



Using a Patched Vulnerability to Bypass Windows 8 x64 Driver Signature Enforcement

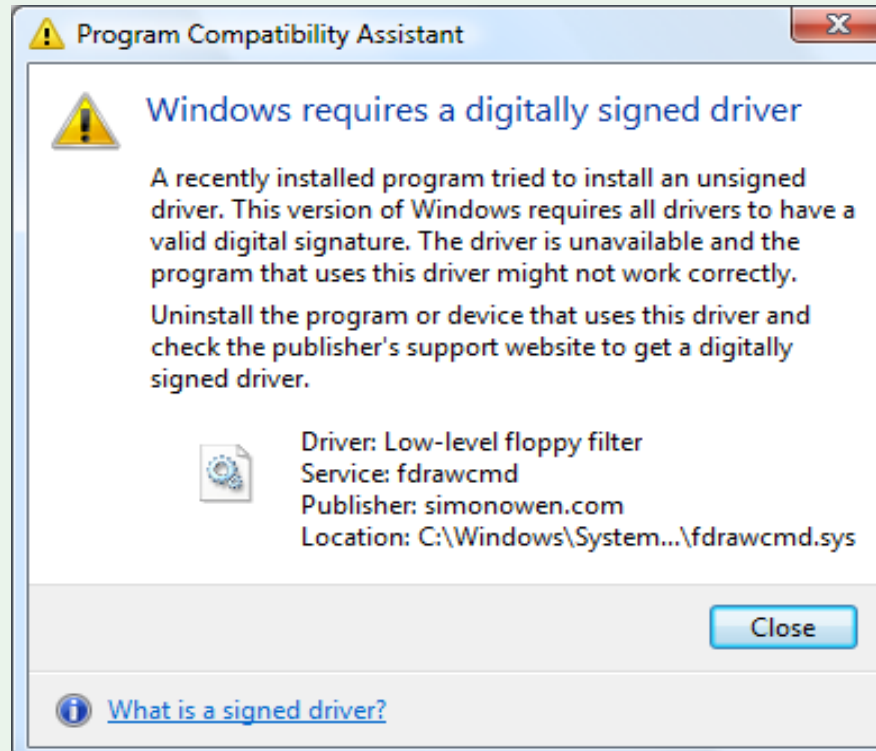
MJ0011

th_decoder@126.com

- Background
- A Patched Vulnerability: CVE-2010-4398
- Bypass DSE on Windows7 x64
- Windows8 Kernel Security Improvements
- Bypass DSE on Windows8 x64

- x64 Driver Signature Enforcement(DSE)
- A new feature introduced from Vista x64, which requires a special Kernel Mode Code Signing(KMCS) in order to load kernel-mode driver even though administrator privilege.
- x64 OS kernel will use MiValidateImagePages and ci.dll's callbacks to check whether kernel-mode driver has KMCS.
- If you try to load any kernel-mode module which is not certified by KMCS, the loading will be refused.

- If an unsigned driver tries to install, it will pop-up window tips.



- Developers must purchase a Class 3 Commercial Software Publisher Certificate from VeriSign.
- It costs \$500 per year, and is only available to commercial entities.

CERTIFICATE NAME	DIGICERT'S CODE SIGNING	STARTCOM'S CODE SIGNING	GLOBALSIGN'S CODE SIGNING CERTIFICATE	SYMANTEC'S CODE SIGNING CERTIFICATE
BUY NOW	Buy Now	Buy Now	Buy Now	Buy Now
CERTIFICATE AUTHORITY	 DigiCert	 StartCom	 GlobalSign	 Symantec
CERTIFICATE AUTHORITY RATING	 From 362 reviews	 From 163 reviews	 From 76 reviews	 From 11 reviews
PRICE FOR 1 YEAR	\$223	\$199	\$229	\$499
PRICE FOR 2 YEARS	\$397.00	\$199.90	\$422.00	\$895.00
 SUPPORTS WINDOWS AUTHENTICODE SIGNING				

- Known x64 DSE bypass method in the past:
 - Press F8 button during system boot and choose "Disable Driver Signature Enforcement"
 - It's messy to press F8 and choose it every time
 - Ugly watermark will be displayed on the desktop
 - Bypass DSE with modified MBR
 - Modify MBR at risk. It may introduce malicious software
 - Not compatible with Secure Boot in Windows8
 - Need to restart to be effective
 - Using kernel 0day vulnerability to bypass DSE (eg. CVE-2012-0217)
 - No public stable exploit code
 - Microsoft will fix kernel vulnerabilities quickly.

- I want to show a better way to bypass x64 DSE. It demands:
 - Bypass automatically. No need to manually set.
 - More safer. Without modify MBR or other boot stuff.
 - Effective immediately without reboot.
 - Do not use 0day vulnerabilities.
 - Support Windows7 and Windows8 with fully patched.

- Target: A patched kernel vulnerability : CVE-2010-4398
 - Driver Improper Interaction with Windows Kernel Vulnerability
- All Microsoft Windows NT operating systems are affected from Windows NT4 to Windows 7 and Server 2008 R2. The code to cause this vulnerability has been written in 20 years ago(1992).
- After I reported this vulnerability to Microsoft in 2010, Microsoft released the security bulletin (MS11-011) in Mar 2011 to fix it.

Other Information

Acknowledgments

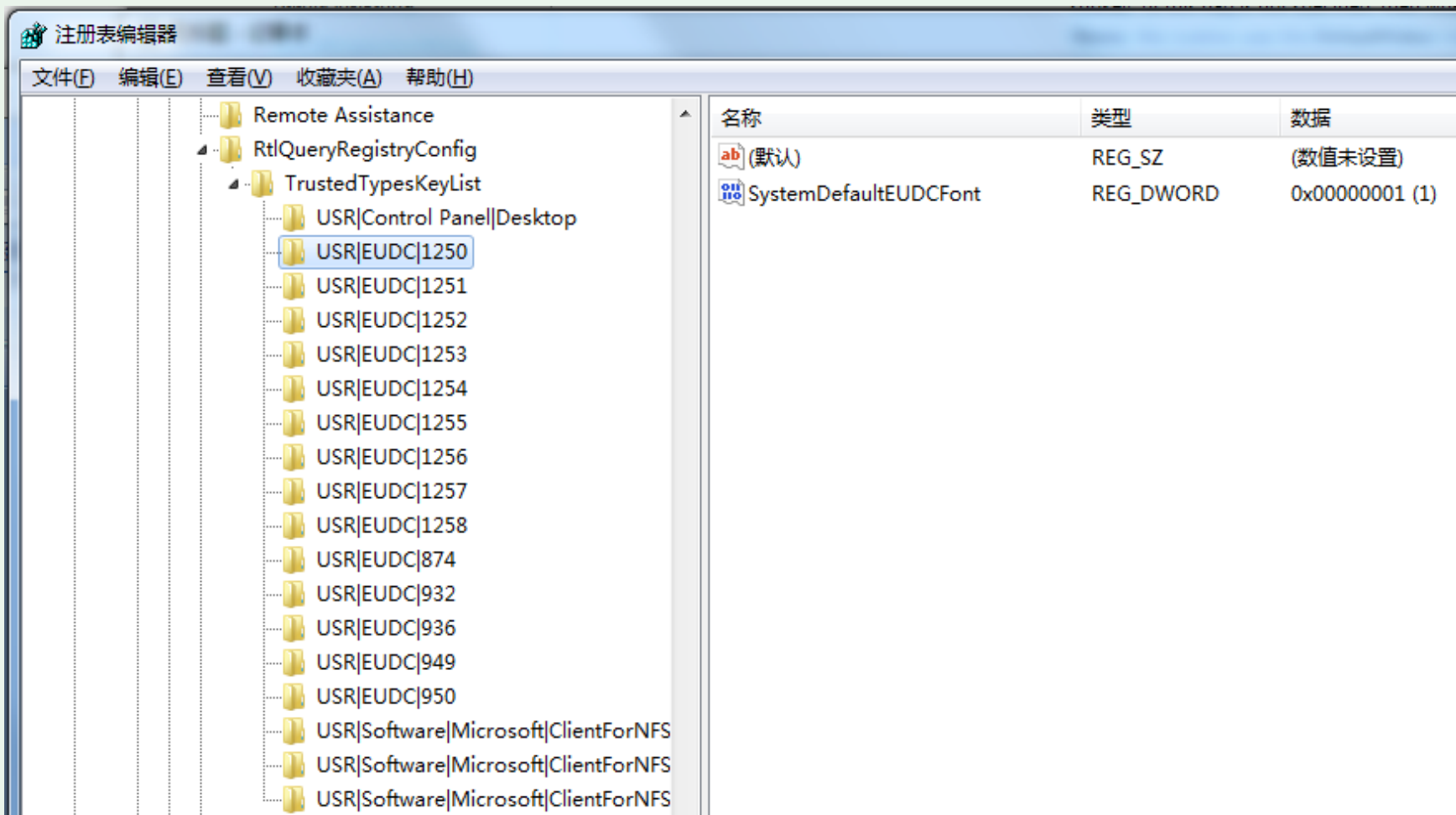
Microsoft [thanks](#) the following for working with us to help protect customers:

- Zhengwenbin of [360safe](#) for reporting the Driver Improper Interaction with Windows Kernel Vulnerability (CVE-2010-4398)

- Formation conditions of CVE-2010-4398:
 - Use RTL_QUERY_REGISTRY_DIRECT flag.
 - Not initialize EntryContext according to key value type.
- The root cause of CVE-2010-4398:
 - When using RTL_QUERY_REGISTRY_DIRECT flag, RtlQueryRegistryValues copy data into EntryContext according to key value type.
 - According to key value type, EntryContext need to set in different buffer length. The problem is that key value type in registry can be changed
 - If the given key value type is unexpected and the expected buffer length is too small, it will cause stack overflow.

- Starting with MS11-011, Microsoft has gradually fix this vulnerability:
 - Use *HKLM\SYSTEM\CurrentControlSet\Control\RtlQueryRegistryConfig\TrustedTypesKeyList* to control read operations of registry which using by known improper interaction drivers
 - New drivers can use RTL_QUERY_REGISTRY_TYPECHECK flag to mandatory RtlQueryRegistryValues to do the type check. If value type does not match, it will return a failure (mainly used in Windows8).
 - Initialize EntryContext properly, eg. if expected value type is REG_DWORD then initialize it to 0.
 - Add security cookie check in functions which calling RtlQueryRegistryValues(mainly used in Windows8).

TrustedTypesKeyList on Windows7 x64



The screenshot shows the Windows Registry Editor (注册表编辑器) window. The left pane displays the tree structure, with the path `Remote Assistance > RtlQueryRegistryConfig > TrustedTypesKeyList > USR|EUDC|1250` selected. The right pane shows a list of registry values:

名称	类型	数据
ab (默认)	REG_SZ	(数值未设置)
SystemDefaultEUDCFont	REG_DWORD	0x00000001 (1)

- The situation of CVE-2010-4398 on Windows7 with fully patched
 - The TrustedTypesKeyList config only protect user-specific areas of registry, but sensitive areas of registry is unprotect.
 - The TrustedTypesKeyList config can be deleted by Admin user
 - The Windows7 with fully patched has fixed some known positions which can trigger this vulnerability under admin privileges
 - nt!RtlQueryTimezoneInformation
 - win32k!SetDPISettings
 - win32k!bReadUserSystemEUDCRegistry ...

But, because of cost considerations, there are a lot of drivers with improper interaction problem are unfixed
- The target driver to bypass Windows7 x64 DSE: AppID.sys

- AppID.sys: It is the kernel part of Application Identity service, and it helps AppLocker to block specified program.
- AppID!AipReadConfigOption, uses RtlQueryRegistryValues to read registry key values from:
HKLM\SYSTEM\CurrentControlSet\Control\AppID. It has improper interaction problem.
- AppID.sys reserves REG_DWORD length of buffer for the key value data. If the data length exceeds, it will cause stack overflow.
- AipReadConfigOptions can be triggered in the context of caller process by DeviceIoControl, and this function has no security cookie checks. So we can directly overwrite the return address by stack overflow and jump to shell code.

- The complete process of bypass Windows7x64 DSE:
 - The AppID device is set to a specific security descriptor, only LocalService account's process can open it. Here I duplicate the token from AppIDSvc service's process, use duplicated token to create new process and open the AppID device.
 - Set Control\AppID\EnablePath key value data with enough length in REG_BINARY format, and overwrite the return address.
 - Send control code 0x22A010 to AppID device. It will trigger AipReadConfigOptions.
 - Jump to shell code, and modify nt!g_CiEnabled to 0, then disable x64 DSE.
- Demo source code: <http://code.google.com/p/bypass-x64-dse>

- Kernel Security Improvements on Windows8:
 - Fixed improper interaction problem for all the Win8 drivers
 - Now there are initialize EntryContext according to key value type properly, and add security cookie check for every function which calling RtlQueryRegistryValues.
 - Introducing the new RtlQueryRegistryValuesEx function.
 - Windows8 drivers use this new function as much as possible. If driver calls new function and the registry key is untrusted, it would cause BugCheck = KERNEL_SECURITY_CHECK_FAILURE.
 - Enhanced kernel's security cookie mechanism
 - It makes forging kernel security cookie becomes almost impossible.
 - Add SMEP and non-executable non-paged pool supports.

- In the past, kernel security cookie could be forecasted:
 - *J00ru: "Windows Kernel-mode GS Cookies subverted"*
 - The paper introduced a way to using module load time to forecast security cookie, bypass security cookie check and exploit CVE-2010-4398. The success rate of more than 46%.
- Windows8 enhances kernel security cookie mechanism:
 - OS loader tries 5 ways to get high entropy source.
 - It includes external entropy, TPM entropy, clock entropy, ACPI entropy and RDRAND entropy.
 - The RDRAND entropy is generated by Intel Secure Key technology, using new "RDRAND" CPU instruction to generate high quality entropy through hardware.
- It is difficult to forecast security cookie on Windows8 x64.

- SMEP: Supervisor-Mode Execution Prevention.
- The new feature introduced in Intel 3rd generation Core processor - Ivy Bridge: After enabling SMEP, it could allows pages to be protected from supervisor mode instruction fetches.
- Windows8 enables SMEP by default:(cr4.SMEP = 1), When kernel code running under Ring0 directly jump to code in ring3, it will cause Bug Check = ATTEMPTED_EXECUTE_OF_NONEXECUTE_MEMORY.
- The most of kernel vulnerability attacks use some trick to make kernel code jump to preset shell code which is placed in user address space
- So most of kernel exploits(included the exploit for CVE-2010-4398) can not use directly on Windows8.

- *Artem Shishkin . Intel SMEP overview and partial bypass on Windows 8*
 - This paper mentioned a way to manipulate data in win32k object , and get the address of win32k object to bypass SMEP.
- It only can use on Windows8 x86, dose not work on Windows8 x64
- Paged session pool used in win32k is non-executable on Windows8 x64
- Combined with the newly introduced and widely used 'NonpagedPoolNx' type pool and non-executable paged pool , bypassing SMEP with manipulate kernel data can not work on Windows8 x64

- To use CVE-2010-4398 to bypass DSE on Windows8 x64 , we need to solve these problems:
 - Find driver with improper interaction problem which accords with the following conditions:
 - Do not initialize EntryContext properly
 - The function which calls RtlQueryRegistryValues does not be protected by security cookie.
 - Do not use RTL_QUERY_REGISTRY_TYPECHECK flag.
 - Bypass SMEP

- It looks like difficult to find suitable drivers:
I manually reviewed all the drivers on Windows8 x64 which calling RtlQueryRegistryValues(Ex) function. But I did not find matching driver. All the drivers on Windows8 did the right interaction.
- Using CVE-2010-4398 to Windows8 x64 DSE is impossible?
- No! With administrator privileges, we can reuse signed driver in Windows7.
- My method: to find a driver which has improper interaction problem on Windows7. If this driver has embedded KMCS, it can be load on Windows8 x64!
- Finally, I find the matched driver: mountmgr.sys.

- mountmgr.sys: Mount point manager driver which handles volume assignments.
- It has embedded KMCS which can be verified on Windows8.
- Mountmgr!DriverEntry->MountmgrReadNoAutoMount function has RtlQueryRegistryValues improper interaction problem, and it has no security cookie check.
- We can unload the new mountmgr.sys with NtLoadDriver API , and reload the old mountmgr.sys with NtLoadDriver . Then triggers stack overflow.
- It reads registry key value from:
HKLM\System\CurrentControlSet\Services\Mountmgr\NoAutoMount


- Bypass SMEP: use ROP (Return-Oriented Programming)
- *Artem Shishkin & Ilya Smit: Bypassing Intel SMEP on Windows 8 x64 Using Return-oriented Programming(2012/9/19)*
 - This paper mentioned a way to set ROP chain and modifying cr4.SMEP to 0 to bypass SMEP
 - Can not use in the scene of mountmgr: It triggers stack overflow in DriverEntry which in the context of SYSTEM process. Even if we turned off SMEP, it also can not access user mode memory
- My method: Setting ROP chain with ExAllocatePool + MmCopyVirtualMemory combo to bypass SMEP

- Setting ROP chain under x64: most of API use registers to pass parameters. So we need to control registers.
- The compact design of ROP chain:
 - HvlEndSystemInterrupt : manipulate rdx / rax / rcx
 - KeInitializeEnumerationContext: save rcx to address
 - IoCallDriver: manipulate jump
 - RtlIsServicePackVersionInstalled : manipulate r8
 - EtwEventEnabled : manipulate r9
 - KeSetHardwareCounterConfiguration : manipulate r8
 - PsGetThreadTeb : load rax from address
 - MmMapIoSpace : manipulate rsp
- All of above are exported functions, without brute force code search.

- The first problem in setting ROP chain:
 - The MountmgrReadNoAutoMount will overwrite stack of outside function, and it will overwrite an important variable DeviceObject.
 - Before DriverEntry returning, it would call IoRegisterShutdownNotification to register shutdown callback for this DeviceObject. A wrong variable will cause BSOD.
 - Solution: Use NtQuerySystemInformation to get current thread object address, and use it to overwrite DeviceObject variable.
 - IoRegisterShutdownNotification will do the bit-or operation with DeviceObject->Flags(offset 0x30) and DO_SHUTDOWN_REGISTERED. There corresponds StackLimit field in thread object, and do not affect of thread execution.
 - After bypassing SMEP and taking control, we need unregister the shutdown callback and fix thread object.

- The second problem in setting ROP chain:
 - Under x64, ExAllocatePool will destroy $rsp+8 \sim rsp+0x20$ and use this area as temporary variables.
 - That will destroy the ROP chain.
 - Solution: put a 'add rsp , 0x28' invoke next to the ExAllocatePool 's invoke, and bypass the temporary variables area.
 - We can continue to run ROP chain after bypassed temporary variables area.

- The complete process of bypass DSE on Windows8 x64:
 - Prepare information of ROP chain, and write stack data into NoAutoMount key value.
 - Call NtUnloadDriver to unload loaded mountmgr driver , and modify service registry key value, then call NtLoadDriver to load old mountmgr driver.
 - Start to trigger the calling of ROP chain, control register to call ExAllocatePool to allocate kernel executable memory.
 - Control register to call MmCopyVirtualMemory and copy user mode shell code memory from attack process into kernel space
 - Load the address from KiBugcheckData and jump to shell code
 - Fix up stack environment and shutdown callback, modify ci!g_CiOptions to disable DSE on Windows8
- Demo source code: <http://code.google.com/p/bypass-x64-dse>

-
- J00ru . [Exploiting the otherwise non-exploitable:Windows Kernel-mode GS Cookies subverted](#)
 - Artem Shishkin . [Intel SMEP overview and partial bypass on Windows 8](#)
 - Artem Shishkin & Ilya Smit. [Bypassing Intel SMEP on Windows 8 x64 Using Return-oriented Programming](#)
- 

- Q&A