

# Dyre Banking Trojan Exploits CVE-2015-0057

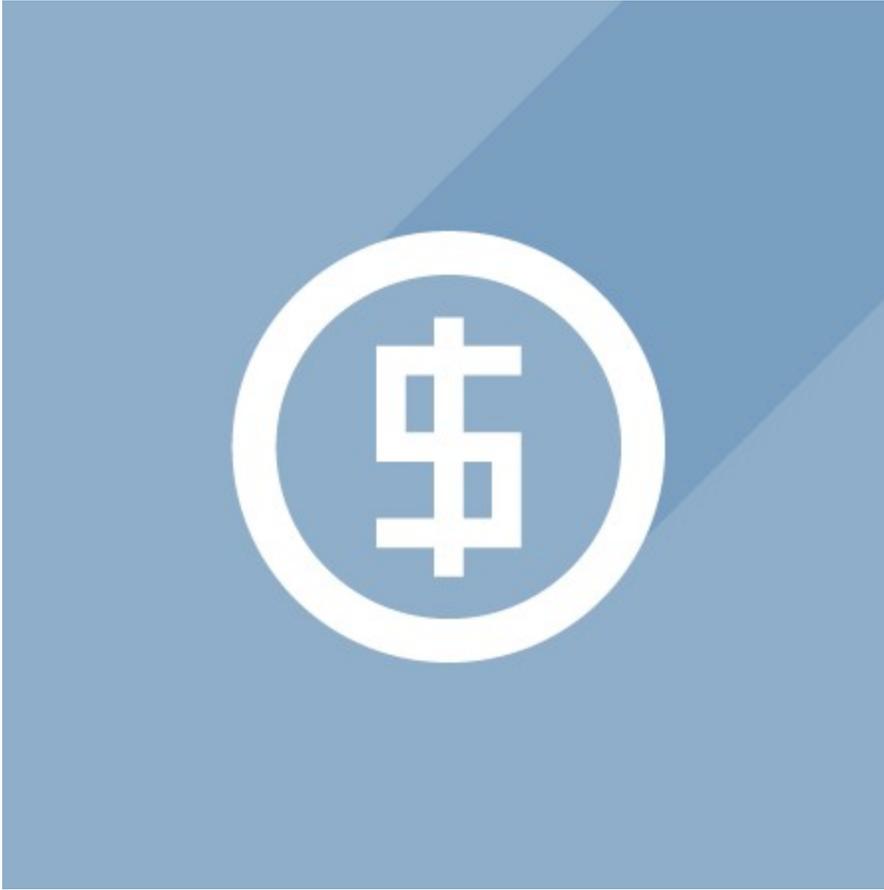
---

[fireeye.com/blog/threat-research/2015/07/dyre\\_banking\\_trojan.html](http://fireeye.com/blog/threat-research/2015/07/dyre_banking_trojan.html)



New variants of the Dyre Banking Trojan are exploiting a patched vulnerability in Microsoft Windows (**CVE-2015-0057**). We first observed these new variants on June 17<sup>th</sup> -- more than 4 months after Microsoft released the patch ([MS15-010](#)).

CVE-2015-0057 is a Use-After-Free vulnerability that exists in the win32k.sys component of the Windows Kernel which can be exploited to perform local privilege escalation. The vulnerability was reported to Microsoft by Udi Yavo, and, after the patch was released, Udi authored [a short blog](#) describing a few details of the vulnerability. However, it is important to note that the exploit code for **CVE-2015-0057** has not yet been made public.



In addition to this, these new variants of Dyre also include an exploit for **CVE-2013-3660** as a fallback in case the system is patched for CVE-2015-0057.

### **Initial Validation**

Initially, the malware first checks its privilege level. If it already has administrator rights, it decrypts the embedded resource **FILE1** using a single byte XOR decryption. The decrypted binary is then written to the file system and executed.

If the malware does not have administrator rights, it proceeds to perform the following sequence:

1. If the NT version is 5.x (Windows XP/Windows 2000/Windows 2003), it scans the following registry paths for the pattern "3036220":

`SOFTWARE\Microsoft\Updates\Windows XP\SP0`

`SOFTWARE\Microsoft\Updates\Windows XP\SP10`

`SOFTWARE\Microsoft\Updates\Windows XP\SP3`

`SOFTWARE\Microsoft\Updates\Windows XP\SP4`

This pattern is used to detect if the Microsoft Update (KB3036220 or MS-15-010) is installed and thus CVE-2015-0057 is patched. In this case, Dyre checks if CVE-2013-3660 is patched as well by scanning the registry for the pattern: "KB2850851" which corresponds to the security update for CVE-2013-3660. It looks for this pattern in the same registry paths as mentioned above. If it is not patched, then the exploit code corresponding to CVE-2013-3660 is used for escalation of privileges.

2. If the NT version is 6.0 or 6.1 (Windows Vista/Windows 7/Windows Server 2008), then it proceeds to scan the following path in the registry for the pattern "3036220":

SOFTWARE\Microsoft\Windows\CurrentVersion\Component Based Servicing\Packages.

Now, let us discuss in depth how this binary exploits the vulnerability from CVE-2015-0057.

### **Exploitation of CVE-2015-0057**

The main vulnerability is in the function: win32k!xxxEnableWndSBArrows which is used to manage the Window scroll bars. The vulnerable code in xxxEnableWndSBArrows modifies the cEntries field of a previously allocated tagPROPLIST structure which is used by the exploit to modify other critical data structures.

Below are the steps used by Dyre to exploit the use-after-free vulnerability present in win32k!xxxEnableWndSBArrows:

1. Allocate large numbers of contiguous kernel memory by calling the routine win32k!CreateProp (to create tagPROPLIST structure of a particular size).
2. Free some tagPROPLIST structures selectively, to create memory holes for the vulnerable structure to fall into.
3. Allocate SBINFO structure by calling the routine win32k!\_InitPwSB. tagPROPLIST size can be controlled from user mode.

An attacker would set the size of allocated tagPROPLIST to the size of SBINFO. After calling \_InitPwSB, an SBINFO structure will fall into one of the holes in memory created in step 2.

4. Within the function, win32k!xxxEnableWndSBArrows, the routine xxxDrawScrollBar initiates a user mode callback.

a) In the process of the user mode callback (USER32!\_\_ClientLoadLibrary), the attacker destroys the window, which frees a block of memory.

b) Before returning to xxxEnableWndSBArrows (in kernel mode), the attacker will invoke win32k!NtUserSetProp (in user mode). This action will occupy the kernel memory previously freed with a new tagPROPLIST structure, replacing the previously allocated SBINFO structure.

5. When the user mode callback returns to the kernel, the routine win32k!xxxEnableWndSBArrows will cause a modification (OR Instruction) of the tagPROPLIST.cEntries field. This will increase the size of the tagPROPLIST.aprop array. The main vulnerability is present here which is used by the exploit in the next steps.

Figure 1 shows the modification of cEntries field of tagPROPLIST structure in win32k!xxxEnableWndSBArrows:

```

Command - Kernel 'com:pipe,port=\\.\pipe\com_1,baud=115200,reconnect' - WinDbg:6.3.9600.16384 AMD64
1: kd> r
eax=0000000c ebx=0000000a ecx=00000001 edx=00000001 esi=bc65ae98 edi=bc6665e0
eip=bf9d12b0 esp=f5968d18 ebp=f5968d2c iopl=0         nv up ei pl zr na pe nc
cs=0000  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000206
win32k!xxxEnableWndSBArrows+0xe2:
bf9d12b0 0906             or             dword ptr [esi],eax  ds:0023:bc65ae98=00000004
1: kd> dd bc65ae90
tagPROPLIST object
bc65ae90 00060006 00080100 00000004 00000004  Use-After-Free
bc65aea0 00376e38 0000a918 00000000 00002f00  tagPROPLIST.cEntries
bc65aeb0 00000000 00003f00 44444444 00004f00  OR instruction overwrite
bc65aec0 00060006 00080100 00000004 00000004  tagPROPLIST object
bc65aed0 00babb58 0000a918 00000000 0000262d
bc65aee0 00000000 0000362d 00000000 0000462d
bc65aef0 00060006 00080100 00000004 00000004
bc65af00 00babfd0 0000a918 00000000 0000262e
1: kd> kb
ChildEBP RetAddr  Args to Child                               Call Stack
f5968d2c bf91140e 00000003 00000003 00000003 win32k!xxxEnableWndSBArrows+0xe2
f5968d50 8054160c 00030284 00000003 00000003 win32k!NtUserEnableScrollBar+0x69
f5968d50 7c92eb94 00030284 00000003 00000003 nt!KiFastCallEntry+0xfc
0012fe5c 7746e6ee 5adeb71f 00030284 00000003 ntdll!KiFastSystemCallRet
0012fe7c 77467e01 00030284 00000003 00000003 USER32!NtUserEnableScrollBar+0xc
0012feb0 00406ef1 00030284 00000003 00000003 USER32!EnableScrollBar+0x54
0012ff4 00407de1 00418128 0012ffff 00405b26 0
0012ffb0 00405b26 00418128 0007da40 7c92e1fe 0
0012ffff 00000000 00405ae1 00000000 78746341 0

```

Figure 1: Modification of tagPROPLIST.cEntries field

6. Using this corrupted tagPROPLIST object, an attacker is able to achieve arbitrary relative memory writes. This primitive enables the exploit to overwrite a value within a critical GDI object.

7. Lastly, the attacker calls a function in win32k.sys which uses the above corrupted GDI object and this results in arbitrary memory write of absolute addresses.

8. The attacker is then able to modify a kernel function pointer to point to his shellcode buffer in user mode, leading to escalation of privileges.

Figure 2 summarizes the use-after-free memory overwrite and the sequence of arbitrary memory writes:

```

Command - Kernel 'com:pipe,port=\\.\pipe\com_1,baud=115200,reconnect' - WinDbg:6.3.9600.16384 AMD64
1: kd> dd bc65ae90 Pool Header
bc65ae90 00060006 00080100 0000000c 00000006  tagPROPLIST object
bc65aea0 00376e38 0000a918 00000000 00002f00  tagPROPLIST.cEntries
bc65aeb0 00000000 00003f00 44444444 00004f00  U-A-F overwrite
bc65aec0 00060006 00080100 0000000c 0000000b  tagPROPLIST object
bc65aed0 00000000 00000000 00000000 0000262d  tagPROPLIST.cEntries
bc65aee0 00000000 0000362d 00000000 0000462d  2nd overwrite
bc65aef0 0006000c 00080100 006a00c5 00000000
bc65af00 00000000 81815a38 bc65aef8 00000001
bc65af10 00000000 00000000 ffffffff 0002c021
bc65af20 00000000 00000000 00000000 00000038
bc65af30 00000000 00000000 00000000 00000000
bc65af40 00000000 00000000 00000000 00000000  tagMENU.rgItems
bc65af50 000c0006 00080000 bc6301a8 bc659d28  overwrite
bc65af60 00000000 00000000 00000000 00002630
bc65af70 00000000 00003630 00000000 00004630
bc65af80 00060006 00080100 00000004 00000004

```

Figure 2: Sequence of Arbitrary Memory Writes

The technique used to redirect control flow to user mode shellcode and escalate the privileges is as follows:

1. HalDispatchTable+0x4 will be overwritten by the exploit as mentioned in step 7 above.

Figure 3 shows the exact point when HalDispatchTable+0x4 is overwritten with the address of user mode shellcode.

```
Disassembly - Kernel 'compipe,port=\\.\pipe\com_1,baud=115200,reconnect' - WinDbg.6.3.9600.16384 AMD64
Offset: @$scope!p
bf8374e6 0185a8000000 jne win32k!xxxSetLPIITEMInfo+0x0ee (bf837594)
bf8374ec f6460402 test byte ptr [esi+4], 2
bf8374f0 7406 je win32k!xxxSetLPIITEMInfo+0x129 (bf8374f8)
bf8374f2 8b4610 mov eax, dword ptr [esi+10h]
bf8374f5 8d4309 mov dword ptr [ebx+1], eax
bf8374f8 f6460402 test byte ptr [esi+4], 2
bf8374fc 018521e1ffff jne win32k!xxxSetLPIITEMInfo+0x12f (bf837523)
bf837502 f6460401 test byte ptr [esi+4], 1
bf837506 7420 je win32k!xxxSetLPIITEMInfo+0x15c (bf837528)
bf837508 8d4304 lea eax, [ebx+4]
bf83750b 8320f4 and dword ptr [eax], 0FFFFFF4h

Command - Kernel 'compipe,port=\\.\pipe\com_1,baud=115200,reconnect' - WinDbg.6.3.9600.16384 AMD64
0: kd> dd nt!HalDispatchTable+4
00405e46 8061c946 80571dd9 00000000
00405e4c 804ee554 80571410 80570bbc 0057024c 0x00405e48 points to user mode shellcode
00405e50 805704aa 804ee54e 804ee5d0 804ee5d0
00405e56 8061bd46 8061c7b4 806e7944 8061c338
00405e5c 80571df4 804ee570 804ee57e 8061c934
00405e62 804ee57e 00000002 804ee51e 804ee51e
00405e68 8061bd3e 80571de6 8061836e 80618320
00405e6e 1751a12e 17519f82 806e7108 806eb5ae
0: kd> u 00405e46
8c70899bdd7af6730730b167c59ca96a+0x5e46:
00405e46 55 push ebp
00405e47 8bec mov ebp, esp
00405e49 51 push ecx
00405e4a 51 push ecx
00405e4b 8365fc00 and dword ptr [ebp-4], 0
00405e4d 8365f800 and dword ptr [ebp-8], 0
00405e53 56 push esi
00405e54 57 push edi
```

Figure 3: HalDispatchTable+0x4 Overwrite and User Mode Shellcode

2. From user mode, it calls the API, ZwQueryIntervalProfile. In kernel mode, this transfers control to HalDispatchTable+x04. Since the value at this location is overwritten with user mode shellcode, the control is transferred to privilege escalation shellcode.

This shellcode uses the standard method of copying the token field of the EPROCESS structure of the SYSTEM process and storing it in the EPROCESS structure of the target process. The following steps describe this technique:

1. Call PsLookupProcessByProcessId to get the pointer to EPROCESS structures of both victim process and SYSTEM process.
2. Call PsReferencePrimaryToken to get the token of both the processes.
3. Iterate over the EPROCESS structure of victim process to find the offset where token is stored.
4. Store the SYSTEM process token at that offset to escalate its privileges.

Similar shellcode was used in this binary for exploiting CVE-2013-3660 (used only if security update for CVE-2015-0057 is installed) however, we observed the following differences which indicate that these exploits were most likely written by 2 different authors or re-used from elsewhere.

**CVE-2015-0057:**

It fetches the EPROCESS structure of SYSTEM process using PsLookupProcessByProcessId.

It uses an iteration counter of 0x200 to scan the EPROCESS structure for offset of token.

**CVE-2013-3660:**

It fetches the EPROCESS structure of SYSTEM process using PsInitialSystemProcess.

It uses an iteration counter of 0x300 to scan the EPROCESS structure for offset of token.

**Kernel Mode Protections**

It is important to note that this exploit does not bypass SMEP since it performs a user mode callback from kernel mode as described above. So, Windows 8.1 is not impacted by it.

For users of Win NT 5.x, Win NT 6.0 and Win NT 6.1, they are advised to install the update KB3036220 since this exploit is in the wild now.

**Indicators of this Exploit Technique in User Mode**

Below are some of the indicators of this exploit in user mode:

1. A Long sequence of calls to SetPropA/RemovePropA APIs.
2. Creation of an array of Windows in a loop and setting the Property list of each Window using SetPropA.
3. Destroying few windows from the above created array to create holes.
4. Calling EnableScrollBar with the parameters as shown below:

```
EnableScrollBar(hWnd, SB_BOTH, ESB_DISABLE_BOTH);
```

5. Calling ZwQueryIntervalProfile to perform the actual privilege escalation.

**Conclusion**

The inclusion of this most recent privilege escalation exploit in the code of Dyre indicates the sophistication of the malware authors of banking trojans.