Windows XP/Vista Kernel Registry Handling Denial of Service

by Matthew "j00ru" Jurczyk and Gynvael Coldwind
Hispasec

1. Basic Information

Name	Windows XP/Vista Kernel Registry Handling Denial of Service
Class	Implementation error
Impact	Medium
Credits	Matthew "j00ru" Jurczyk, Gynvael Coldwind
Discovered	2008-12-10
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 (other variations of this executable, e.g. *ntkrnlpa.exe* are also affected). 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 Blue Screen of Death followed by a hard reboot.

The vulnerability covered in this paper relies on the fact that some very internal kernel code makes unsafe assumptions about the registry layout and dependences. The situation causing a system crash would actually never occur in real life, though it is still possible to manually trigger the necessary conditions and take advantage of the fact that the high-level code doesn't fully validate the registry structures being operated on.

Every single Microsoft Windows (NT-family) version prior to Windows7 beta is affected, including Windows Vista SP1 with latest patches.

3. Vulnerability details

The vulnerability is strongly related to the registry "symbolic link" mechanism provided by Microsoft. It is commonly used to create invisible transitions between separate keys, thought the vendor itself does not provide any technical document about neither how could a normal programmer use it in his software nor how does it work internally. This is probably because Microsoft wants to keep this feature for his own purposes and not let people mess too much with the Windows Registry.

Despite presenting such approach, nothing has been done to prevent a restricted user's applications access these extra features and take advantage of unknown possibilities that haven't yet been publicly explained. The "symbolic link" functionality creates really enormous possibilities when it comes to vulnerability research. In this paper, it will be shown how to crash modern Windows versions using the mechanism in a way that the developers probably haven't ever thought of.

By describing symbolic links as "invisible forwarders", it is meant every single setting related to the destination key is inherited by the link, including security descriptors. If one wants to get or set the security for given key using the link's name, all the changes will be applied to the final (after all the translations) key. A Denial of Service condition occurs while changing the security descriptor of a link that itself points to another link.

To be exact, the BugCheck is raised when trying to free a NULL Pointer:

```
FreeBuffer(ObjectName->Buffer);
```
(*vulnerable function pseudocode*)

Typically after such call, a Blue Screen of Death appears informing the users of a wrong deallocation address being passed to the Free() function.

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 change their access rights. The impact of a single attack, considering the above conditions, is rated as medium.

5. Disclaimer