How not to use a driver to execute code with kernel privileges

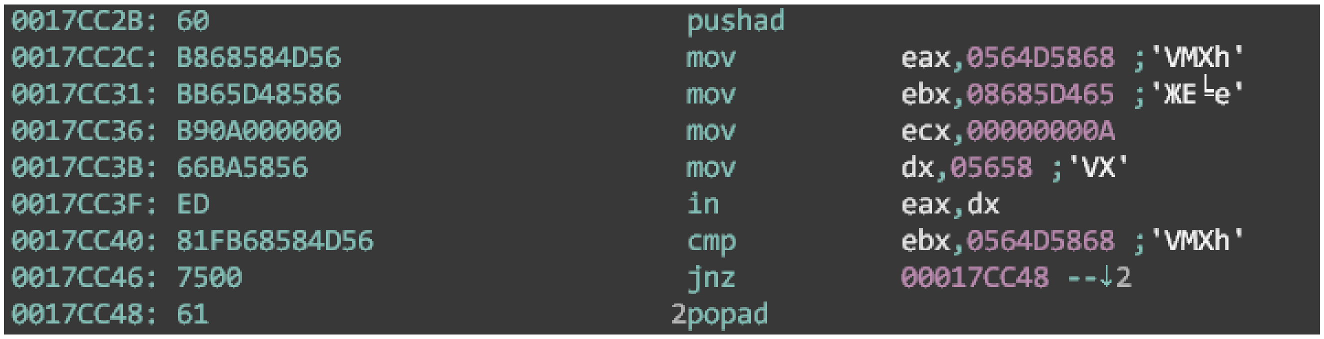
securelist.com/elevation-of-privileges-in-namco-driver/83707/

Authors

- Expert Vladislav Stolyarov
- Expert Boris Larin

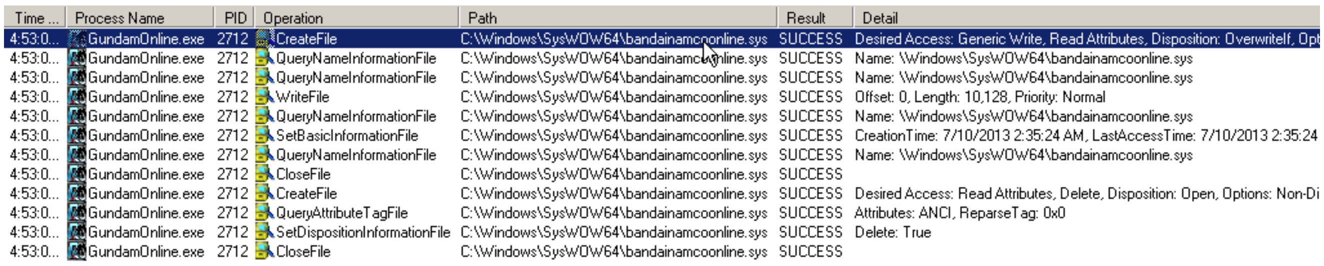
Recently, we started receiving suspicious events from our internal sandbox Exploit Checker plugin. Our heuristics for supervisor mode code execution in the user address space were constantly being triggered, and an executable file was being flagged for further analysis. At first, it looked like we'd found a zero-day local privilege escalation vulnerability for Windows, but the sample that was triggering Exploit Checker events turned out to be the clean signed executable GundamOnline.exe, part of the multiplayer online game Mobile Suit Gundam Online from BANDAI NAMCO Online Inc.

The initial sample is packed using a custom packer and contains anti-analysis techniques that complicate static analysis. For example, it tries to detect if it's being launched inside a virtual machine by performing a well-known VMware hypervisor detection routine. It first loads the EAX register with the hypervisor magic value VMXh, and the ECX register with the value 0x0A, which is a special command to receive the hypervisor version. Then it performs an 'in' command to the VMware hypervisor IO port 0x5658. If the EBX register is overwritten with VMXh as a result of that operation, it means the executable file is running on the VMware machine.



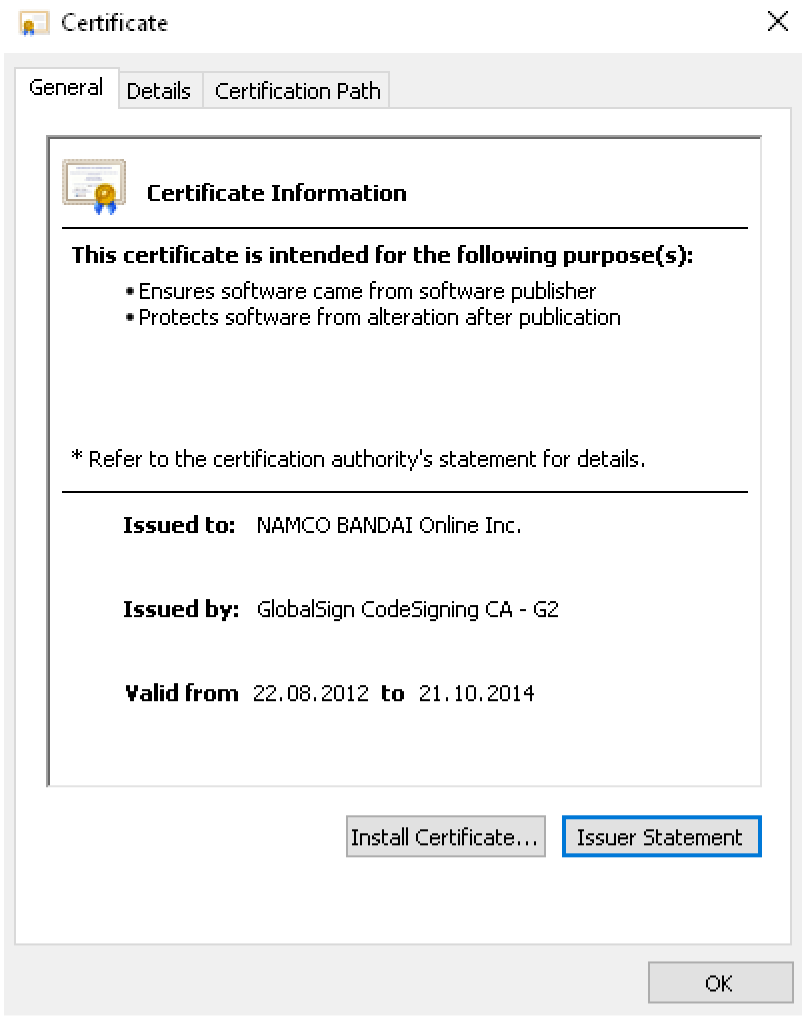
```
0017CC2B: 60                      pushad
0017CC2C: B868584D56              mov        eax,0564D5868 ;'VMXh'
0017CC31: BB65D48586              mov        ebx,08685D465 ;'ЖЕ╘e'
0017CC36: B90A000000              mov        ecx,00000000A
0017CC3B: 66BA5856                mov        dx,05658 ;'VX'
0017CC3F: ED                      in         eax,dx
0017CC40: 81FB68584D56            cmp        ebx,0564D5868 ;'VMXh'
0017CC46: 7500                    jnz        00017CC48 --↓2
0017CC48: 61                     2popad
```

Our sandbox execution logs showed that the user space memory page is called from the driver bandainamcoonline.sys immediately after IOCTL request 0xAA012044 to device object \.Htsysm7838 that is created by the driver. The driver itself is installed just before that. It is first dropped to the directory C:WindowsSysWOW64 by a GundamOnline executable, loaded using NtLoadDriver() and deleted immediately afterwards.



Time ...	Process Name	PID	Operation	Path	Result	Detail
4:53:0...	GundamOnline.exe	2712	CreateFile	C:\Windows\SysWOW64\bandainamcoonline.sys	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: OverwriteIf, Opt
4:53:0...	GundamOnline.exe	2712	QueryNameInformationFile	C:\Windows\SysWOW64\bandainamcoonline.sys	SUCCESS	Name: \Windows\SysWOW64\bandainamcoonline.sys
4:53:0...	GundamOnline.exe	2712	QueryNameInformationFile	C:\Windows\SysWOW64\bandainamcoonline.sys	SUCCESS	Name: \Windows\SysWOW64\bandainamcoonline.sys
4:53:0...	GundamOnline.exe	2712	WriteFile	C:\Windows\SysWOW64\bandainamcoonline.sys	SUCCESS	Offset: 0, Length: 10,128, Priority: Normal
4:53:0...	GundamOnline.exe	2712	QueryNameInformationFile	C:\Windows\SysWOW64\bandainamcoonline.sys	SUCCESS	Name: \Windows\SysWOW64\bandainamcoonline.sys
4:53:0...	GundamOnline.exe	2712	SetBasicInformationFile	C:\Windows\SysWOW64\bandainamcoonline.sys	SUCCESS	CreationTime: 7/10/2013 2:35:24 AM, LastAccessTime: 7/10/2013 2:35:24
4:53:0...	GundamOnline.exe	2712	QueryNameInformationFile	C:\Windows\SysWOW64\bandainamcoonline.sys	SUCCESS	Name: \Windows\SysWOW64\bandainamcoonline.sys
4:53:0...	GundamOnline.exe	2712	CloseFile	C:\Windows\SysWOW64\bandainamcoonline.sys	SUCCESS	
4:53:0...	GundamOnline.exe	2712	CreateFile	C:\Windows\SysWOW64\bandainamcoonline.sys	SUCCESS	Desired Access: Read Attributes, Delete, Disposition: Open, Options: Non-Di
4:53:0...	GundamOnline.exe	2712	QueryAttributeTagFile	C:\Windows\SysWOW64\bandainamcoonline.sys	SUCCESS	Attributes: ANCI, ReparseTag: 0x0
4:53:0...	GundamOnline.exe	2712	SetDispositionInformationFile	C:\Windows\SysWOW64\bandainamcoonline.sys	SUCCESS	Delete: True
4:53:0...	GundamOnline.exe	2712	CloseFile	C:\Windows\SysWOW64\bandainamcoonline.sys	SUCCESS	

Normally, this kind of behavior should not be allowed due to SMEP (Supervisor Mode Execution Prevention). This is a security feature present on the latest Intel processors that restricts supervisor mode execution on user memory pages. Page type is determined using the User/Supervisor flag in the page table entry. If a user memory page is called while in supervisor execution mode, SMEP generates an access violation exception and, as a result, the system will trigger a bug check and halt. This is commonly referred to as a BSOD.



Certificate Information

This certificate is intended for the following purpose(s):

- Ensures software came from software publisher
- Protects software from alteration after publication

* Refer to the certification authority's statement for details.

Issued to: NAMCO BANDAI Online Inc.

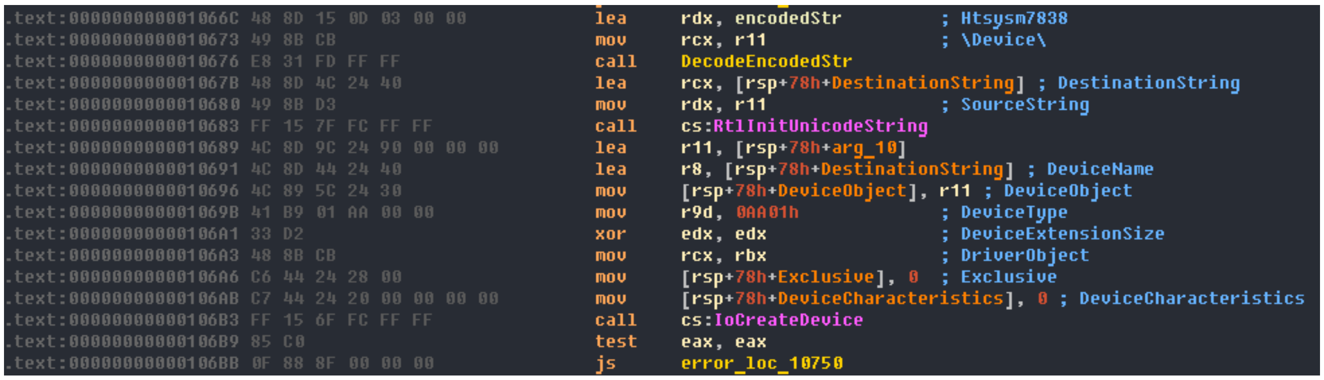
Issued by: GlobalSign CodeSigning CA - G2

Valid from 22.08.2012 to 21.10.2014

The dropped driver itself is a legitimate driver, signed with a certificate issued to NAMCO BANDAI Online Inc.

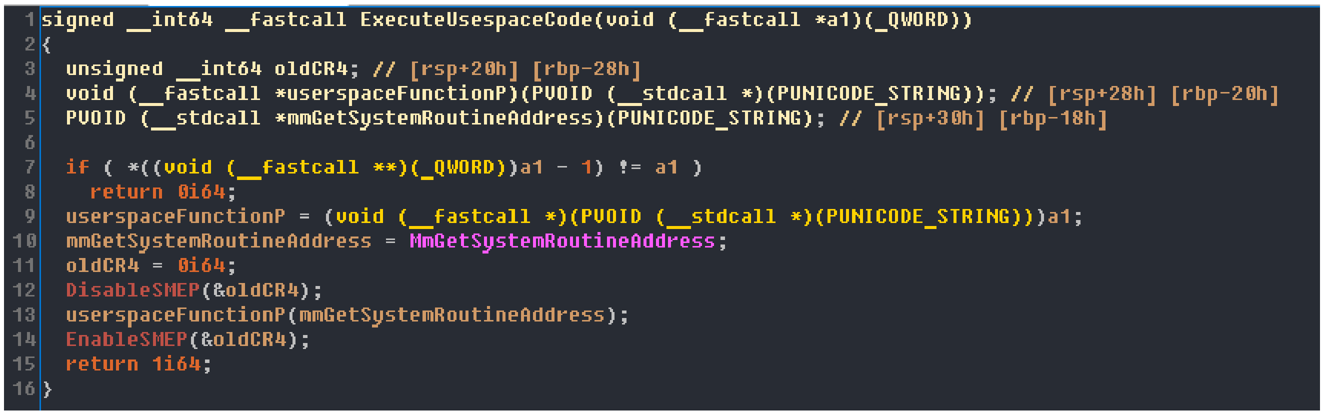
The certificate validity period tells us two things. First, this certificate has been valid since 2012, which could mean that the first vulnerable version of the driver was released around the same time. However, we were unable to find one; the earliest sample of bandainamcoonline.sys that we found dates back to November 2015. Secondly, because it expired more than three years ago, you could be forgiven for thinking it's impossible to install a driver signed with this certificate in a system. Actually, there's nothing stopping you from installing and loading a driver with an expired certificate validity period.

In order to find the cause of the heuristics trigger, we need to do a static analysis of the driver itself. In the DriverEntry function it first decodes the device object name string in memory, and then creates the device \.Htsysm7838. The other two encoded strings – bandainamcoonline and bandainamcoonline.sys – are not used in the driver.



```
.text:000000000001066C 48 8D 15 0D 03 00 00          lea     rdx, encodedStr          ; Htsysm7838
.text:0000000000010673 49 8B CB                      mov     rcx, r11                 ; \Device\
.text:0000000000010676 E8 31 FD FF FF                call    DecodeEncodedStr
.text:000000000001067B 48 8D 4C 24 40                lea     rcx, [rsp+78h+DestinationString] ; DestinationString
.text:0000000000010680 49 8B D3                      mov     rdx, r11                 ; SourceString
.text:0000000000010683 FF 15 7F FC FF FF             call    cs:RtlInitUnicodeString
.text:0000000000010689 4C 8D 9C 24 90 00 00 00       lea     r11, [rsp+78h+arg_10]
.text:0000000000010691 4C 8D 44 24 40                lea     r8, [rsp+78h+DestinationString] ; DeviceName
.text:0000000000010696 4C 89 5C 24 30                mov     [rsp+78h+DeviceObject], r11 ; DeviceObject
.text:000000000001069B 41 B9 01 AA 00 00             mov     r9d, 0AA01h              ; DeviceType
.text:00000000000106A1 33 D2                         xor     edx, edx                 ; DeviceExtensionSize
.text:00000000000106A3 48 8B CB                      mov     rcx, rbx                 ; DriverObject
.text:00000000000106A6 C6 44 24 28 00                mov     [rsp+78h+Exclusive], 0   ; Exclusive
.text:00000000000106AB C7 44 24 20 00 00 00 00       mov     [rsp+78h+DeviceCharacteristics], 0 ; DeviceCharacteristics
.text:00000000000106B3 FF 15 6F FC FF FF             call    cs:IoCreateDevice
.text:00000000000106B9 85 C0                         test    eax, eax
.text:00000000000106BB 0F 88 8F 00 00 00             js      error_loc_10750
```

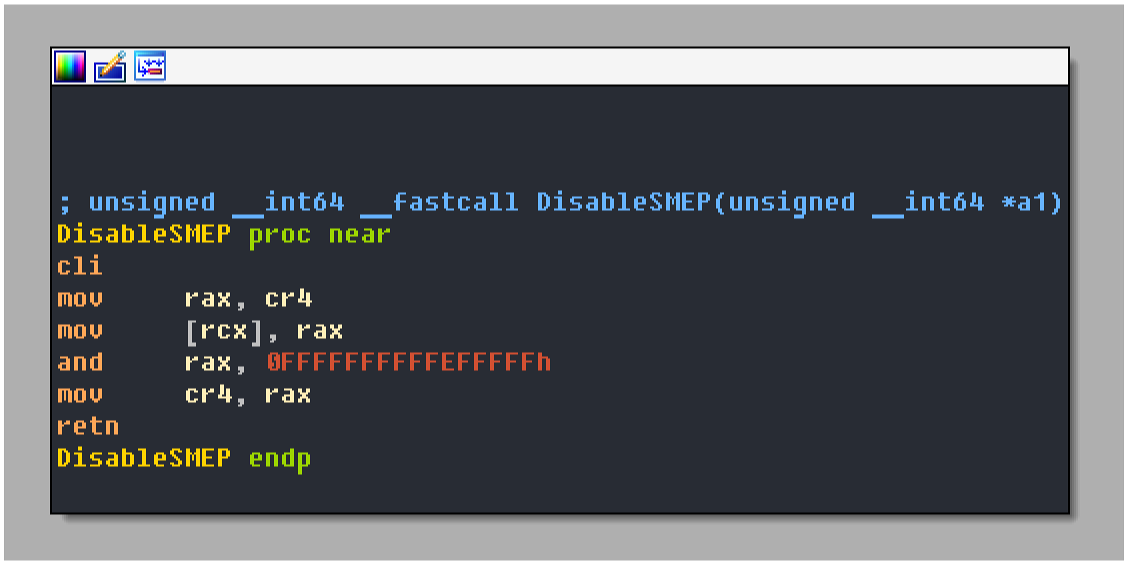
The driver itself is very small and contains only three registered major functions. Function IRP_MJ_DEVICE_CONTROL, which handles requests, accepts only two IOCTLs: 0xAA012044 and 0xAA013044. When called, it checks the size of the input and output buffers and eventually calls the ExecuteUserspaceCode function, passing on the contents of the input buffer to it.



```
signed __int64 __fastcall ExecuteUsespaceCode(void (__fastcall *a1)(_QWORD))
{
  unsigned __int64 oldCR4; // [rsp+20h] [rbp-28h]
  void (__fastcall *userspaceFunctionP)(PVOID (__stdcall *)(PUNICODE_STRING)); // [rsp+28h] [rbp-20h]
  PVOID (__stdcall *mmGetSystemRoutineAddress)(PUNICODE_STRING); // [rsp+30h] [rbp-18h]

  if ( *((void (__fastcall **)(_QWORD))a1 - 1) != a1 )
    return 0i64;
  userspaceFunctionP = (void (__fastcall *)(PVOID (__stdcall *)(PUNICODE_STRING)))a1;
  mmGetSystemRoutineAddress = MmGetSystemRoutineAddress;
  oldCR4 = 0i64;
  DisableSMEP(&oldCR4);
  userspaceFunctionP(mmGetSystemRoutineAddress);
  EnableSMEP(&oldCR4);
  return 1i64;
}
```

The function ExecuteUserspaceCode performs a single check on the input buffer, which contains a pointer to a user space function or a shellcode, and disables SMEP while saving old CR4 register values. It then calls that function, passing it a pointer to the MmGetSystemRoutineAddress as an argument. After that it restores the original register state, re-enabling SMEP.

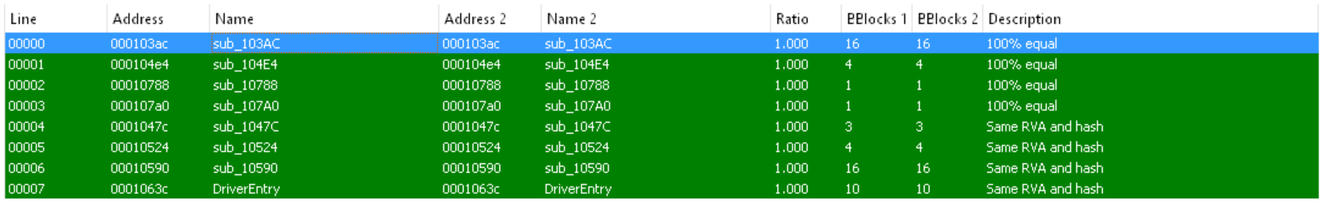


```
; unsigned __int64 __fastcall DisableSMEP(unsigned __int64 *a1)
DisableSMEP proc near
cli
mov     rax, cr4
mov     [rcx], rax
and     rax, 0FFFFFFFFFFEFFFFFh
mov     cr4, rax
retn
DisableSMEP endp
```

To be able to directly call the user function from the provided pointer driver it is necessary to remove a specific bit in the CR4 register first to temporarily stop SMEP, which is what the DisableSMEP function does. The original CR4 values are then restored by the EnableSMEP function.

The vulnerability in this case is that other than the basic checks on the format of the input buffer, no additional checks are done. Therefore, any user on the system can use this driver to elevate their privileges and execute arbitrary code in the Ring 0 of the OS. Even if the driver is not present in the system, an attacker can register it with Windows API functions and exploit the flaw.

We realized that this vulnerability looks exactly like the one found in Capcom's driver last year.



Line	Address	Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Description
00000	000103ac	sub_103AC	000103ac	sub_103AC	1.000	16	16	100% equal
00001	000104e4	sub_104E4	000104e4	sub_104E4	1.000	4	4	100% equal
00002	00010788	sub_10788	00010788	sub_10788	1.000	1	1	100% equal
00003	000107a0	sub_107A0	000107a0	sub_107A0	1.000	1	1	100% equal
00004	0001047c	sub_1047C	0001047c	sub_1047C	1.000	3	3	Same RVA and hash
00005	00010524	sub_10524	00010524	sub_10524	1.000	4	4	Same RVA and hash
00006	00010590	sub_10590	00010590	sub_10590	1.000	16	16	Same RVA and hash
00007	0001063c	DriverEntry	0001063c	DriverEntry	1.000	10	10	Same RVA and hash

Binary diffing bandainamcoonline.sys and capcom.sys proves exactly that, showing there are almost no differences between the two drivers. The only slight variations are the encoded strings and digital signatures. Because the earliest sample of the vulnerable driver that we've been able to find dates to November 2015, it can be assumed that this vulnerability first appeared in the bandainamcoonline.sys driver – almost a year before a similar driver was used by Capcom.

We believe both drivers were almost certainly compiled from the same source code, as a part of an anti-hacking solution to prevent users from cheating in the game. The presence of functions that implicitly disable and re-enable SMEP show that this design decision was intentional. But because the driver makes no additional security checks, any user can call and exploit the vulnerable IO control code by using Windows APIs such as DeviceIoControl(). This essentially makes the driver a rootkit, allowing anyone to interact with the operating system at the highest privilege level. In fact, we found multiple malware samples (already detected by our products) using a previously known vulnerability in capcom.sys to elevate their privileges to System level.

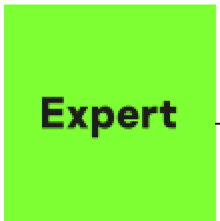
After finding the vulnerability we contacted BANDAI NAMCO Online Inc. The vendor responded promptly and released a patch three days later. They removed the driver altogether, and it is no longer loaded by the game executable. This is very similar to what Capcom did, and is perfectly acceptable in this case.

Finding this vulnerability wouldn't have been possible without our Exploit Checker technology, which is a plugin for our sandbox, and can be also found in KATA (Kaspersky Anti Targeted Attack Platform). The technology was designed to monitor suspicious events that occur at the earliest post-exploitation phases and can detect common techniques used in exploits, such as ROP, Heap Spray, Stack Pivot, and so on. In this particular case, multiple heuristics for executing code in supervisor mode in the user address space were triggered, and the sample was flagged for further analysis. If a token-swapping attempt was performed to elevate process privileges, a technique that's widely used in LPE exploits, it would have been automatically detected by Exploit Checker heuristics.

Kaspersky Lab solutions detect the vulnerable drivers mentioned in this article as HEUR:HackTool.Win32.Banco.a and HEUR:HackTool.Win32.Capco.a.

- Drivers
- Virtualization
- Vulnerabilities

Authors

- 

 Vladislav Stolyarov
-

 Boris Larin

A vulnerable driver: lesson almost learned