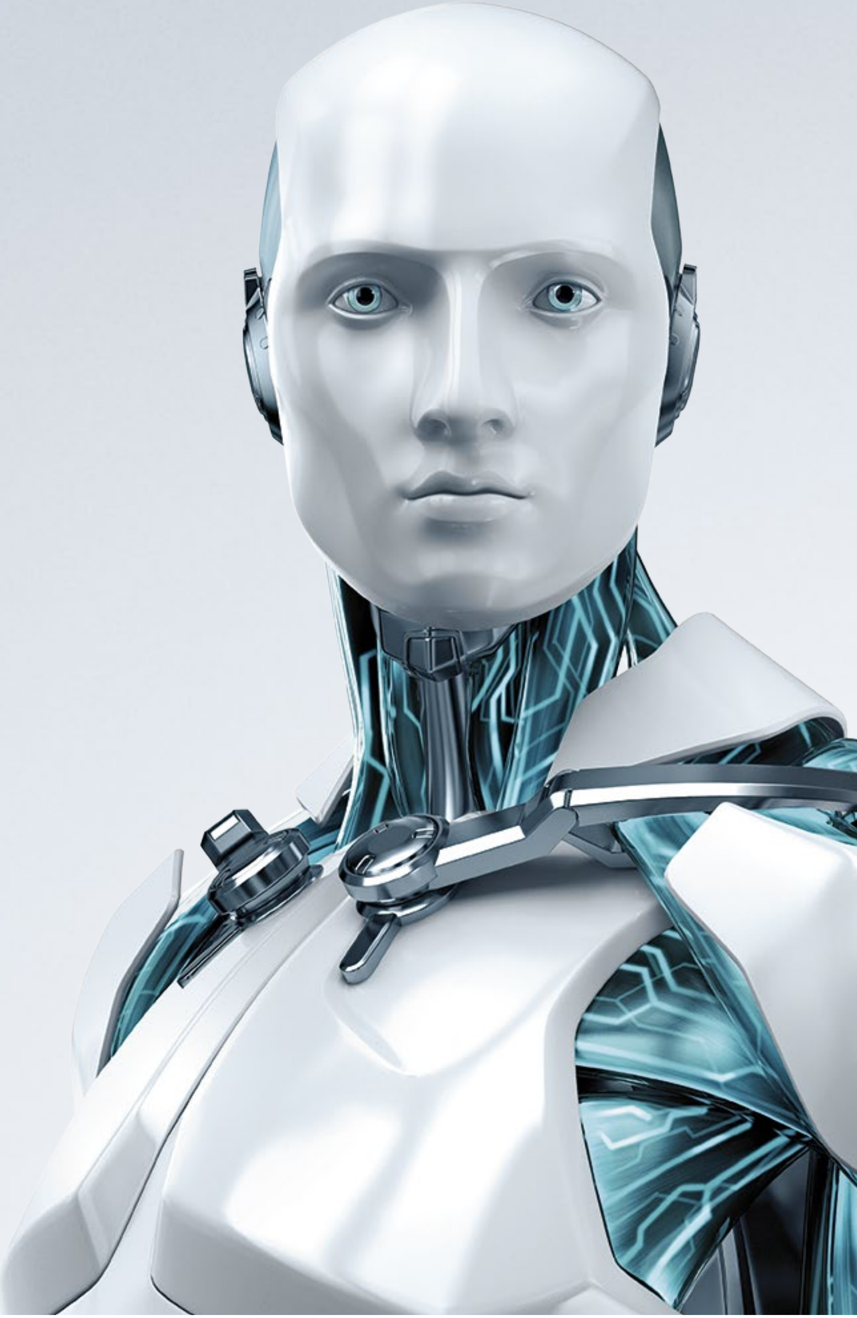
Windows Exploitation in 2014



ENJOY SAFER TECHNOLOGY™

We have decided to write a new version of our *earlier report* on major trends in Windows exploitation and mitigation for 2014. In that report we mentioned that *0day attacks* were a major trend in 2013 and also that cyber criminals have developed 0day exploits specifically for targeted attacks. This trend has maintained its progress in 2014 too.

In this annual report we have added a special section with notes about Internet Explorer (IE). 2014 was really tough on users of this browser, as Microsoft (MS) has addressed twice as many IE vulnerabilities as in 2013. We have also added additional information about exploit mitigation techniques for Windows users and why it's not as easy to secure the operating system as it seems at first glance.

General Information

Table 1 below gives information about vulnerabilities that were closed for the Internet Explorer browser (versions 6 – 11). Vulnerabilities that were exploited by attackers before corresponding updates were available (0day) are highlighted in red. We'll discuss exploitation of Internet Explorer in more detail later, in the "Internet Explorer" section.

Component	Bulletin	Type	Vulnerability
Internet Explorer	MS14-010, MS14-012, MS14-018, MS14-021, MS14-029, MS14-035, MS14-037, MS14-051, MS14-052, MS14-056, MS14-065, MS14-080	Remote Code Execution (12)	CVE-2014-0267, CVE-2014-0268, CVE-2014-0269, CVE-2014-0270, CVE-2014-0271, CVE-2014-0272, CVE-2014-0273, CVE-2014-0274, CVE-2014-0275, CVE-2014-0276, CVE-2014-0277, CVE-2014-0278, CVE-2014-0279, CVE-2014-0280, CVE-2014-0281, CVE-2014-0283, CVE-2014-0284, CVE-2014-0285, CVE-2014-0286, CVE-2014-0287, CVE-2014-0288, CVE-2014-0289, CVE-2014-0290, CVE-2014-0293, CVE-2014-0297, CVE-2014-0298, CVE-2014-0299, CVE-2014-0302, CVE-2014-0303, CVE-2014-0304, CVE-2014-0305, CVE-2014-0306, CVE-2014-0307, CVE-2014-0308, CVE-2014-0309, CVE-2014-0311, CVE-2014-0312, CVE-2014-0313, CVE-2014-0314, CVE-2014-0321, **CVE-2014-0322**, **CVE-2014-0324**, CVE-2014-0235, CVE-2014-1751, CVE-2014-1752, CVE-2014-1753, CVE-2014-1755, CVE-2014-1760, **CVE-2014-1776**, CVE-2014-0310, **CVE-2014-1815**, CVE-2014-0282, CVE-2014-1762, CVE-2014-1764, CVE-2014-1766, CVE-2014-1769, CVE-2014-1770, CVE-2014-1771, CVE-2014-1772, CVE-2014-1773, CVE-2014-1774, CVE-2014-1775, CVE-2014-1777, CVE-2014-1778, CVE-2014-1779, CVE-2014-1780, CVE-2014-1781, CVE-2014-1782, CVE-2014-1783, CVE-2014-1784, CVE-2014-1785, CVE-2014-1786, CVE-2014-1788, CVE-2014-1789, CVE-2014-1790, CVE-2014-1791, CVE-2014-1792, CVE-2014-1794, CVE-2014-1795, CVE-2014-1796, CVE-2014-1797, CVE-2014-1799, CVE-2014-1800, CVE-2014-1802, CVE-2014-1803, CVE-2014-1804, CVE-2014-1805, CVE-2014-2753, CVE-2014-2754, CVE-2014-2755, CVE-2014-2756, CVE-2014-2757, CVE-2014-2758, CVE-2014-2759, CVE-2014-2760, CVE-2014-2761, CVE-2014-2763, CVE-2014-2764, CVE-2014-2765, CVE-2014-2766, CVE-2014-2767, CVE-2014-2768, CVE-2014-2769, CVE-2014-2770, CVE-2014-2771, CVE-2014-2772, CVE-2014-2773, CVE-2014-2775, CVE-2014-2776, CVE-2014-2777, CVE-2014-1763, CVE-2014-1765, CVE-2014-2785, CVE-2014-2786, CVE-2014-2787, CVE-2014-2788, CVE-2014-2789, CVE-2014-2790, CVE-2014-2791, CVE-2014-2792, CVE-2014-2794, CVE-2014-2795, CVE-2014-2797, CVE-2014-2798, CVE-2014-2800, CVE-2014-2801, CVE-2014-2802, CVE-2014-2803, CVE-2014-2804, CVE-2014-2806, CVE-2014-2807, CVE-2014-2809, CVE-2014-2813, CVE-2014-2783, CVE-2014-2774, CVE-2014-2784, CVE-2014-2796, CVE-2014-2808, CVE-2014-2810, CVE2014-2811, **CVE-2014-2817**, CVE-2014-2818, CVE-2014-2819, CVE-2014-2820, CVE-2014-2821, CVE-2014-2822, CVE-2014-2823, CVE-2014-2824, CVE-2014-2825, CVE-2014-2826, CVE-2014-2827, CVE-2014-4050, CVE-2014-4051, CVE-2014-4052, CVE-2014-4055, CVE-2014-4056, CVE-2014-4057, CVE-2014-4058, CVE-2014-4063, CVE-2014-4067, CVE-2014-4145, **CVE-2013-7331**, CVE-2014-2799, CVE-2014-4059, CVE-2014-4065, CVE-2014-4079, CVE-2014-4080, CVE-2014-4081, CVE-2014-4082, CVE-2014-4083, CVE-2014-4084, CVE-2014-4085, CVE-2014-4086, CVE-2014-4087, CVE-2014-4088, CVE-2014-4089, CVE-2014-4090, CVE-2014-4091, CVE-2014-4092, CVE-2014-4093, CVE-2014-4094, CVE-2014-4095, CVE-2014-4096, CVE-2014-4097, CVE-2014-4098, CVE-2014-4099, CVE-2014-4100, CVE-2014-4101, CVE-2014-4102, CVE-2014-4103, CVE-2014-4104, CVE-2014-4105, CVE-2014-4106, CVE-2014-4107, CVE-2014-4108,

Component	Bulletin	Type	Vulnerability
			CVE-2014-4109, CVE-2014-4110, CVE-2014-4111, CVE-2014-4123, CVE-2014-4124, CVE-2014-4126, CVE-2014-4127, CVE-2014-4128, CVE-2014-4129, CVE-2014-4130, CVE-2014-4132, CVE-2014-4133, CVE-2014-4134, CVE-2014-4137, CVE-2014-4138, CVE-2014-4140 (ASLR Bypass), CVE-2014-4141, CVE-2014-4143, CVE-2014-6323, CVE-2014-6337, CVE-2014-6339 (ASLR Bypass), CVE-2014-6340, CVE-2014-6341, CVE-2014-6342, CVE-2014-6343, CVE-2014-6344, CVE-2014-6345, CVE-2014-6346, CVE-2014-6347, CVE-2014-6348, CVE-2014-6349, CVE-2014-6350, CVE-2014-6351, CVE-2014-6353, CVE-2014-6327, CVE-2014-6328, CVE-2014-6329, CVE-2014-6330, CVE-2014-6366, CVE-2014-6368 (ASLR Bypass), CVE-2014-6369, CVE-2014-6373, CVE-2014-6374, CVE-2014-6375, CVE-2014-6376, CVE-2014-8966

Table 1

The table shows that in 2014 Microsoft fixed approximately twice as many vulnerabilities as they did in the previous year. Figure 1 below represents these statistics visually. Microsoft still supports the old (and completely unsafe) browser version Internet Explorer 6. This version is still being distributed with Windows Server 2003. Support for this browser will end in 2015.

Table 2 shows vulnerabilities addressed and updates issued for various types of Windows components. We have combined all Windows user-mode components (UMC) in the section “Windows UMC”. And as you can see there are also several vulnerabilities that were used by attackers for 0day exploits. Even a minimal Windows session runs many services, and attackers can, potentially, make use of vulnerabilities in system services to penetrate the system.

Component	Bulletin	Type	Vulnerability
Windows UMC (VBScript, Direct2D, MSXML, DirectShow, SAMR, File Handling/kernel32.dll, Shell handler/shell32.dll, Remote Desktop, Journal, On-Screen Keyboard, Media center/mcplayer.dll, Installer, Task Scheduler, OLE, Message Queuing, Schannel, Kerberos, Audio Service, IIS, IME (Japanese), GDI+/gdi32.dll, RPC/rpcrt4.dll, Graphics/windowscodecs.dll	MS14-011, MS14-007, MS14-005, MS14-013, MS14-016, MS14-027, MS14-030, MS14-033, MS14-038, MS14-039, MS14-041, MS14-043, MS14-049, MS14-054, MS14-060, MS14-062, MS14-064, MS14-066, MS14-067, MS14-068, MS14-071, MS14-074, MS14-076, MS14-078, MS14-036, MS14-047, MS14-084, MS14-085	Remote Code Execution(11), Information Disclosure(3), Security Feature Bypass(4), Elevation of Privilege(9), Tampering(1)	CVE-2014-0271, CVE-2014-0263, CVE-2014-0266, CVE-2014-0301, CVE-2014-0317, CVE-2014-0315, CVE-2014-1807, CVE-2014-1816, CVE-2014-0296, CVE-2014-1824, CVE-2014-2781, CVE-2014-2780, CVE-2014-4060, CVE-2014-1814, CVE-2014-4074, CVE-2014-4114, CVE-2014-4971, CVE-2014-6332, CVE-2014-6352, CVE-2014-6321, CVE-2014-4118, CVE-2014-6324, CVE-2014-6322, CVE-2014-6318, CVE-2014-4078, CVE-2014-4077, CVE-2014-1818, CVE-2014-0316, CVE-2014-6363, CVE-2014-6355
Win32k	MS14-003, MS14-015, MS14-045, MS14-058, MS14-079	Elevation of Privilege(4), Denial of Service(1)	CVE-2014-0262, CVE-2014-0300, CVE-2014-0323, CVE-2014-0318, CVE-2014-1819, CVE-2014-4113, CVE-2014-4148, CVE-2014-6317
KM drivers (ndproxy.sys, tcpip.sys, afd.sys, fastfat.sys)	MS14-002, MS14-006, MS14-031, MS14-040, MS14-045, MS14-063, MS14-070	Elevation of Privilege(5), Denial of Service(2)	CVE-2013-5065, CVE-2014-0254, CVE-2014-1811, CVE-2014-1767, CVE-2014-4064, CVE-2014-4115, CVE-2014-4076
.NET Framework	MS14-009, MS14-026, MS14-046, MS14-053, MS14-057, MS14-072	Elevation of Privilege(3), Security Feature Bypass(1), Denial of Service(1), Remote Code Execution(1)	CVE-2014-0253, CVE-2014-0257, CVE-2014-0295 (ASLR Bypass), CVE-2014-1806, CVE-2014-4062 (ASLR Bypass), CVE-2014-4072, CVE-2014-4073, CVE-2014-4121, CVE-2014-4122 (ASLR Bypass), CVE-2014-4149

Table 2: Vulnerabilities and Patches

Figure 1 represents the number of vulnerabilities closed this year across a range of components.

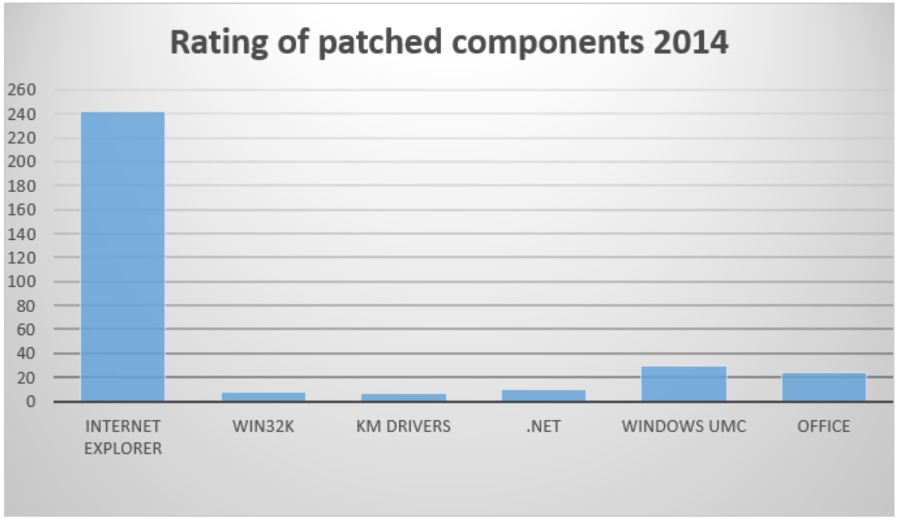


Figure 1

We can see that a great number of vulnerabilities in the web-browser Internet Explorer have been closed in 2014. Almost all of these vulnerabilities were of the "Remote Code Execution" (RCE) type. This meant that an attacker could execute code remotely in a vulnerable environment, with the help of a specially-crafted web page. Such a web pages could contain special code, called an exploit, to trigger a specific vulnerability. Usually attackers use such exploits for silently installing malware when they detect a vulnerable Windows version. This attack is an example of a *drive-by download* and this is why we highlighted such exploitations as a major trend in attacks on Internet Explorer, as shown in Figure 2 below.

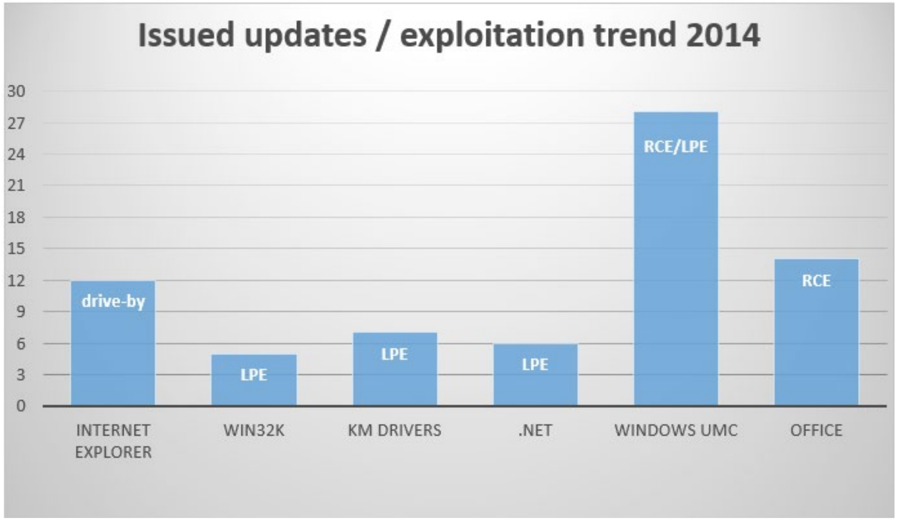


Figure 2

As mentioned above, "drive-by" refers to the silent installation of malware using an RCE (Remote Code Execution) exploit. We distinguish between RCE vulnerabilities and drive-bys, because the term drive-by mostly relates to malware installation via web browser, unlike other remote code execution, for example, with the help of Microsoft Office applications. LPE means *Local Privilege Escalation* or what Microsoft calls Elevation of Privilege (EoP). An attacker uses such vulnerabilities for obtaining the maximum level of access to any resources in Windows: for example, to work under the SYSTEM account that gives a program the ability to execute arbitrary kernel mode code on 32-bit versions of Windows. Both drive-by downloads and LPE attacks will be discussed in more detail below, in the section "Drive-by download and Local Privilege Escalation attacks".

We can see that the driver `win32k.sys` and other drivers in Windows, highlighted in the column named KM drivers (Kernel Mode drivers), are typical components used by attackers to obtain maximum privileges within the OS. Such exploits can be used by malware authors for bypassing restrictions built into Windows so that the attackers can execute kernel mode code (also known as user-mode restrictions escape). In another scenario, an attacker can use such exploits in conjunction with RCE exploits in order to bypass the web browser's sandbox restrictions.

Comparing the number of vulnerabilities addressed in 2014 with the number addressed the previous year (Figure 3) is interesting and instructive.

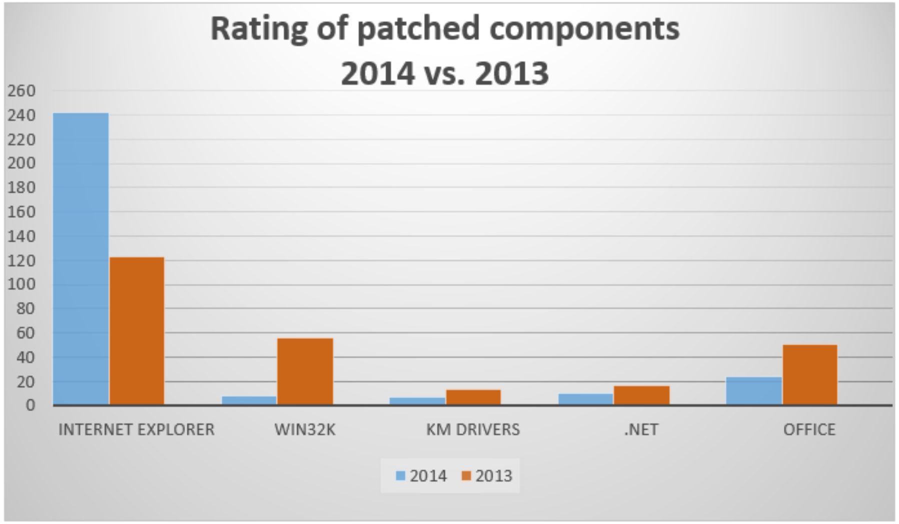


Figure 3

These statistics show us that in 2014 fewer vulnerabilities were closed than in 2013 in all components/products, except for Internet Explorer, in which nearly twice as many bugs were dealt with in 2014.

Vulnerabilities in Office are also often targeted by attackers. During 2014 we discovered various attacks where attackers have used a vulnerability in Microsoft Office and Windows for delivering malicious software. The ESET Research Team was the first to discover a notorious 0day vulnerability – *CVE-2014-4114* – in the OLE package manager (`packager.dll`), which allowed the installation of malware on a victim's computer via a specially-crafted Microsoft PowerPoint presentation. My colleagues *Robert Lipovsky* and *Anton Cherepanov* did a *detailed analysis* of a *malicious campaign* used by cybercriminals to deliver *BlackEnergy* malware using this vulnerability.

Exploitation

In the past year we have seen many vulnerabilities exploited by attackers. Tables in the "General Information" section give the real picture of the vulnerabilities that were used in these attacks. Our malware analysts are closely monitoring this situation and have added exploits for these vulnerabilities into our detection databases as soon as they were discovered as a result of attacks on users. The tables below give additional information about these vulnerabilities and detections by ESET's software that were created for the corresponding exploits.

CVE	Type	Component	Vulnerability	Fixed	Bypass DEP & ASLR, other
CVE-2014-0322	Remote Code Execution	Internet Explorer 10	use-after-free	MS14-012	ActionScript-heap-spray/ROP/EMET check
CVE-2014-0502	Remote Code Execution	Flash Player	double-free	APSB14-07	ActionScript- non-ASLR-hxds.dll/ROP non-ASLR-msvcrt.dll/ROP non-ASLR-msvcr71.dll/ROP
CVE-2014-1761	Remote Code Execution	Word 2003-2013	memory-corruption	MS14-017	non-ASLR-mscomctl.dll/ROP (<=Word 2010)
CVE-2014-1776	Remote Code Execution	Internet Explorer 6-11	use-after-free	MS14-021	ActionScript-heap-spray/ROP
CVE-2014-0515	Remote Code Execution	Flash Player	buffer-overflow	APSB14-13	–
CVE-2014-0160	Information Disclosure	Windows 8.1 & RT 8.1 In-Box Junos Pulse Client (Juniper Networks)	Heartbleed	KB2962140	–
CVE-2014-4114	Remote Code Execution	Windows 2003 Server+ OLE package manager (Packager.dll)	by design (bug)	MS14-060	Remote malware installation via .INF-file
CVE-2014-4113	Elevation of Privileges	Win32k	integer-overflow	MS14-058	NULL pointer dereference on x32 wild pointer dereference on x64
CVE-2014-6352	Remote Code Execution	Windows Vista+ OLE package manager (Packager.dll)	MS14-060 fail	MS14-064	Malware installation via malicious OLE-object
CVE-2014-6332	Remote Code Execution	Windows 2003 Server+ Windows OLE (OleAut32.dll)	by design (bug)	MS14-064	Bypasses (by design) DEP&ASLR

Table 3

We can see exploits here for Internet Explorer, the Adobe Flash Player plugin, Microsoft Word, and various Windows components.

Vulnerability in-the-wild	ESET detection	Month	Targeted attack*
CVE-2014-0322	Win32/Exploit.CVE-2014-0332	February	Yes
CVE-2014-0502	SWF/Exploit.CVE-2014-0502	February	Yes
CVE-2014-1761	Win32/Exploit.CVE-2014-1761	March	Yes
CVE-2014-1776	Win32/Exploit.CVE-2014-1776 SWF/Exploit.CVE-2014-1776 JS/Exploit.Agent.NGS	April	Yes
CVE-2014-0515	SWF/Exploit.CVE-2014-0515	April	Yes
CVE-2014-4114	Win32/Exploit.CVE-2014-4114	October	Yes
CVE-2014-4113	Win64/Dianti.A Win32/Dianti.A	October	Yes
CVE-2014-6352	Win32/Exploit.CVE-2014-6352.A	October	Yes
CVE-2014-6332	Win32/Exploit.CVE-2014-6332.A	November	No

Table 4

The column on the right demonstrates what we already said at the beginning of this article: attackers create 0day exploits to use in targeted attacks. Detection for the last vulnerability listed, CVE-2014-6332, was added after the vulnerability was already fixed by Microsoft so it is not strictly a 0day.

As we have seen, Microsoft has introduced exploit mitigation techniques into every new version of Windows. Most important and already discussed many times in the past are DEP (*Data Execution Prevention*) & ASLR (*Address Space Layout Randomization*). To bypass DEP, attackers resorted to use various types of ROP (*Return Oriented Programming*) gadgets that can be easy located in DLLs compiled without ASLR support. These ROP gadgets represent pieces of code that can help attackers to modify protection of memory pages with shellcode. We came across the use of ROP to bypass DEP in an Adobe Flash Player exploit: more details can be found in the *research* published by my colleague *Sébastien Duquette*.

Another well-known technology for reducing the effectiveness of exploits is ASLR (Address Space Layout Randomization). This can serve as a means of impeding penetration by shellcode, by randomizing the locations of a program's data areas such as the stack and heap, including the addresses used by malicious shellcode. Thus the attacker is unable to predict reliably the correct address in memory into which malicious shellcode needs to be dropped, or to create favourable conditions for using vulnerabilities. In this way, the ASLR security feature protects users from a broad class of vulnerabilities and creates problems for attackers, increasing the overall cost of developing a reliable exploit.

Unfortunately, Windows and its components – as well as.NET Framework or Microsoft Office – can contain legacy non-secure DLL files not compiled with secure options. These executable libraries are very useful for attackers, because they are located in memory at predictable addresses. In table 5 below we can see various vulnerabilities of this kind, most of them being non-secured: that is, compiled without ASLR support.

Vulnerability (ASLR bypass)	Product/Component	Details
CVE-2014-0295 (MS14-009)	.NET Framework 2.0 SP2, .NET Framework 3.5.1	Vulnerability in vsavb7rt.dll Being exploited ITW
CVE-2014-4062 (MS14-046)	.NET Framework 2.0 SP2 – 3.5.1	–
CVE-2014-4122 (MS14-057)	.NET Framework 2.0 SP2 – 3.5.1	–
CVE-2014-0319 (MS14-014)	Silverlight 5	–
CVE-2014-1809 (MS14-024)	Windows Common Control MS Office 2007 - 2013	Vulnerability in mscomctl.ocx Being exploited ITW
CVE-2014-0316 (MS14-047)	Windows 7 – 8.1 / RT 8.1 Local RPC (LRPC)	Implicit ASLR bypass via out-of-process memory-spray
CVE-2014-4140 (MS14-056)	Internet Explorer 9 - 11	–
CVE-2014-6339 (MS14-065)	Internet Explorer 8-9	–
CVE-2014-6368 (MS14-080)	Internet Explorer 11	–

Table 5

Vulnerability CVE-2014-0316 is really interesting, because it can used by attackers to bypass ASLR "remotely" (*out-of-process*) from another process by sending specially crafted LRPC (Local Remote Procedure Call) requests. Attackers can use this vulnerability in conjunction with an RCE vulnerability to facilitate the process of penetration into the system from the browser.

Drive-by Download and Local Privilege Escalation attacks

Today drive-by downloads represent a typical type of attack that criminals use to execute malicious code remotely. As we know, they usually use various types of exploit kits to redirect users so as to enable malware installation. In this scenario, criminals can compromise a legitimate web site by introducing malicious content, which will redirect users onto a landing page where exploits are present.

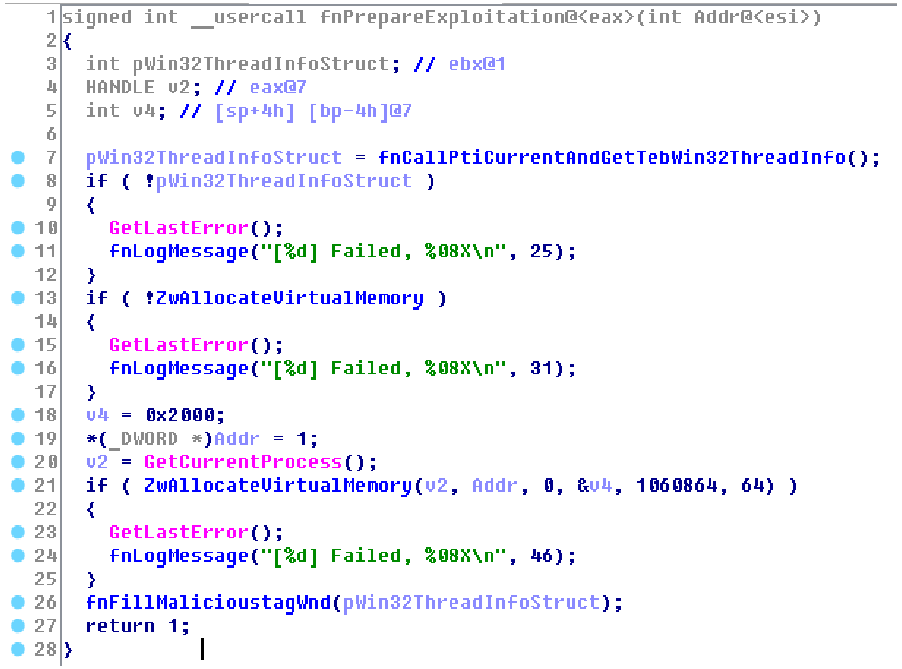
A great example of how a drive-by download can be implemented was demonstrated by my colleagues in a paper called "Operation Windigo". Criminals can use accounts stolen from servers to compromise Linux with malicious binaries. They can also distribute such malicious binaries as additional modules, for example, for Apache software. After the server has been compromised, malware gets full control over the systems of visitors to web sites running on the server and can inject malicious HTML-code into legitimate pages. This is one of the scenarios that allows an attacker to organize drive-by download attacks.

During the last year, we also wrote many times about various groups that use drive-by downloads as the primary vector for the spread of malware. For example, the *Sednit espionage group* used drive-by downloads via exploit kits to deliver malicious software onto the systems of corporate users in Eastern Europe. This group used various Microsoft Internet Explorer exploits to deploy backdoors on compromised computers.

Local Privilege Escalation (LPE) (or EoP) vulnerabilities are usually employed by attackers in only two cases. In the first scenario, an exploit used for a drive-by download attack can use such a vulnerability in conjunction with an RCE vulnerability so as to bypass the browser's sandbox. The sandbox cannot allow shellcode to install malicious software, so attackers need an additional vulnerability to allow them to bypass this restriction.

In the second scenario, malware can use an LPE vulnerability to bypass Windows user-mode restrictions and insert arbitrary code into kernel mode (Ring0) on 32-bit versions of Windows. A bootkit is one such type of malware, and can use an LPE exploit to load its own driver into memory. My colleagues *Eugene Rodionov*, *Aleks Matrosov* and *David Harley* discussed bootkits in their VB2014 presentation *Bootkits: past, present & future*.

This year we saw another LPE exploit (CVE-2014-4113) that used a notorious technique to run arbitrary code in kernel mode with help of NULL pointer dereference (detected by ESET as *Win32/Dianti.A* and *Win64/Dianti.A*). An exploit uses a bug in the `win32k.sys` driver function *win32k!xxxHandleMenuMessages*. Moreover, it can work in a 64-bit version of Windows due to "wild pointer" dereference.

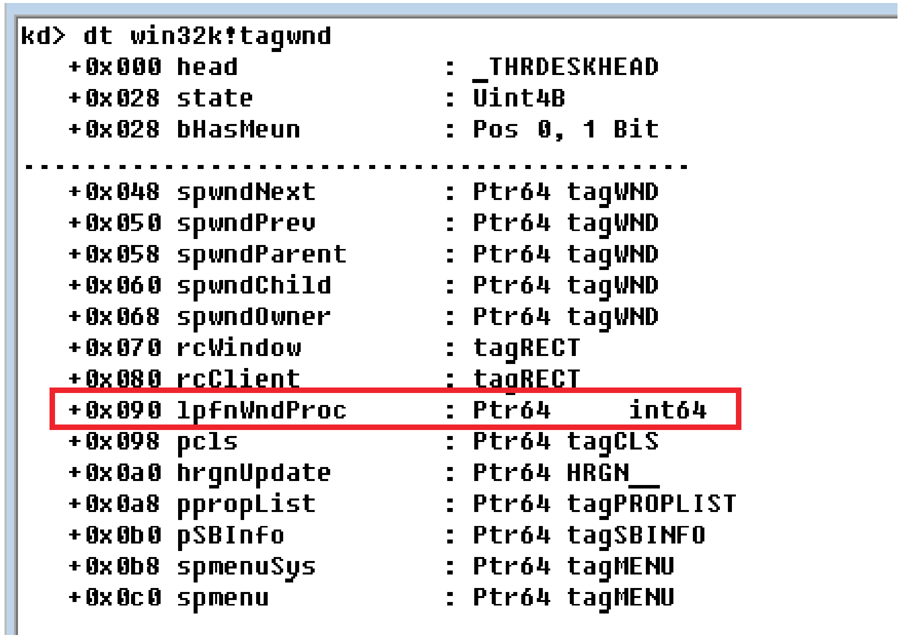


```
signed int __usercall fnPrepareExploitation@<eax>(int Addr@<esi>)
{
  int pWin32ThreadInfoStruct; // ebx@1
  HANDLE v2; // eax@7
  int v4; // [sp+4h] [bp-4h]@7

  pWin32ThreadInfoStruct = fnCallPtiCurrentAndGetTebWin32ThreadInfo();
  if ( !pWin32ThreadInfoStruct )
  {
    GetLastError();
    fnLogMessage("[%d] Failed, %08X\n", 25);
  }
  if ( !ZwAllocateVirtualMemory )
  {
    GetLastError();
    fnLogMessage("[%d] Failed, %08X\n", 31);
  }
  v4 = 0x2000;
  *(_DWORD *)Addr = 1;
  v2 = GetCurrentProcess();
  if ( ZwAllocateVirtualMemory(v2, Addr, 0, &v4, 1060864, 64) )
  {
    GetLastError();
    fnLogMessage("[%d] Failed, %08X\n", 46);
  }
  fnFillMalicioustagWnd(pWin32ThreadInfoStruct);
  return 1;
}
```

Figure 4

In the screenshot above you can see the function in the exploit that generates responses for of setting up exploitation CVE-2014-4113 (Win32/Dianti.A). This is a typical example of the use of memory allocation on a NULL page (32-bit Windows versions). Malicious code allocates memory on this page, where a specially crafted win32k structure called *win32k!tagwnd* is stored. This structure contains a pointer to a special callback function which is called by legitimate kernel mode code during exploitation. It also calls the special function *fnCallPtiCurrentAndGetTebWin32ThreadInfo* to retrieve a legitimate pointer that will used to initialize the malicious *win32k!tagwnd*. The format of this structure is show in the screenshot below (64-bit Windows 7).



```
kd> dt win32k!tagwnd
   +0x000 head             : _THRDESKHEAD
   +0x028 state            : Uint4B
   +0x028 bHasMeun         : Pos 0, 1 Bit
-------------------------------------------
   +0x048 spwndNext        : Ptr64 tagWND
   +0x050 spwndPrev        : Ptr64 tagWND
   +0x058 spwndParent      : Ptr64 tagWND
   +0x060 spwndChild       : Ptr64 tagWND
   +0x068 spwndOwner       : Ptr64 tagWND
   +0x070 rcWindow         : tagRECT
   +0x080 rcClient         : tagRECT
   +0x090 lpfnWndProc      : Ptr64     int64
   +0x098 pcls             : Ptr64 tagCLS
   +0x0a0 hrgnUpdate       : Ptr64 HRGN__
   +0x0a8 ppropList        : Ptr64 tagPROPLIST
   +0x0b0 pSBInfo          : Ptr64 tagSBINFO
   +0x0b8 spmenuSys        : Ptr64 tagMENU
   +0x0c0 spmenu           : Ptr64 tagMENU
```

Figure 5

As we already mentioned in last year's report “Windows exploitation in 2013”, Microsoft has added a security feature in Windows 8 that forbids allocation of memory on the NULL page. For Windows 7 users this security feature became available with the MS13-031 update. Moreover, 64-bit versions of Windows 8 use another security feature called Supervisor Mode Execution Prevention (SMEP). SMEP forbids execution of user-mode memory pages with code from kernel mode.

As we can see from table in the "General Information" section, Microsoft has closed very few vulnerabilities for `win32k.sys` this year. But in any case, this driver contained many vulnerabilities which were closed years ago. This situation was most likely possible because `win32k.sys` code was previously located in user mode DLLs. Before Windows NT 4.0, when this driver was released as part of the Operating System, the whole GUI subsystem was located in special user mode libraries and processes. But in Windows NT 4.0, part of this code was merged into kernel mode and the `win32k.sys` driver. This is why it is possible that it contains so many vulnerabilities, because some snippets of code were taken directly from user mode libraries, perhaps without additional checks.

Internet Explorer

In 2014 Microsoft closed many, many vulnerabilities for their browser. These flaws were of the RCE type and can be used by an attacker for silently installing various types of malware. Moreover, seven of these vulnerabilities were exploited in the wild. The newest versions of Internet Explorer have special f security features that help the user by mitigating these types of attack. Let's look at the details.

First of all, we should mention Enhanced Protected Mode or EPM, which was introduced in Internet Explorer 10 (IE10). EPM is a *full sandbox* for the browser and it can isolate the working browser's tabs from Windows resources, like the application sandboxes that are also present in almost every modern operating system, including Apple's Mac OS X and iOS as well as Google's Android. EPM is supported only on Windows 8+, because it relates to a special Windows kernel mechanism called *AppContainer*. It's easy to understand that AppContainer is an extension of the Integrity Level (IL or *partial sandbox*) mechanism known as Protected Mode, and was introduced in Internet Explorer 7, which was released in 2006. Note that IE10+ EPM relies on kernel features built into Windows, so it is impossible to use this mitigation feature in Windows versions earlier than Windows 8, as AppContainer is not supported in older versions. The one exception is 64-bit Windows 7, where EPM activates 64-bit processes for the browser.

As we have seen in the last two years, attackers can take advantage of DLLs that don't use ASLR to build more resistant exploits or "to bypass ASLR by default". In IE10 Microsoft introduced a special option called ForceASLR. Like EPM, ForceASLR relies on Windows kernel innovations, which are present in Windows 8 by default or in Windows 7 where the special update *KB2639308 has been applied*. This option applies ASLR by default to all DLLs, which are loaded into the context of browser's processes, even where such modules were not compiled with ASLR support (`/DYNAMICBASE` option). This option corresponds to EMET's *Mandatory ASLR* option. We will discuss EMET in more detail in the "Mitigations" section. Actually, you can use the Windows 8+ ForceASLR option to instruct an executable files loader that it should apply the ASLR option by default for all DLLs loaded into a specific process.

Note that *by default* Internet Explorer (including the newest version, IE11) works as 32-bit even on 64-bit systems. You should manually activate the corresponding option (*Enable 64-bit processes for Enhanced Protection Mode*). You can learn more about this option in the posts *Windows Exploitation in 2013* and *Exploit Protection for Microsoft Windows*. Using IE in default settings significantly reduces its resistance to exploitation. In 32-

bit address space it is considerably easier to bypass ASLR with *heap-spray* whereas in 64-bit address space this is almost impossible.

As we can see in Table 3 above and from *our previous report*, use-after-free (UAF) vulnerabilities represent a common type of vulnerability not only for IE, but also for many Windows components. In a normal situation the browser works with memory according to the following steps.

1) Browser allocates block of memory in the memory heap.
2) Browser uses memory block.
3) Browser frees memory block.

If the browser's code contains a UAF vulnerability, normal behavior has changed and after deletion of corresponding block, it is referenced again.

1) Browser allocates block of memory in the memory heap.
2) Browser uses memory block.
3) Browser frees memory block.
4) Browser repeatedly refers to freed block.

So an exploit can use this situation for its own purposes.

1) Browser allocates block of memory in the memory heap.
2) Browser uses memory block.
3) Browser frees memory block.
4) User visits web page with exploit.
5) Exploit performs heap spray (to bypass ASLR) and fills allocated blocks with shellcode or with special addresses that are required to trigger vulnerability.
6) Exploit creates special conditions for browser, to force it to refer to invalid pointer, which already validated by step 4.
7) Browser repeatedly refers to the previously freed and already valid block, and triggers the vulnerability. Next, malicious code gains control, before executing ROP gadgets to bypass DEP, for example, via *ntdll!NtProtectVirtualMemory*.

To protect potentially unsafe browser code from such vulnerabilities, Microsoft introduced a special protective measure called *anti-UAF with isolated heap and deferred heap freeing algorithm*. It's easy to understand, that deferred freeing of heap blocks is an interesting solution, because from the point of view of Windows, the freed block of memory is still occupied, so an attacker will not get to appropriate it for his own use. In Table 6 below you can see the updates that introduced these anti-UAF mitigations. The same applies to the isolated heap, its mission being to isolate allocation of memory for special security-critical browser's objects from the general heap, where attackers can find necessary data faster or just use it effortlessly for triggering a vulnerability.

Innovation	Starting from	Description
Enhanced Protected Mode	IE10	For Windows 7 x64 turns on 64-bit virtual address space for tab's processes. For Windows 8+ x32/x64 turns on *AppContainer* level for sandboxing
ForceASLR	IE10 (Windows 8+ or Windows 7 w/ KB2639308)	Forcibly apply ASLR for all modules loaded by the browser.
64-bit tabs	IE11	For Windows 7+ x64 turns on 64-bit virtual address space for tab's processes
Protected Heap & Delay Free (Anti-UAF)	IE11 (MS14-035, MS14-037)	Introduces mitigation practices for exploits that trigger use-after-free vulnerabilities.
Out-of-date ActiveX control blocking	IE 8-11 on Windows 7 SP1+ Oracle Java & MS Silverlight Plugins (KB2991000)	Blocks loading out-of-date Oracle Java and MS Silverlight plugins for Internet Explorer.

Table 6

Another IE security feature introduced this year is called *Out-of-date ActiveX control blocking*. As is clear from its name, this feature specializes in blocking deprecated versions of various browser plugins. Today, this feature can block out-of-date versions of Oracle Java and Microsoft Silverlight plugins. This feature can block a range of exploits, based on exploitation of old versions of vulnerable plugins. This feature was delivered to customers with the August 2014 Patch Tuesday updates.

Unfortunately, the aforementioned *full sandbox* (EPM) security feature in IE11 can be bypassed with special tricks. As was demonstrated by researchers from Google Zero Team, the browser has weaknesses in its security model. As reported by Google, the *sandbox-escape* vulnerability CVE-2014-6350 (MS14-065) can be used by attackers to bypass this important security feature. The IE sandbox uses measures based on a special *broker process* which can be opened for memory reading operations and, potentially, can be used for *information-disclosure* about internal sandbox data and subsequent exploitation. This so-called broker process can provide special access to Windows resources for sandboxed tabs (processes). Another exploitation method, also mentioned by Google's researchers, is based on a special section object used by all processes in IE's running processes tree for sharing necessary settings between them.

Mitigations

We already mentioned various exploit mitigation techniques above. Now we can discuss another optional security feature and how it can be used to make exploit protection stronger. In 2014 Microsoft released a newer version of its famous security toolkit, EMET (5.1 is the current version). If you are not familiar with this tool, there is information about it on the Microsoft Support site.

In the newer version of EMET, Microsoft introduced some useful security features. These features are called ASR (Attack Surface Reduction) and EAF+ (Export Address Table Filtering Plus). ASR is similar to IE's option called *Out-of-date ActiveX control blocking*, but it can cover a wider range of exploits. If the aforementioned IE option can block loading only of out-of-date plugins into process address space, ASR can block *all* specified modules from loading into the address spaces of the following processes: Microsoft Internet Explorer, Excel, Word and PowerPoint.

By default ASR blocks the loading of modules that often used by attackers for drive-by download into these processes, including Oracle Java plugins (`npjpi*.dll, jp2iexp.dll`), Vector Markup Language DLL (`vgx.dll`) (SA 2963983) and Flash Player (`flash*.ocx`). So, if you use this option, which is enabled by default, for these programs, you can't play the corresponding content. The specific set of blocking modules depends on specific applications. This option works for special *non-trusted IE zones*, so that in the Intranet Zone the user can play appropriate content. In Table 7 below you can see modules blocked by ASR from loading into the context of the corresponding process.

Process	ASR option
Internet Explorer (iexplore.exe)	npjpi*.dll;jp2iexp.dll;vgx.dll; msxml4*.dll; wshom.ocx;scrrun.dll
PowerPoint (powerpnt.exe)	flash*.ocx
Word (winword.exe)	flash*.ocx
Excel (excel.exe)	flash*.ocx

Table 7

A more recent EMET option is called EAF+. It improves on an existing option called EAF and can work independently of the older option. As we know, shellcode from exploits is interested in retrieving the addresses of various exported functions from system modules like `ntdll.dll` during run-time by analyzing the export address table (EAT) of these modules. The usual EAF option blocks read-access attempts by shellcode to see pages of memory where the EAT is located. This is specific to the EATs of `kernel32.dll` and `ntdll.dll`. EAF+ extends this protection and also protects the EAT of `kernelbase.dll`. Moreover, this new option can also mitigate special shellcode techniques, which involve the use of ROP gadgets from libraries known to bypass the original EAF option. This means that an exploit's access to the EAT will be carried out using known code, which is not in itself malicious and represents specific ROP-gadgets from legitimate libraries. EAF+ can control this situation by adding a special feature, which allows EMET to recognize such types of attack. In addition to blocking access to the EAT from unknown code, it also blocks attempts by code from legitimate libraries that could be used by attackers for scanning the EAT. By default, the EAF+ option is turned on for MS Internet Explorer and Adobe Reader processes. In Table 8 below are listed specific modules: their code will be blocked from accessing the EAT of kernel32, ntdll, and kernelbase by EMET.

Process	EAF+ option
Internet Explorer (iexplore.exe)	mshtml.dll; flash*.ocx;jscript*.dll; vbscript.dll; vgx.dll
Adobe Acrobat (acrobat.exe, acrord32.exe)	AcroRd32.dll; Acrofx32.dll; AcroForm.api

Table 8

Actually, in practice, the aforementioned mitigation techniques work with special Windows configurations. In Table 9 below you can see how sometimes it's hard to understand which option could be useful against a specific attack. We will look at the CVE-2014-1776 vulnerability as example.

Option	Effective?	Details
Enhanced Protected Mode IE10 & IE11 (EPM) on Windows 7 x32	No	Useless by design.
Enhanced Protected Mode IE10 & IE11 (EPM) on Windows 8+ x32	No	Isolation of AppContainer sandbox is insufficient to block an exploit actions (only in conjunction with 64-bit tabs)
Enhanced Protected Mode IE10 (EPM) on Windows 8+ x64	No	IE10 doesn't contain option of 64-bit tabs.
Enhanced Protected Mode IE10 (EPM) on Windows 7 x64	Yes	Instead of AppContainer, turns on 64-bit virtual address space for browser tabs.

Option	Effective?	Details
Enhanced Protected Mode IE10 (EPM) on Windows 7+ x64 & 64-bit tabs	Yes	Used by malware ActionScript-heap-spray from Flash Player object is not effective in 64-bit address space
EMET 5 ASR	Yes	Blocks loading of VGX.DLL & Flash (.ocx) into running IE process for «Internet» zone.
EMET 5 EAF+	Yes	Does not allow shellcode to get access to memory page with ntdll.dll exports.
EMET 5, 4.x Heap Spray	Yes	Blocks ActionScript-heap-spray method used by exploit (ASLR bypass).
EMET 5, 4.x ROP (StackPivot, Caller, MemProt)	Yes	Only with turned on option «Deep hooks».

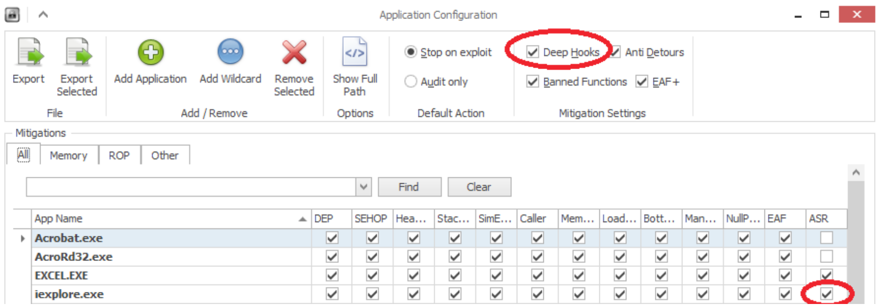
Table 9

Another interesting EMET 5.1 option activated by default is *Deep Hooks*. For example, this option was useful for blocking the notorious 0day exploit of the CVE-2014-1776 vulnerability. More details are given in the post *More Details about Security Advisory 2963983 IE 0day*. The *Deep Hooks* option allows EMET deeper monitoring of various operations by additional hooking of the Windows API & Internal API (ntdll). You can find example of such hooks in Table 10 below.

Deep Hooks OFF		Deep Hooks ON
Memory functions	kernel32!VirtualAllocw	ntdll!NtAllocateVirtualMemory
		kernelbase!VirtualAlloc
	kernel32!VirtualAllocEx	ntdll!NtAllocateVirtualMemory
		kernelbase!VirtualAllocEx
	kernel32!VirtualProtect	ntdll!NtProtectVirtualMemory
		kernelbase!VirtualProtect
	kernel32!VirtualProtectEx	ntdll!NtProtectVirtualMemory
		kernelbase!VirtualProtectEx

Table 10

As we can see from Table 10, the *Deep Hooks* option turns on hooking of lower level Windows functions that called from the controlled API. For example, if this option is turned on, EMET will control not only the *kernel32!VirtualAlloc* API, but also the functions *ntdll!NtAllocateVirtualMemory* and *kernelbase!VirtualAlloc*. The screenshot below shows these EMET security features in its GUI interface.



● Stop on exploit
○ Audit only

☑ Deep Hooks ☑ Anti Detours
☑ Banned Functions ☑ EAF+

App Name	DEP	SEHOP	Hea…	Stac…	SimE…	Caller	Mem…	Load…	Bott…	Man…	NullP…	EAF	ASR
Acrobat.exe	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐
AcroRd32.exe	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐
EXCEL.EXE	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
iexplore.exe	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

Figure 6

Instead of conclusion

Ultimately, the improved mitigations against RCE-exploits lead to increases in the cost of exploit development. This is shown in the scheme below. Attackers need more money and time for investigating new vulnerabilities that can help to bypass improved anti-exploit security features. As we have shown in this report, today, they need a set of two or even three exploits to penetrate into the system and get full control under computer.

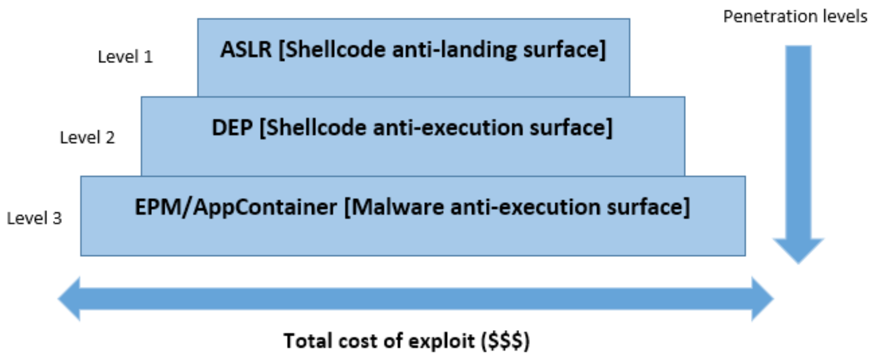
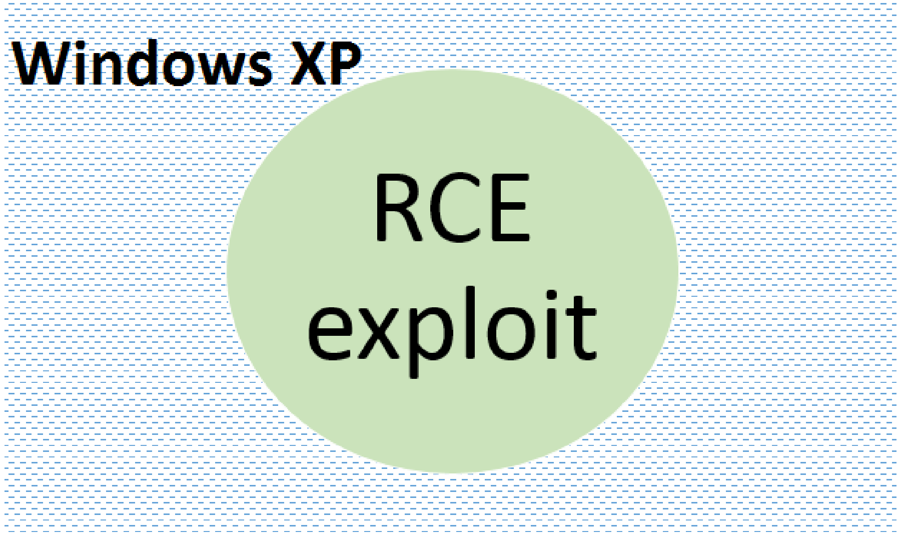
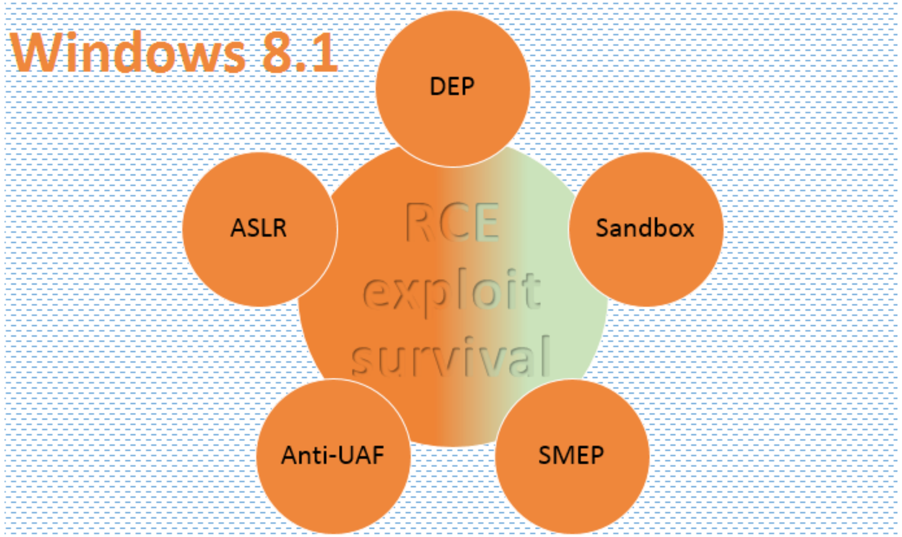

Penetration levels
Level 1
ASLR [Shellcode anti-landing surface]
Level 2
DEP [Shellcode anti-execution surface]
Level 3
EPM/AppContainer [Malware anti-execution surface]
Total cost of exploit ($$$)

Figure 7

Unfortunately, many users still use unsafe versions of Windows, like Windows XP. These Windows versions do not contain the modern anti-exploit security features described here and the user should understand that the use of outmoded versions exposes his system to a significant risk. In the figures below you can see how RCE exploits compare under Windows XP and Windows 8.1. It's enough to say that the user should think at least twice before continuing to use Windows XP.





We can predict for next year that drive-by download attacks will remain as the main avenue for exploiting vulnerabilities and delivering malicious code. Due to the significant and increasing complexity of exploit development, we also can predict that such exploits will continue to be developed by specialist engineers for use in targeted attacks.

Baranov Artem
Malware researcher, ESET Russia