2016-3272, CVE-2016-3287, CVE-2016-0040, CVE-2016-0041, CVE-2016-0042, CVE-2016-0044, CVE-2016-0049, CVE-2016-0087, CVE-2016-0153, CVE-2016-0151, CVE-2016-3320, CVE-2016-3300, CVE-2016-3237, CVE-2016-3305, CVE-2016-3306, CVE-2016-3371, CVE-2016-3372, CVE-2016-3373, CVE-2016-3344, CVE-2016-3345, CVE-2016-3341, CVE-2016-0070, CVE-2016-0073, CVE-2016-0075, CVE-2016-0079, CVE-2016-0026, CVE-2016-3332, CVE-2016-3333, CVE-2016-3334, CVE-2016-3335, CVE-2016-3338, CVE-2016-3340, CVE-2016-3342, CVE-2016-3343, CVE-2016-7184, CVE-2016-7218, CVE-2016-7223, CVE-2016-7224, CVE-2016-7225, CVE-2016-7226, CVE-2016-7216, CVE-2016-7247, CVE-2016-7219, CVE-2016-7271, CVE-2016-7258, CVE-2016-7295
.NET Framework	MS16-019, MS16-035, MS16-041, MS16-065, MS16-091, MS16-155	Denial of Service(1), Security Feature Bypass (1), Remote Code Execution(1), Information Disclosure(3)	CVE-2016-0033, CVE-2016-0047, CVE-2016-0132, CVE-2016-0148, CVE-2016-0149, CVE-2016-3255, CVE-2016-7270

Table 3. Vulnerabilities in the Kernel and .NET Framework

Exploitation

The two most common types of exploit attacks in the Windows world are Remote Code Execution (RCE) and Local Privilege Escalation (LPE). The first is used by attackers to penetrate a system and the second to obtain maximum privileges on that system. In fact, RCE exploits are commonly used to target vulnerabilities in web browsers with the intention of downloading and running malicious executables – such attacks are called drive-by downloads. Once a system is penetrated, attackers need maximum privileges so that their code can get full control over the compromised system. In most cases, exploited LPE vulnerabilities are located in the standard Windows `win32k.sys` driver. If attackers can successfully exploit such a vulnerability in `win32k.sys`, they will get full SYSTEM privileges and the ability to run malicious code in kernel mode (Ring 0). As we will describe below, attackers could then use vulnerabilities in firmware to get "god mode" privileges with the ability to bypass hypervisor security measures and get full control over a system with running VMs.

Vulnerability in-the-wild	ESET detection	Month	Targeted attack*
CVE-2015-8651	SWF/Exploit.CVE-2015-8651	January	Yes
CVE-2016-0034	Win32/Exploit.CVE-2016-0034	February	Yes
CVE-2016-1019	SWF/Exploit.CVE-2016-1019	April	Yes
CVE-2016-0189	Win32/Exploit.CVE-2016-0189	May	Yes
CVE-2016-4117	SWF/Exploit.CVE-2016-4117	May	Yes
CVE-2016-4273	SWF/Exploit.CVE-2016-4273	October	—

Table 4. ESET Detection of selected vulnerabilities in the wild

In the [previous issue](#) of this report we pointed to a Stuxnet-like vulnerability, [CVE-2015-1769](#), present in the Windows Mount Manager subsystem on Windows Vista and later. That vulnerability allowed attackers to execute arbitrary code with system privileges when a specially-crafted removable USB drive was inserted into a PC. In 2016, Microsoft fixed [CVE-2016-0133](#), which was located in the USB mass storage class drivers `Usbstor.sys` and `Tsusbhub.sys`, with the [MS16-033](#) security update. Like CVE-2015-1769, this vulnerability allows attackers to execute arbitrary code with SYSTEM privileges. To exploit it, attackers need physical access to the computer; that is why it was marked as important but not critical.

The vulnerabilities below are examples of Stuxnet-like type.

Vulnerability	Bulletin	Details
CVE-2015-0096	MS15-020	Patch for MS10-046
CVE-2015-1769	MS15-085	Mountmgr LPE
CVE-2016-0133	MS16-033	Usbstor LPE

Table 5. Examples of so-called Stuxnet-like vulnerabilities that allow attackers to execute malicious code from a specially crafted removable drive.

We also have special detection for the Duqu 2 `win32k.sys` LPE exploit (MS15-061), which has been used in targeted attacks: [Win32/Exploit.CVE-2015-2360.A](#).

Mitigations as the best approach for prevention of exploitation

We can see that Microsoft is making serious efforts to improve the security of modern Windows versions incrementally. These security measures are called mitigations and serve to decrease risks of vulnerabilities being exploited. We can also compare this approach with the EMET feature called Attack Surface Reduction (ASR), because mitigations can block a wide range of exploits that use similar techniques.

Starting with Windows 8, Microsoft introduced the special API *SetProcessMitigationPolicy* to turn on mitigations that are supported by the current Windows version. In the *Table 6* below you can see various types of mitigations. Each of these mitigations is intended to block a specific exploitation vector.

Mitigation (<i>SetProcessMitigationPolicy</i>)	Windows 8.1	Windows 10
DEP (<i>ProcessDEPPolicy</i>)	x	x
ASLR (<i>ProcessASLRPolicy</i>)	x	x
Dynamic code prohibited (<i>ProcessDynamicCodePolicy</i>)	x	x
Strict handle checks (<i>ProcessStrictHandleChecksPolicy</i>)	x	x
Win32k system calls disabled (<i>ProcessSystemCallDisablePolicy</i>)	x	x
Extension points disabled (<i>ProcessExtensionPointDisablePolicy</i>)	x	x
Control Flow Guard enabled (<i>ProcessControlFlowGuardPolicy</i>)	x	x
Signatures restricted (<i>ProcessSignaturePolicy</i>)		x
Non-system fonts disabled (<i>ProcessFontDisablePolicy</i>)		x
Loading of remote and low IL images disabled (<i>ProcessImageLoadPolicy</i>)		x

Table 6. List of mitigations that are available for applications to use to improve their own security.

As we can see from the preceding table, Windows 10 introduced three new types of mitigations: *ProcessSignaturePolicy*, *ProcessFontDisablePolicy* and *ProcessImageLoadPolicy*. The first mitigation is used as a security measure to allow only images with a specific type of digital signature to load into a target application. For example, the Edge web browser turns on this mitigation to allow loading in its address space only those images that are signed with a special Windows Store digital certificate.

We already mentioned the *ProcessFontDisablePolicy* mitigation in our "Windows exploitation in 2015" paper as a security option that can be turned on for a specific application by EMET. This option helps applications to be protected from one type of LPE exploit that uses specially crafted font files and loads them from a non-system directory. Thus, this option forbids loading of font files into the process from any locations except %windir%\fonts.

The last new security feature is called *ProcessImageLoadPolicy*; it forbids loading executable images from remote locations into the process address space. It also can forbid loading of executables marked as Low Integrity Level (IL).

Another interesting security feature is used by the IE11 web browser, beginning with Windows 8 Update 3 and with Edge on Windows 10. It is called Control Flow Guard (CFG) and is used to prevent exploitation of several types of vulnerabilities. CFG allows applications to mitigate exploitation vectors that are involved with indirect control flow transfer to exploit code — for example, in the case of use-after-free (UAF) vulnerabilities. Unlike the previously mentioned mitigations, this security measure requires support from both players — the executable application and Windows. CFG is integrated into the application by the [compiler and linker](#); for example, by Visual Studio 2015. From the Windows side, CFG [is implemented](#) by user mode and kernel mode components (the executable loader and Windows kernel). The CFG security mechanism allows Windows to fix calls of functions inside the executable and thus to take control of execution flow that could be modified by exploits that specialize in modification of the table of virtual methods (vtable). CFG is also supported by Adobe Flash Player.

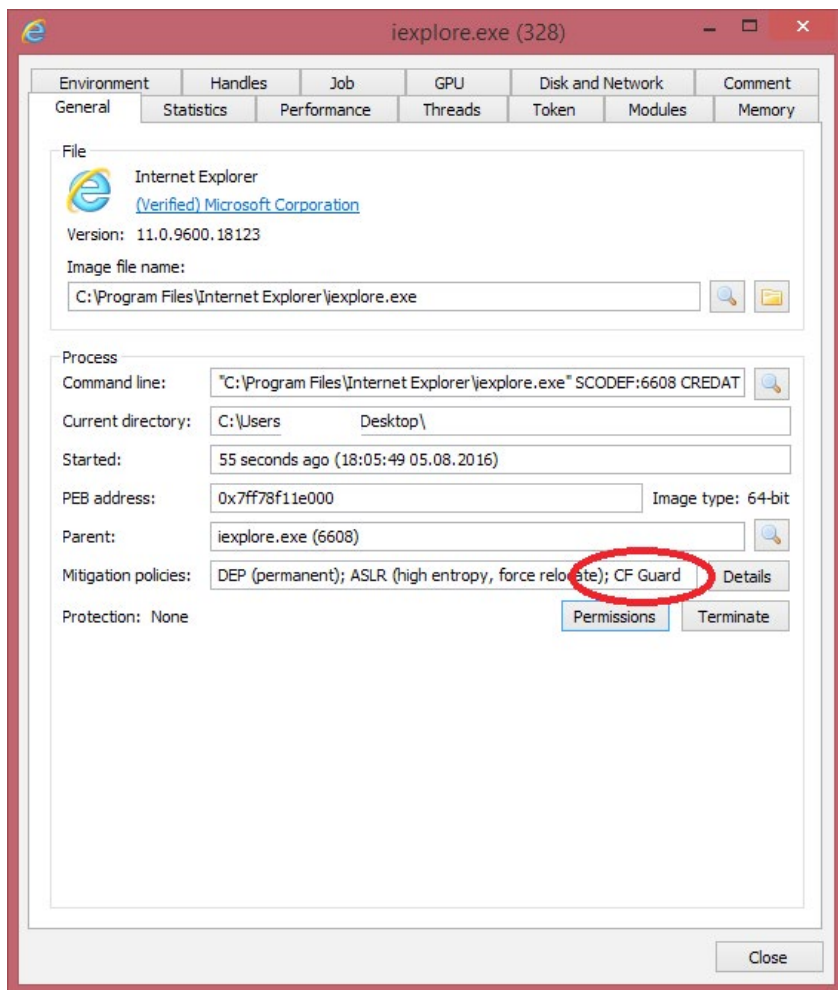


Figure 4. IE 11 on up-to-date Windows 8.1 has CFG support.

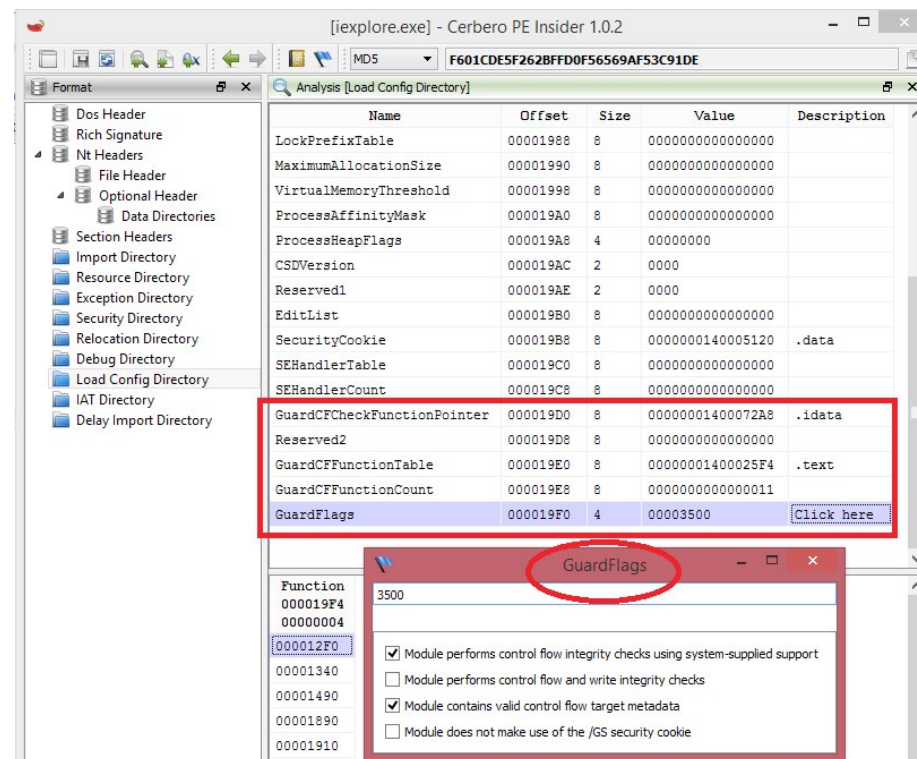


Figure 5. Information about CFG in IE's executable – IMAGE_LOAD_CONFIG_DIRECTORY directory.

It is important for Windows not only to introduce security features for applications, but also to integrate them into built-in system components. *Table 7* shows various important Windows components and security features that are applied to them by default.

Process	Services	Smss	Csrss	Winlogon	Lsass	Explorer
Mitigation						
DEP	X	X	X	X	X	X
HEASLR, force relocate	X	X	X	X	X	ASLR
Dynamic code prohibited	X	X	X			
Strict handle checks	X	X	X	X	X	
Win32k system calls disabled						
Extension points disabled	X					
Control Flow Guard enabled	X	X	X	X	X	X
Signatures restricted	X (MS only)	X (MS only)	X (MS only)			
Non-system fonts disabled						
Loading of remote and low IL images disabled						

Table 7. Mitigations that are applied by default for important processes (Windows 10).

From the table above we can see exactly which of the mitigations mentioned are applied to highly important Windows processes.

- **Services** – Service Control Manager (SCM), which is used for operations with services.
- **Smss** – Session Manager Subsystem Service is the first user mode process that is created by ntoskrnl.
- **Csrss** – Client/Server Runtime Subsystem is a very important process that is responsible for the Windows GUI subsystem implementation in user mode.
- **Winlogon** – a very important Windows component, which is responsible for operations involving logon into a system.
- **Lsass** – Local Security Authority Subsystem Service is another very important process that is used for applying various security policies in a system.

The built-in Windows hypervisor system that is called Hyper-V also demands a new level of security. Because attackers are potentially able to get control under the host OS and all running VMs, Microsoft introduced several important security features for Windows 10. These features rely on the concept of a secure environment that is isolated from all guest VMs as well as from the host OS. It is called *Virtual Secure Mode (VSM)* and is often referred to as *Virtualization Based Security (VBS)*. On the foundation of VSM, Microsoft implemented such already-known security features as *Device Guard*, *Credential Guard* and *Hypervisor Code Integrity (HVCI)*. All the above-mentioned components and the Windows kernel work in a special isolated environment, which cannot be directly accessed from the host OS and running VMs.

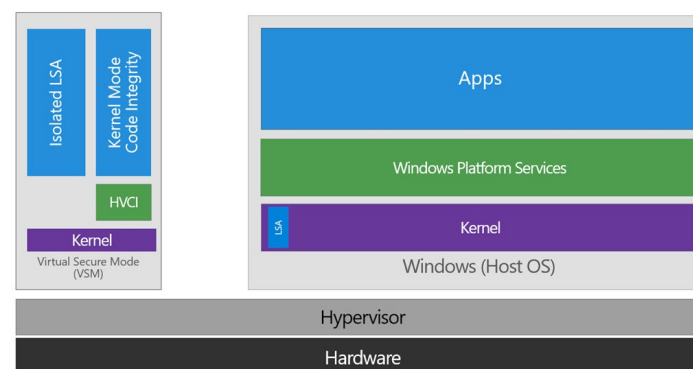


Figure 6. Hyper-V architecture with VSM as it is [described](#) by the Microsoft security guys.

It is important to understand that VSM assumes responsibility for implementing several highly important security functions that can be compromised on VMs. By introducing VSM, Microsoft solves two related tasks: it isolates the execution of security operations from potentially non-trusted environments and provides a strong place for storing sensitive data. For example, Device Guard hardens already existing firmware security features like UEFI Secure Boot with its own additional checks and brings Kernel Mode Code Integrity (KMCI) as well as the Hypervisor Code Integrity (HVCI) subsystem into the VSM. Both KMCI and HVCI subsystems could, potentially, be compromised by attackers in order to turn off checking of digital signatures of kernel mode executables.

Another security operation, which can be potentially isolated in VSM, is a subsystem responsible for working with user credentials data. This feature is called Credential Guard and is based on executing its own VSM-related copy of the Local Security Authority Subsystem Service (LSASS). Thus, all operations with users' credentials from all running VMs are isolated in a secure environment and cannot be compromised by attackers.

According to the famous Windows internals researcher [Alex Ionescu](#), VSM leverages special execution modes for isolated environments. These execution modes are called Secure Kernel Mode (SKM) and Isolated User Mode (IUM). Despite this division, both still use the usual CPU privilege levels, i.e. Ring 0 for SKM and Ring 3 for IUM, but a major difference is that the hypervisor trusts both, unlike kernel mode and user mode code from the host OS and guest VMs.

Note that SKM uses a special "light" version of NT Kernel (`ntoskrnl`) that is called NT Secure Kernel (`securekernel.exe`). The size of `securekernel`, at about 445 KB, is much smaller than original full `ntoskrnl` (7.45 MB). The new kernel has some export functions that are absent in `ntoskrnl`. Such functions have the `Sk` prefix: for example, `SkobCreateHandle`, `SkobCreateObject`, `SkobDereferenceObject`, `SkobReferenceObject`, `SkobReferenceObjectByHandler`. As suggested by the names of these functions, their purpose is to work with kernel objects in the context of the secure kernel.

Microsoft also leverages Hyper-V to make RCE vulnerability exploitation more difficult for attackers. In September 2016 they [introduced](#) a special security measure called *Windows Defender Application Guard for Microsoft Edge*. Like VSM, this feature is based on Hyper-V isolation and intended for running non-trusted content accessed by the Edge web browser in a separate virtual machine. This is the perfect solution for mitigating RCE exploits and drive-by attacks. When the user clicks on a link, any potentially dangerous content is opened into a separate virtual machine that has no access to important user data on the host. Windows Defender Application Guard for Microsoft Edge will become available for all users of Windows 10 Enterprise in 2017.

Bulletin	Vulnerability	Details	Windows
MS16-045	CVE-2016-0088 CVE-2016-0089 CVE-2016-0090	Hyper-V Remote Code Execution Vulnerabilities	Windows 8.1+
MS16-066	CVE-2016-0181	Hypervisor Code Integrity (VSM, HVCI) Security Feature Bypass	Windows 10
MS16-089	CVE-2016-3256	Windows Secure Kernel Mode Information Disclosure Vulnerability	Windows 10
MS16-094	CVE-2016-3287	Secure Boot Security Feature Bypass	Windows 8.1+
MS16-100	CVE-2016-3320	Secure Boot Security Feature Bypass Vulnerability	Windows 8.1+
MS16-113	CVE-2016-3344	Windows Secure Kernel Mode Information Disclosure Vulnerability	Windows 10
MS16-137	CVE-2016-7220	Virtual Secure Mode Information Disclosure Vulnerability	Windows 10
MS16-140	CVE-2016-7247	Secure Boot Security Feature Bypass Vulnerability	Windows 8.1+
MS16-150	CVE-2016-7271	Windows Secure Kernel Mode Elevation of Privilege Vulnerability	Windows 10

Table 8. Fixed vulnerabilities that are related to firmware and hypervisor security.

A few words about ASLR

We previously wrote about ASLR mitigation in our previous report "[Windows exploitation in 2014](#)". To avoid repeating information already published, let's talk about it from a slightly different viewpoint. ASLR was introduced by Microsoft in Windows Vista and improved in subsequent versions of Windows. In the case of older Windows versions, for example for the still popular but no longer supported Windows XP, Microsoft has recommended using EMET for enabling ASLR. This has its limitations, though, since EMET can provide ASLR only for /DYNAMICBASE-linked PE-files and not for Windows system structures or heaps.

Unlike DEP, which supports opt-out working mode and is actually set by default in all modern Windows versions, ASLR was initially developed by Microsoft for running in opt-in mode for executable files (opt-in mode does not refer to randomizing Windows system structures and heaps; it is enabled by default). To close this weak default, Microsoft introduced the so-called Force ASLR option, starting in Windows 7 and in Windows 8 as well with update [KB2639308](#). This option forces the Windows executable loader to apply ASLR to PE-files that were not compiled with the /DYNAMICBASE flag. Setting this flag for an executable in a special registry key will mean ASLR will be applied for all libraries loaded into that process's address space. This option is used by Internet Explorer 10+ to increase its own security level and thus renders impossible exploits that use non-ASLR libraries that would otherwise provide for easier exploitation.

Microsoft has improved ASLR several times since it was introduced in Windows Vista, by randomizing [locations](#) of PE-files, PEB, stack and heap memory blocks. The next ASLR [update](#) brought forced randomization of PE-files as well as randomizing the location of allocated blocks of virtual memory. These security improvements are included in Windows 8, which also introduces so-called High Entropy ASLR (HEASLR). We [described](#) HEASLR in "[Windows exploitation in 2013](#)".

As we can see, Microsoft is making a good effort to make ASLR more secure, but previously we were talking only about ASLR for user mode programs. Since [Windows Vista SP1](#), ASLR also works in kernel mode Ring 0 (KASLR) for drivers built with the /DYNAMICBASE linker flag. KASLR was also enhanced, starting with the major Windows 10 update in August 2016. Now KASLR is [applied](#) to almost all Windows system structures that are located in Ring 0 virtual address space, including page tables and page directory, the key structures of Windows memory manager for linear virtual addresses translation (PFN database, WSL, etc).

Web browser security

The security of web browsers is one of the most important things for user safety. This is explained by the fact that web browsers represent very attractive targets for attackers to execute malicious code remotely. Attackers use specially crafted web pages with RCE exploits to penetrate a system. Vulnerabilities are present in all software and web browsers are no exception. This means that web browsers should have special security mitigations to block malicious actions achievable through potential exploits.

It is also interesting to compare the security features of the most widespread browsers. The *Table 9* shows examples of such information.

Web-browser	MS Internet Explorer 11	Microsoft Edge	Google Chrome	Mozilla Firefox
Mitigation				
Sandbox	AppContainer (EPM)	AppContainer	AppContainer	
DEP	X	X	X	X
HEASLR, force relocate	XX	XX	X	ASLR
Dynamic code prohibited		X		
Strict handle checks	X	X	X	
Win32k system calls disabled			X	
Extension points disabled				
Control Flow Guard enabled	X	X		
Signatures restricted		X		
Non-system fonts disabled				
Loading of remote and low IL images disabled		X	X	

Table 9. Comparison of mitigations in web browsers.

In the last year, the web browsers Google Chrome and Microsoft Edge began to refuse to auto-play Flash content by default. Users of Edge got this feature with the Windows 10 Anniversary [Update](#) in August 2016. Flash content that is not located at the center of a web page will now be paused by Edge automatically. The same situation applies with Google Chrome: it introduced click-to-play Flash content starting with Chrome 53. This security measure is also used for saving battery life.

The Mozilla Foundation also has demonstrated steps on the way to Firefox sandboxing. For example, starting from Mozilla Firefox 48 beta, their web browser was given [support](#) for the important capability of splitting one Firefox process, which was responsible for all GUI operations and opened tabs, into several processes. It is known that, unlike Chrome or Edge, Firefox did not have an option for running an opened tab in a separate process, so all tabs opened by the user ran in one process address space. Now, Firefox will be running in two processes: one that is responsible for GUI drawing, and another that handles the user's tabs. This is a good start to implementing more comprehensive sandboxing in the future.

Microsoft's Edge browser also [got](#) a new security feature that helps to mitigate malicious actions of LPE exploits, which use vulnerabilities in the notorious `win32k.sys` driver. This measure is called *Win32k syscalls filtering* and allows the Windows kernel to disable the calling of specific Win32k system services from the context of the Edge process. By disabling the ability to call some vulnerable services or services that have been used by exploits to corrupt kernel memory and trigger a vulnerability, Edge protects users from a wide range of LPE exploits.

Sandboxing of Google Chrome, MS IE11 and Edge were discussed in detail in previous versions of the "Windows exploitation" report and in the blog post ["Exploit protection for Microsoft Windows"](#).

Third-party drivers as a real vector of exploitation

As we know, Windows' internal world is divided into two major parts: user mode (Ring 3) and kernel mode (Ring 0). Yes, modern Windows versions have three parts, including the hypervisor, but in the context of our chosen theme, we will skip the latter. Unlike "normal" applications that run in user mode, drivers work in kernel mode and have full access to hardware, physical memory, I/O ports, etc. All these resources are maintained by the Windows kernel (`ntoskrnl`) and its various sub-systems.

Compromising Windows kernel mode provides attackers with maximum SYSTEM level privileges for running a host or guest VM. The driver `win32k.sys` has already been used many times by attackers in real attacks to gain SYSTEM privileges in compromised systems. The same problem applies to any other legitimate driver that contains an LPE vulnerability and can be used for privilege elevation.

With the Windows 10 anniversary update, Microsoft [introduced](#) more restricted requirements for kernel mode drivers. Now, in order to work in Windows 10, a driver must be digitally signed by Microsoft, i.e. submitted to and approved by the Windows Hardware Developer Center. This measure should raise the bar of system security and stability, discarding drivers that were developed inappropriately.

Third-party drivers can contain security flaws "by design", because their developers don't integrate important security checks into them. A quite common mistake is that authors neglect to check the context of caller processes during dispatch of `IRP_MJ_CONTROL` driver requests performing critical operations. For example, one version of the anti-cheat driver `Capcom.sys` (DA6CA1FB539F825CA0F012ED-6976BAF57EF9C70143B7A1E88B4650BF7A925E24), which was distributed with Capcom's Street Fighter V computer game, contained a function for disabling the [SMEP](#) (Supervisor Mode Execution Protection) security measure. The rootkit function called `fnDisableSMEPAndCallFunction` allows the caller process to execute user mode code from kernel mode -- that is disabled by SMEP starting with Windows 8.

```
.text:0000000000010524      ; CODE XREF: fnDispatchIoControl:loc_1060E1p
.text:0000000000010524      ; DATA XREF: .pdata:0000000000010924j
.text:0000000000010524      fnDisableSMEPAndCallFunction proc near
.text:0000000000010524      var_28             = qword ptr -28h
.text:0000000000010524      pFunction_         = qword ptr -20h
.text:0000000000010524      Argument           = qword ptr -18h
.text:0000000000010524      pFunction          = qword ptr 8
.text:0000000000010524      mov     [rsp+pFunction], rcx
.text:0000000000010529      sub     rsp, 48h
.text:000000000001052D      mov     rax, [rsp+48h+pFunction]
.text:0000000000010532      mov     rcx, [rsp+48h+pFunction]
.text:0000000000010537      cmp     [rax-8], rcx
.text:000000000001053B      jz      short loc_10541
.text:000000000001053B      xor     eax, eax
.text:000000000001053D      jmp     short loc_1058A
.text:000000000001053F      ; -----
.text:0000000000010541      loc_10541:          ; CODE XREF: fnDisableSMEPAndCallFunction+17fj
.text:0000000000010541      mov     rax, [rsp+48h+pFunction]
.text:0000000000010546      mov     rax, cs:[rdi+SystemOutlineAddress]
.text:000000000001054B      mov     [rsp+48h+Argument], rax
.text:0000000000010552      mov     [rsp+48h+var_28], 0
.text:0000000000010557      lea     rax, fnDisableSMEP
.text:0000000000010560      lea     rcx, [rsp+48h+var_28]
.text:000000000001056C      call    rax
.text:000000000001056C      mov     rcx, [rsp+48h+Argument]
.text:0000000000010573      call    [rsp+48h+pFunction]
.text:0000000000010579      lea     rax, fnEnableSMEP
.text:000000000001057E      lea     rcx, [rsp+48h+var_28]
.text:0000000000010583      call    rax
.text:0000000000010583      mov     eax, 1
.text:0000000000010585      loc_1058A:          ; CODE XREF: fnDisableSMEPAndCallFunction+18fj
.text:000000000001058A      add     rsp, 48h
.text:000000000001058A      retn     c3
.text:000000000001058E      fnDisableSMEPAndCallFunction endp
.text:000000000001058E
```

Figure 7. Function of `Capcom.sys` driver executes code, supplied by the calling process, with SMEP turned off.

Unfortunately, the authors of `Capcom.sys` forgot to add a check to ensure that the caller process was trusted and related to the game. This means that a legitimate digitally-signed driver can be used by malicious software to bypass SMEP. In this case, exploitation is trivial: the malware process just needs to open a descriptor on a device created by the `Capcom.sys` driver and send to it a specific IOCTL code with the `DeviceIoControl` function.

Another common mistake that can make drivers unsafe for the system is the fact that developers don't use the right synchronization during work with CPU-specific operations. For example, the aforementioned `Capcom.sys` driver doesn't use an affinity of thread to the current CPU before disabling SMEP, i.e., it performs this operation incorrectly. If the Windows scheduler decides to swap context after the

function `fnDisableSMEP` is executed, the code that should be run on the CPU with SMEP disabled will instead be executed on another CPU where SMEP is still active.

The situation is complicated where a legitimate driver is compromised by the fact that it is signed with a trusted digital certificate. This means that it is not sufficient only to issue an update for patching the vulnerability. Authors also should revoke the certificate used for signing the vulnerable driver. Otherwise, it can be used by any malicious code for privilege elevation.

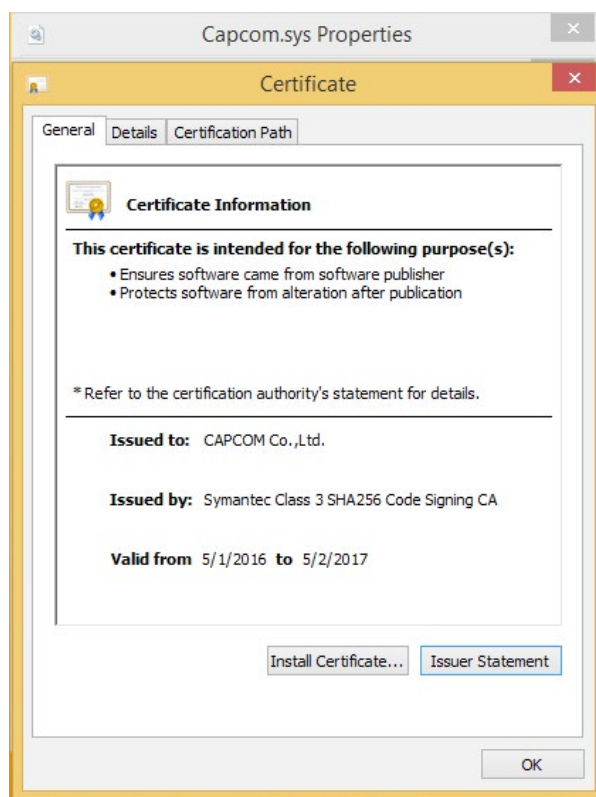


Figure 8. Information about a digital certificate that has been used for signing vulnerable `Capcom.sys` driver. The certificate has still not been revoked.

In August of 2016, specialists in the Russian special service FSB [published](#) a very interesting press release about a sophisticated, highly targeted cyberattack on Russian government organizations. The malware that was used in the cyberattack was multicomponent and contained various modules for silent cyberespionage. ESET security solutions detect this malware as **Win32/Cremes** and **Win64/Cremes**.

This cyberattack differs from way that is chosen by conventional criminals who are interested in personal monetary profit. Our own conclusions coincided with the conclusions of [Kaspersky](#) and [Symantec](#), who also believe that Cremes relates to so-called cyberweapon and was probably deployed by a state-sponsored actor. For example, Cremes uses Lua scripts in its work; this method was also used by other cyberweapons like [EvilBunny](#) and [Flamer](#). The complexity of the modules and their quantity also suggests that Cremes is related to the Flame authors. Among these indicators, we can say that the authors of the Trojan were highly skilled, write tight code, used unusual tricks and wanted to remain undetected as long as possible.

Cremes contains two plugins with the names *kgate* (kernel gate) and *xkgate* (extended kernel gate) that are used by the attackers to run arbitrary code with SYSTEM privileges. Unlike such well-known bootkits as [TDL4](#) or [Gapz](#), the Cremes plugins don't leverage early-boot-stage low-level NT kernel operations and MBR rewriting; instead they use legitimate drivers from two security vendors, Agnitum and Avast. The plugin *kgate* uses a flaw in Agnitum's `Sandbox.sys` driver to load its own malicious driver into kernel mode. It masks its malicious driver as a legitimate Agnitum plugin and forces `Sandbox.sys` to load it into memory.


```

jCheckOnSMEPBypass:                ; CODE XREF: FnDispatchDeviceIoControl+7F↑j
    cmp     eax, 1173000Ch
    jnz     short loc_4003BD

    call    [esi_RootkitStruct+RootkitStruct.pKeQueryActiveProcessors]

    lea     edx, ds:0FFFFFFFh[eax*2]
    and     edx, eax
    push    edx
    call    [esi_RootkitStruct+RootkitStruct.pKeSetSystemAffinityThread]

    mov     ecx, [ebp+DeviceObject]
    call    fnDisableSMEP

    mov     edi, eax
    test    edi, edi
    js      short loc_4003B8

    push    dword ptr [ebx_InputUserBuffer+1Ch]
    call    dword ptr [ebx_InputUserBuffer+18h] ; execute function with SMEP bypass

    mov     ecx, [ebx_InputUserBuffer+20h]
    xor     edi, edi
    mov     [ecx], eax

loc_4003B8:
    call    [esi_RootkitStruct+RootkitStruct.pKeRevertToUserAffinityThread] ; CODE XREF: FnDispatchDeviceIoControl+D2↑j
    jmp     short jCleanupAndRet

```

Figure 9. Kernel mode driver of Cremes malware turns off SMEP before executing user mode function in right way (unlike Capcom.sys authors).

The second plugin, for 64-bit Windows versions, uses Avast's virtualization driver to run its own code in kernel mode. Both plugins run their own kernel mode code as a gateway for code execution with maximum SYSTEM privileges.

```

.text:0000000180004C1A
.text:0000000180004C1A jDropFiles:                ; CODE XREF: FnLoadAvast+1AD↑j
.text:0000000180004C1A
.text:0000000180004C21
.text:0000000180004C28
.text:0000000180004C2C
.text:0000000180004C2F
.text:0000000180004C2F
.text:0000000180004C34
.text:0000000180004C3B
.text:0000000180004C42
.text:0000000180004C44
.text:0000000180004C44
.text:0000000180004C4A
.text:0000000180004C51
.text:0000000180004C54
.text:0000000180004C54
.text:0000000180004C5A
.text:0000000180004C5A
.text:0000000180004C5F
.text:0000000180004C62
.text:0000000180004C64
.text:0000000180004C64
.text:0000000180004C6A
.text:0000000180004C71

    lea     r9, [rbp+1A0h+h_snxhk64.dll]
    lea     r8, [rbp+1A0h+h_snxhk.dll]
    lea     rdx, [rbp+1A0h+ApplicationName_aswSnx.exe]
    mov     ecx, r12d
    call    fnDropAvastDriverAndPrepareEnv

    mov     r15, [rbp+1A0h+h_snxhk.dll]
    mov     rsi, [rbp+1A0h+h_snxhk64.dll]
    test    eax, eax
    jz      jCleanupAndRet

    mov     [rbp+1A0h+var_4C], dil
    test    r12d, r12d
    jz      loc_180004D19

    call    fnCreateAvastServiceAndLoadDriver

    xor     r12d, r12d
    test    eax, eax
    jz      jCleanupAndRet

    lea     rcx, aNtdll ; "ntdll"
    call    cs:GetModuleHandleA

```

Figure 10. Cremes plugin uses the Avast driver to run its own code in kernel mode.

Everything updated, everything secured, everything... EMETed

We have already written several times about the Enhanced Mitigation Experience Toolkit (EMET) in our previous reports. This tool was introduced by Microsoft as a freeware solution to provide strong protection for users against RCE exploits and drive-by-download cyberattacks. Another purpose of EMET is to raise the level of system protection under older and thus less secure Windows versions. Starting from version 5.5, released at the beginning of 2015, EMET provides protection against LPE exploits with its *Block Untrusted Fonts* mitigation. This feature works only on Windows 10.

Like other security software, EMET may potentially contain certain types of vulnerabilities that enable the bypassing of stronger security restrictions that have been deployed. One such vulnerability was [discovered](#) by FireEye security researchers and US-CERT has issued a security [advisory that describes it](#). The vulnerability allows attackers to remove all EMET hooks into a protected process using an internal EMET function. It is based on using a standard internal feature of EMET in order to disable its measures for process protection at run-time. To disable it, attackers just need to call `emet!DllMain` with the `DLL_PROCESS_DETACH` constant. Since the vulnerable version of EMET didn't control the way in which `kernel32!GetModuleHandleW` is called, shellcode may use this function for retrieving its base address and pass it into `emet!DllMain` along with `DLL_PROCESS_DETACH`. The vulnerability was fixed in EMET 5.5.

Hooked/Modified Object	Hook Redirection/Info	Type of Hook	Original Instruction / Bytes	New Instruction / Bytes
[5936] OUTLOOK.EXE->ntdll.dll!NtAllocateVirtualMemory	0x7782C3B0 => [0x35860708] :: \$xR35860000+0x708	Inline - Detour [12 Bytes]	mov eax, 00000017h	jmp 35860708h
[5936] OUTLOOK.EXE->ntdll.dll!ZwMapViewOfSection	0x7782C4B0 => [0x35861950] :: \$xR35860000+0x1950	Inline - Detour [12 Bytes]	mov eax, 00000027h	jmp 35861950h
[5936] OUTLOOK.EXE->ntdll.dll!ZwUnmapViewOfSection	0x7782C4D0 => [0x35861808] :: \$xR35860000+0x1808	Inline - Detour [12 Bytes]	mov eax, 00000029h	jmp 35861808h
[5936] OUTLOOK.EXE->ntdll.dll!NtWriteVirtualMemory	0x7782C5D0 => [0x35861158] :: \$xR35860000+0x1158	Inline - Detour [12 Bytes]	mov eax, 00000039h	jmp 35861158h
[5936] OUTLOOK.EXE->ntdll.dll!ZwCreateSection	0x7782C6D0 => [0x35861608] :: \$xR35860000+0x1608	Inline - Detour [12 Bytes]	mov eax, 00000049h	jmp 35861608h
[5936] OUTLOOK.EXE->ntdll.dll!ZwCreateProcessEx	0x7782C790 => [0x35860E10] :: \$xR35860000+0x0E10	Inline - Detour [12 Bytes]	mov eax, 0000004Ch	jmp 35860E10h
[5936] OUTLOOK.EXE->ntdll.dll!NtProtectVirtualMemory	0x7782C7B0 => [0x35860960] :: \$xR35860000+0x0960	Inline - Detour [12 Bytes]	mov eax, 0000004fh	jmp 35860960h
[5936] OUTLOOK.EXE->ntdll.dll!NtCreateFile	0x7782C7B0 => [0x358613B0] :: \$xR35860000+0x13B0	Inline - Detour [12 Bytes]	mov eax, 00000054h	jmp 358613B0h
[5936] OUTLOOK.EXE->ntdll.dll!NtCreateProcess	0x7782CC00 => [0x35860D98] :: \$xR35860000+0x0D98	Inline - Detour [12 Bytes]	mov eax, 000000A0h	jmp 35860D98h
[5936] OUTLOOK.EXE->ntdll.dll!NtCreateThreadEx	0x7782CCD0 => [0x35860FF0] :: \$xR35860000+0x0FF0	Inline - Detour [12 Bytes]	mov eax, 000000B0h	jmp 35860FF0h
[5936] OUTLOOK.EXE->ntdll.dll!NtCreateUserProcess	0x7782CD80 => [0x35860D20] :: \$xR35860000+0x0D20	Inline - Detour [12 Bytes]	mov eax, 000000B7h	jmp 35860D20h
[5936] OUTLOOK.EXE->ntdll.dll!LdrLoadDll	0x77846680 => [0x358604B0] :: \$xR35860000+0x04B0	Inline - Detour [8 Bytes]	mov edi, edi	jmp 358604B0h
[5936] OUTLOOK.EXE->ntdll.dll!RtlCreateHeap	0x7784C060 => [0x35860AC8] :: \$xR35860000+0x0AC8	Inline - Detour [10 Bytes]	push 000000E0h	jmp 35860AC8h
[5936] OUTLOOK.EXE->kernel32.dll!CreateFileMappingA	0x758770F0 => [0x35861428] :: \$xR35860000+0x1428	Inline - Detour [8 Bytes]	mov edi, edi	jmp 35861428h
[5936] OUTLOOK.EXE->kernel32.dll!VirtualProtect	0x75878AB0 => [0x35860780] :: \$xR35860000+0x0780	Inline - Detour [12 Bytes]	mov edi, edi	jmp 35860780h
[5936] OUTLOOK.EXE->kernel32.dll!MapViewOfFile	0x75878B90 => [0x35861680] :: \$xR35860000+0x1680	Inline - Detour [12 Bytes]	mov edi, edi	jmp 35861680h
[5936] OUTLOOK.EXE->kernel32.dll!VirtualAlloc	0x75878B90 => [0x35860528] :: \$xR35860000+0x0528	Inline - Detour [12 Bytes]	mov edi, edi	jmp 35860528h
[5936] OUTLOOK.EXE->kernel32.dll!LoadLibraryA	0x75878F80 => [0x35860168] :: \$xR35860000+0x0168	Inline - Detour [9 Bytes]	mov edi, edi	jmp 35860168h

Figure 11. Some EMET hooks. The above vulnerability enables the legitimate removal of hooks from a process.

Detailed information about other EMET security features can be found in ["Windows exploitation in 2014"](#).

Let's talk about firmware security

Exploiting firmware vulnerabilities in order to achieve the deepest level of persistence in the system is the Holy Grail for real-world attackers. Such a level of persistence gives them one big advantage: their malicious code can be independent from an installed OS or a hypervisor that can run multiple operating systems on the machine. In other words, it can survive not only Windows reinstallation, but also operations on hard drives, including low-level formatting, because firmware is stored on a special [SPI](#) flash chip on the motherboard (NVRAM, NVS). Malicious code, which can be inserted onto the flash chip, is also independent of the OS version or architecture and can contain modules for various operating systems. Hyper-V security measures described earlier, such as Device Guard, Credential Guard and HVCI, also can be compromised with a firmware rootkit.

The aforementioned flash chip is intended to store [UEFI firmware](#), which consists of a set of drivers for servicing the early stages of a PC's boot process and other low-level system operations: for example, [System Management Mode](#) (SMM). Unlike older BIOS firmware, UEFI can provide an authenticated computer boot process to ensure that this process wasn't compromised before execution flow was passed to the OS. Such a security measure protects firmware and the early boot process from various potential threats such as bootkits. This verification of system modules is called [Secure Boot](#) and if you look at a modern Windows versions loader – `winload.efi` – you can see that it is signed with a digital certificate, as are other drivers located in the UEFI file system.

```
D:\SysinternalsSuite>sigcheck C:\Windows\System32\winload.efi

Sigcheck v2.51 - File version and signature viewer
Copyright (C) 2004-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

c:\windows\system32\winload.efi:
Verified:      Signed
Signing date:  23:16:14 02.2016
Publisher:     Microsoft Windows
Company:       Microsoft Corporation
Description:   OS Loader
Product:       Microsoft Windows Operating System
Prod version:  6.3.9600.18233
File version:  6.3.9600.18233 (winblue_ltsb.160210-0600)
MachineType:  64-bit
```

Figure 12. Signed Windows boot loader file that replaces the older `ntldr` and supports UEFI trusted boot.

The image below shows the UEFI boot process in detail.

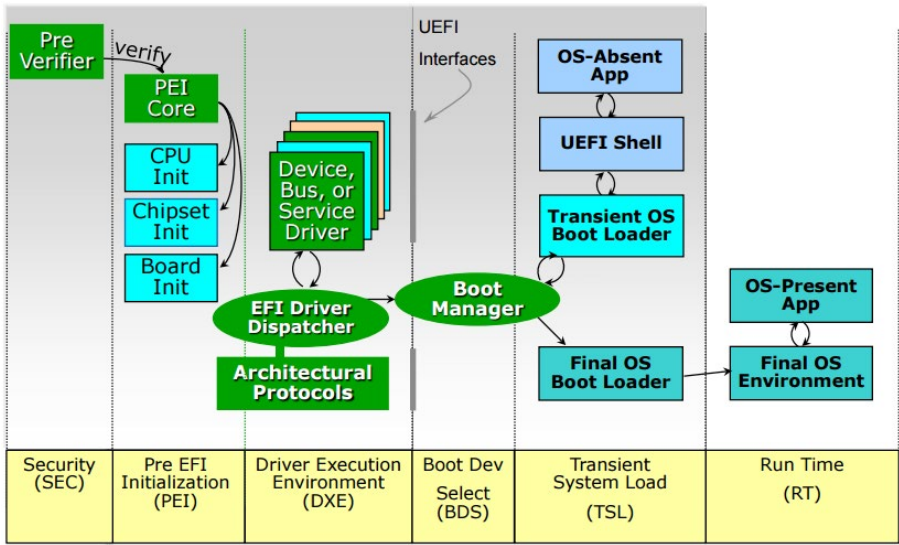


Figure 13. Authentic OS [boot](#) process with UEFI firmware.

Secure Boot is a good security feature for protecting the system's early boot process, but it has nothing to do with protection of the SPI flash chip, the content of which can be easily damaged by malicious code so as to make the system unbootable. To protect the SPI flash chip from overwriting, manufacturers use several hardware options, including the [BIOS_CNTL register](#) and [SPI Protected Ranges](#).

The first feature simply implements protection of the whole chip, while the second can be used to set hardware protection on regions of the chip's memory. Both physically belong to special controllers located on the PC's motherboard.

While a firmware rootkit can provide attackers with the deepest level of persistence, to implement one reliably and across multiple platforms presents major problems. The main issue is that deployment of such a rootkit is very complicated work and the code that needs to be developed is highly platform-specific. Unlike

the usual kernel mode rootkits that work at Windows level and can rely on making use of its API even in the case of low-level disk operations, a firmware backdoor has no such advantage. Moreover, it is forced to work directly with the hardware.

In the last year security researcher Dmytro Oleksiuk has published an analysis of a 0-day LPE vulnerability in the UEFI firmware of computers manufactured by Lenovo. Lenovo's security team called his analysis "uncoordinated disclosure", because details of the vulnerability were published without previously providing details to the vendor. The vulnerability was called ThinkPwn and is located in one of the UEFI drivers, called *SystemSmmRuntimeRt*. To understand how this exploit can be used for defeating the aforementioned flash chip security measures, we should explain several things about [SMM](#).



Figure 14. The indicated Macronix flash chip, with a capacity of 64MB, contains UEFI firmware (ASUS motherboard). UEFI firmware also can be split between several chips if capacity of one is insufficient.

System Management Mode (SMM) is a special working mode of the microprocessor in which separate code with a high level of privilege is executed. When the microprocessor is switched into SMM, execution of any code from the OS is stopped and it executes the handler from so-called SMRAM. SMRAM is a region of computer memory which is not available for access to anyone due to special security restrictions. SMM is used by hardware and the OS to solve issues with several system functions, including power management and system hardware control.

The ThinkPwn vulnerability allows attackers to run arbitrary SMM code with the ability to turn off SPI Protected Ranges (PRx): a security measure that can be misused for deploying a firmware backdoor and compromising Secure Boot. To this end, the exploit uses a known method, described in the presentation [Attacks on UEFI security](#) by Rafal Wojtczuk and Corey Kallenberg, that relies on exploiting the so-called [S3 resume boot path](#) mode of the computer. *S3 resume* is a power saving feature defined in the [Advanced Configuration and Power Interface](#) (ACPI) specification and is used to wake up from the S3 sleep state. To preserve the state of important hardware registers, including PRx, firmware uses a special structure called the Boot Script Table. It is used to save the content of hardware registers during firmware initialization (DXE UEFI loading phase) and to restore their content after S3 resumes. This structure is the main target of firmware attacks because, by modifying it and triggering S3 resume on the next step, attackers can zero PRx registers, thus turning off flash chip protection. The Boot Script Table is stored in the ACPI NVS (Non-Volatile Storage).

To prevent such a vector of attack against the Boot Script Table, the UEFI standard introduced a special security measure called SMM LockBox. Instead of using the usual ACPI NVS, it offers the use of SMRAM as storage for the structure. In this way, only trusted SMM UEFI-code can get access to it. As demonstrated by Oleksiuk, the method of exploitation uses the aforementioned LPE vulnerability in UEFI to execute a malicious SMM handler that can be used to bypass SMM LockBox. After such exploitation succeeds, the exploit can modify PRx fields in the Boot Script Table and trigger S3 resume mode so as to load new values into the registers. Once write protection has been removed from the SPI flash chip, attackers can deploy their own firmware backdoor.

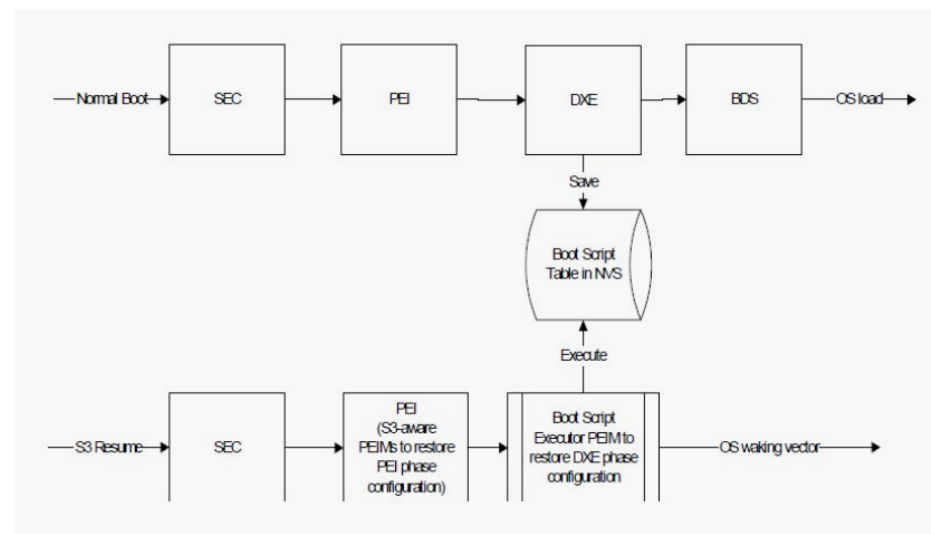


Figure 15. UEFI code saves the content of Boot Script Table into NVS during the [DXE phase](#), but it can be accessed by other non-legitimate code.

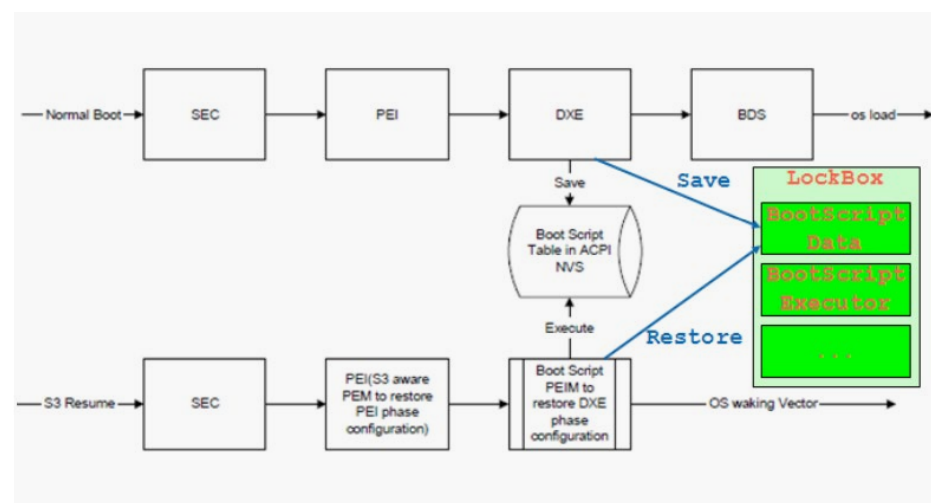


Figure 16. SMM LockBox protects the content of the Boot Script Table from modification using SMRAM.

ThinkPwn has been discovered on the Lenovo laptop ThinkPad T450s and was confirmed by Lenovo in its [LEN-8324](#) security advisory. But the danger from ThinkPwn is greater than it seems at first sight, because the vulnerable firmware driver was not developed by Lenovo and has been used by one of the independent BIOS vendors (IBVs). In turn, IBVs use Intel and AMD code bases to develop UEFI firmware. This means that the range of potentially vulnerable laptops and other computers is not limited to Lenovo, and the same problem [was found](#) on computers from Dell, Hewlett Packard and Fujitsu. Later, Lenovo also confirmed that vulnerable firmware is also installed in the IdeaPad computer series.

After Lenovo's advisory, Hewlett-Packard and Intel also confirmed the presence of ThinkPwn in their hardware. HP has issued security advisory [HPSBBHF3549](#) (ThinkPwn UEFI BIOS SmmRuntime Escalation of Privilege) and listed laptops that are vulnerable to ThinkPwn. Other models of HP laptops are also vulnerable: HP EliteBook 725/745/755 G2 Notebook PC, HP ProBook 4435s/ 4436s/4445s/4446s/4535s/4545s Notebook PC, HP ProBook 445 G1/G2 Notebook PC, HP ProBook 455 G1/G2 Notebook PC and [others](#).

Intel's security advisory is called [INTEL-SA-00056](#) (SmmRuntime Escalation of Privilege) and described motherboards for server computers S1200/1400/1600/2400/2600/4600 series as hardware vulnerable to ThinkPwn. Both Intel and HP patched the vulnerability in firmware.

It is worth noting that privileged SMM mode cannot be leveraged by attackers to compromise hypervisor security measures. Consider the diagram below, where you can see three additional rings of privileges, two of which were [mentioned](#) already long time ago by Invisible Things Lab researchers. The last (-3) privilege level was introduced by Intel Management Engine security researchers.

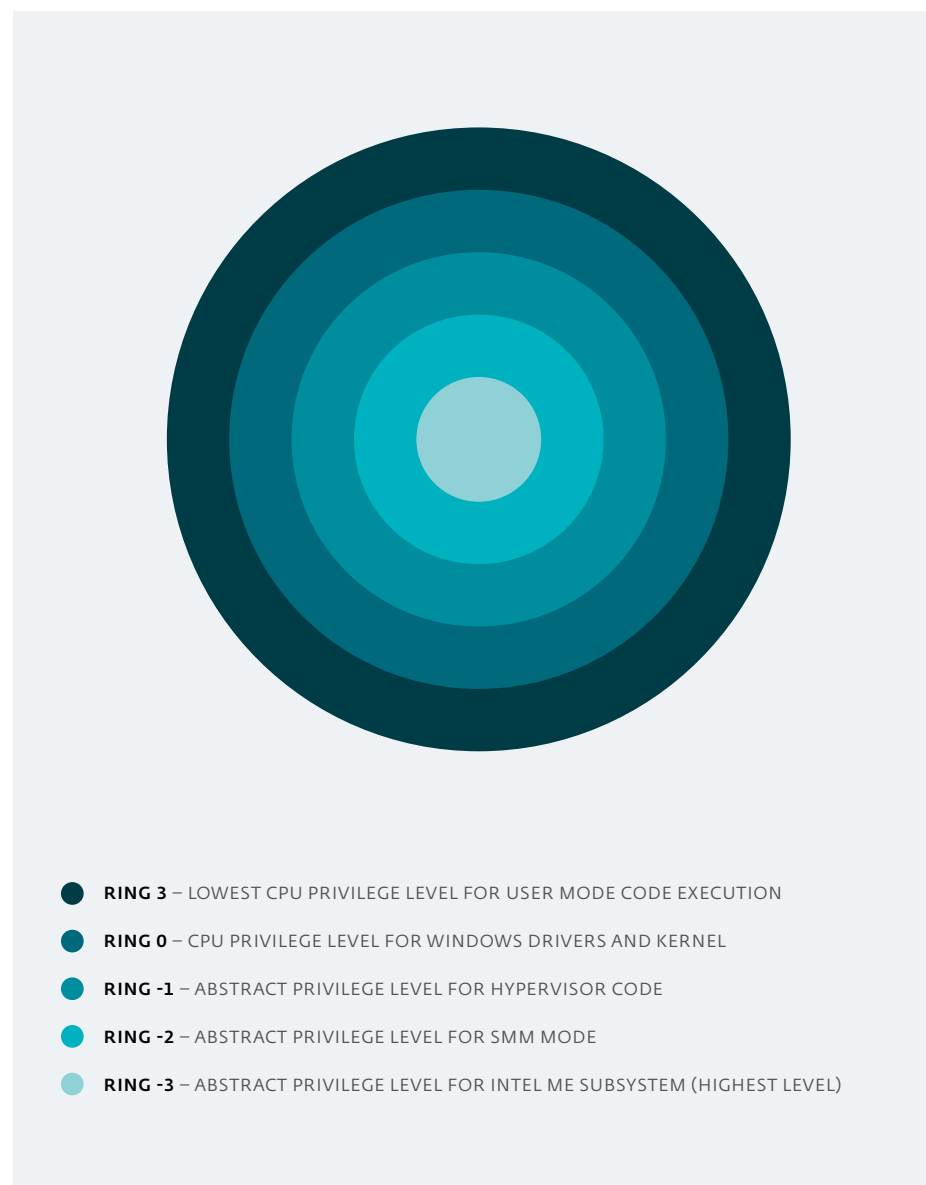


Figure 17. Privilege levels in modern computer systems.

The [Intel Management Engine \(ME\)](#) represents a special firmware subsystem that is located in the chipset and is intended for remote PC management regardless of what OS is running and even, in some cases, when the PC is turned off. It uses some regions of physical memory that should be blocked by firmware for security reasons, because malicious applications can get access to these regions and rewrite them for their own purposes.

Lenovo also fixed one vulnerability in the firmware of its desktop computers and laptops that was related to Intel ME protection. This vulnerability has the identifier CVE-2016-8222 (Intel ME protection not set on some Lenovo Notebook and ThinkServer systems) and was fixed by security update [LEN-9903](#). The vulnerability is related to Security Feature Bypass (SFB) or Local Privilege Escalation (LPE) and allows attackers to get access to physical memory belonging to the Intel ME.

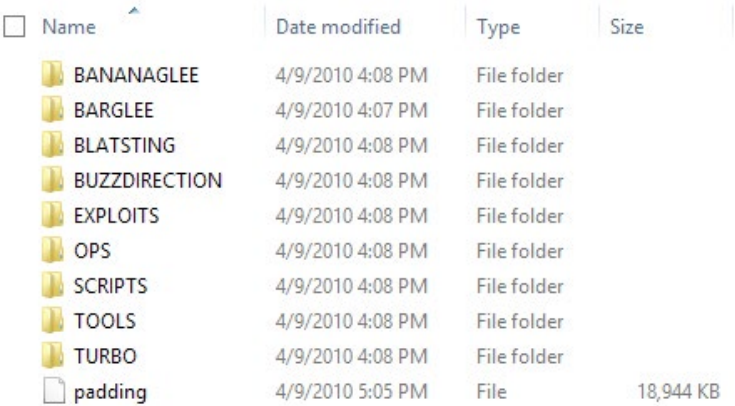
Equation group data breach

In our previous report, we discussed how the famous cybergroup Hacking Team (HT) was compromised. This cybergroup has specialized in developing its own complex surveillance software for various desktop and mobile platforms. The breach was very large and they have lost all confidential data and software sources. Now we already [know](#) that HT was hacked by a hacker with the nickname Phineas Phisher, who used some weaknesses and vulnerabilities in their services with the help of various tools.

As we know from data intended to be secret, but [published](#) by NSA contractor Edward Snowden, NSA has some exploits and special surveillance tools for use in their cyber-operations. Some of them are attributed to a covert NSA unit that is called [Tailored Access Operations \(TAO\)](#). Based on formerly-secret data revealed by Snowden, security researchers have created a special catalog of NSA exploits and implants, which is called [ANT](#). We can liken the NSA TAO unit to the HT cybergroup, because it pursues similar objectives.

Various analyses of the most powerful cyberweapons made by different security companies have demonstrated to the security community and journalists that they can be [attributed](#) to the NSA cybergroup called the [Equation Group](#). According to the style of writing the malicious code, its complexity, and observation of which victims are targeted, it is now supposed that the Stuxnet, Duqu, Flame, and Regin malware were made by the Equation Group [[1](#), [2](#), [3](#), [4](#)]. For example, an unnamed employee of the NSA, interviewed in the documentary movie [Zero Days](#), directed by Alex Gibney, confirmed that Stuxnet was developed by NSA.

In August 2016, one hacking group who call themselves The Shadow Brokers (TSB) [announced](#) that they had gained access to secret data belonging to the Equation Group. They released an archive containing so-called public and private data. The public part of the data is to be found in another archive with installation scripts, configuration files, information about work with C&C, working exploits and implants for [network devices](#) (hardware firewalls) such as those supplied by vendors like Cisco, Fortinet, Juniper Networks, and TOPSEC. The private part consists of an archive that, as declared by TSB, contains sophisticated malware code and exploits. The released public archive contains more than 3,500 files and has a size of around 300MB.



<input type="checkbox"/>	Name	Date modified	Type	Size
	BANANAGLEE	4/9/2010 4:08 PM	File folder	
	BARGLEE	4/9/2010 4:07 PM	File folder	
	BLATSTING	4/9/2010 4:08 PM	File folder	
	BUZZDIRECTION	4/9/2010 4:08 PM	File folder	
	EXPLOITS	4/9/2010 4:08 PM	File folder	
	OPS	4/9/2010 4:08 PM	File folder	
	SCRIPTS	4/9/2010 4:08 PM	File folder	
	TOOLS	4/9/2010 4:08 PM	File folder	
	TURBO	4/9/2010 4:08 PM	File folder	
	padding	4/9/2010 5:05 PM	File	18,944 KB

Figure 18. Content of FIREWALL folder that is a public part of data released by TSB.

Cisco confirmed that the leaked data contain two exploits for its network devices: EXTRABACON (EXBA) and EPICBANANA (EPBA). The first one has the identifier CVE-2016-6366 and represents the [Cisco ASA SNMP RCE vulnerability](#) (0-day), while the second, with the identifier CVE-2016-6367, is called [Cisco ASA CLI RCE](#) (1-day). Both exploits are located in device firmware and can be used by attackers to penetrate into the device and to obtain full control of the system. For example, EPICBANANA affects the following devices: Cisco ASA 5500 Series Adaptive Security Appliances, Cisco ASA 5500-X Series Next-Generation Firewalls, Cisco PIX Firewalls, Cisco Firewall Services Module (FWSM). The vulnerability was fixed as of the release of Cisco ASA v8.4(3).









<input type="checkbox"/> Name	Date modified	Type	Size
 EGBL	4/9/2010 4:08 PM	File folder	
 ELBA	4/9/2010 4:08 PM	File folder	
 ELBO	4/9/2010 4:08 PM	File folder	
 ELCA	4/9/2010 4:08 PM	File folder	
 ELCO	4/9/2010 4:08 PM	File folder	
 EPBA	4/9/2010 4:08 PM	File folder	
 ESPL	4/9/2010 4:08 PM	File folder	
 EXBA	4/9/2010 4:08 PM	File folder	

Figure 19. Directories with exploits for various network devices.

Later, Cisco also [confirmed](#) that another 0-day vulnerability with the identifier CVE-2016-6415 is also present in Cisco IOS, Cisco IOS XE, and Cisco IOS XR software. The vulnerability relates to Information Disclosure type and allows attackers to remotely retrieve sensitive and secret data from a device with vulnerable software, leveraging a specially-crafted network packet for the Internet Key Exchange version 1 (IKEv1) protocol. These sensitive data may represent private RSA keys that are used for encryption purposes, for example, for secure VPN connections.

Fortinet also has issued security advisory [FG-IR-16-023](#) addressing a vulnerability called the Cookie Parser Buffer Overflow Vulnerability and used by the exploit EGREGIOUSBLUNDER (EGBL). This exploit is detected by ESET security products as `Linux/Exploit.Egbl`. The vulnerability affects FortiGate (FOS) firmware v4.3.8 and below. It is used by attackers to execute malicious code with help of a specially crafted HTTP-request.

Acronym	Full name	ESET detection
ESPL	ESCALATEPLOWMAN	Linux/Exploit.Espl
EGBL	EGREGIOUSBLUNDER	Linux/Exploit.Egbl
ELBA	ELIGIBLEBACHELOR	Linux/Exploit.Elba

Table 10. ESET detections for Equation Group exploits.

The exploits mentioned above targeted Cisco ASA devices, but the TSB public archive also contains one special implant named JETPLOW that, unlike those exploits, provides persistence and continuous access for attackers on compromised devices. As stated in a Cisco [blog](#), the JETPLOW backdoor can be successfully mitigated by the Secure Boot security measure that is used to control firmware integrity.

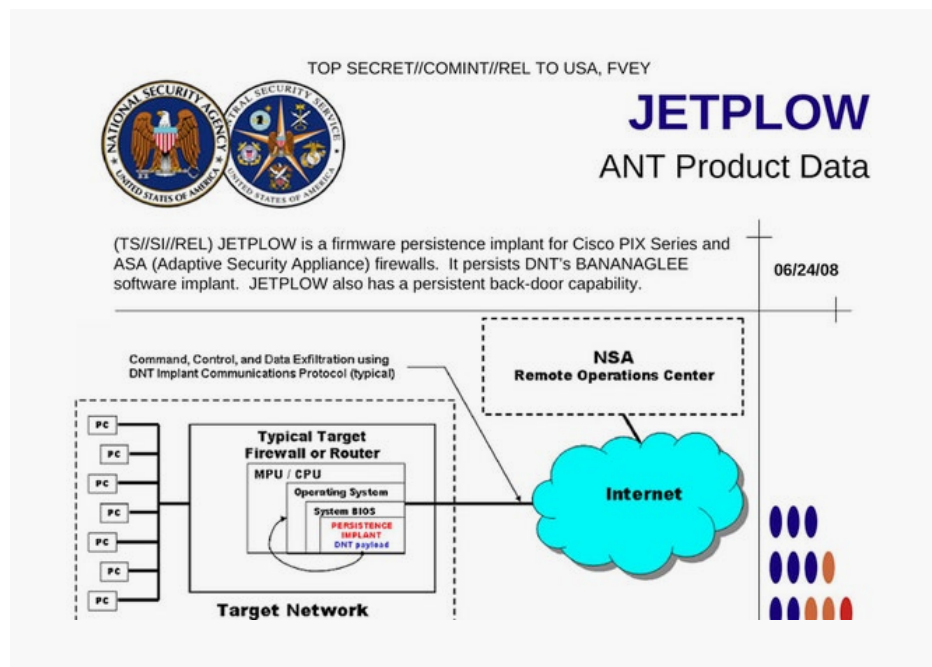


Figure 20. Information about the JETPLOW implant from the ANT catalog.

Juniper Networks also have issued a security advisory [JSA10605](#) that is related to implants such as FEEDTROUGH and ZESTYLEAK for their devices. This malicious software uses various methods to achieve persistence under ScreenOS, including installing malicious BIOS code.

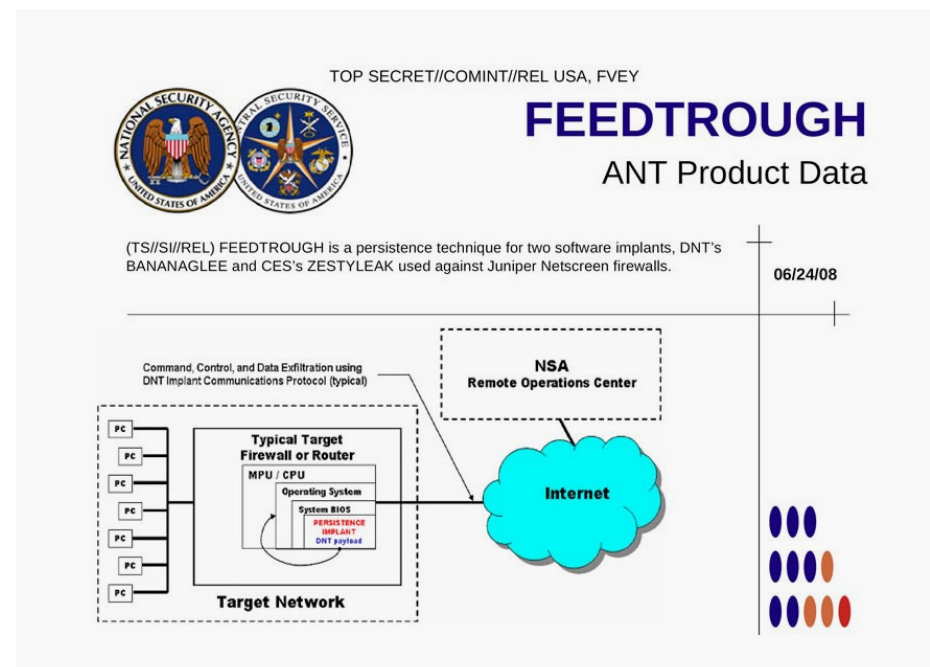


Figure 21. Information about implants for Juniper Netscreen firewalls.

The TSB dump also shows how impressively prepared the cybergroup is for intrusions. For example, components of BANANAGLEE, which is a multicomponent non-persistent implant for Cisco ASA and PIX devices, support such architectures as Intel x86, MIPS, PIX, PowerPC, and Intel XScale. As we can see from source code, another implant called BUZZDIRECTION and targeting Fortigate firewalls, supports the following architectures: Intel x86/x64, PowerPC 32/64-bit, SPARC, MIPS, and ARM. Similarly, the BARGLEE software implant supports Intel x86, MIPS, PowerPC, and XScale.

We have mentioned that some implants provide persistence ability for attackers on compromised devices. For example, directories with components of the BANANAGLEE implant contain various tools for working directly with the BIOS: BB_readBIOS-2100, BB_writeBIOS-2100, BM_readBIOS-2130, BM_writeBIOS-2130, BBALL_ASABIOS-3021.exe. That last tool is intended to work with the BIOS used by Cisco ASA devices.

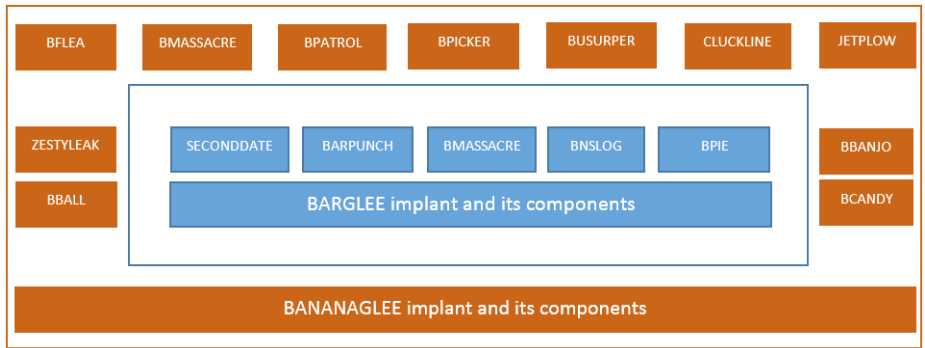


Figure 22. BANANAGLEE implant contains many components mentioned in sources released by TSB.

Our security products detect the BANANAGLEE implant and its components as **Linux/Equation.BananaGlee**. In addition, there are separate detections for its components – **Linux/HackTool.Equation.SecondDate** and **Linux/Equation.BanalRide**. Below are listed other detections for Equation Group tools.

Tool/Implant	ESET detection
BANANAGLEE	Linux/Equation.BananaGlee Linux/HackTool.Equation.BananaGlee
NOPEN	Linux/Equation.Nopen
BARGLEE	Linux/Equation.BarGlee
BUSURPER	Linux/Equation.BananaUsurper
BMASSACRE	Linux/Equation.BananaMassacre
BPIE	Linux/Equation.BananaPie
ZESTYLEAK	Linux/Equation.ZestyLeak
BNSLOG	Linux/Equation.BnsLog
CLUCKLINE	Linux/Equation.CluckLine
BPICKER	Linux/Equation.Bpicker
BBALL	Linux/Equation.Bball
BBANJO	Linux/Equation.Bbanjo
BPATROL	Linux/Equation.Bpatrol
BCANDY	Linux/Equation.Bcandy
ELIGIBLECONTESTANT	Linux/HackTool.Equation.Elco
ELIGIBLECANDIDATE	Linux/HackTool.Equation.Elca
EGREGIOUSBLUNDER	Linux/HackTool.Equation.Egbl
ELIGIBLEBOMBSHELL	Linux/HackTool.Equation.Elbo
ELIGIBLEBACHELOR	Linux/HackTool.Equation.Elba
ESCALATEPLOWMAN	Linux/HackTool.Equation.Espl
SECONDDATE	Linux/HackTool.Equation.SecondDate
PANDAROCK	Linux/HackTool.Equation.PandaRock
BUZZDIRECTION	Linux/HackTool.Equation.BuzzDirection
BLATSTING	Linux/HackTool.Equation.BlatSting
BARICE	Linux/HackTool.Equation.BarIce
DURABLENAPKIN	Linux/HackTool.Equation.DurableNapkin
EXTRABACON	Linux/HackTool.Equation.Exba

Table 11. ESET detections for Equation Group malicious software.

Conclusion

Obviously, the use of a modern up-to-date Windows version, e.g. Windows 10 with the latest updates, is the best approach to being protected from cyberattacks exploiting vulnerabilities. As we have shown above and in previous versions of this report, its components contain useful security features for mitigating RCE and LPE exploits. We can say that actions taken by Microsoft to make modern versions of Internet Explorer more secure were insufficient, because so-called advanced security settings that are built into Edge are still optional in IE.

Baranov Artem

Malware researcher, ESET Russia

I would like to thank my many colleagues at ESET for their suggestions and advice.