

ENIGMA0X3

<< BYPASSING APPLICATION WHITELISTING BY
USING DNX.EXE

LATERAL MOVEMENT USING THE
MMC20.APPLICATION COM OBJECT >>

BYPASSING APPLICATION WHITELISTING BY USING RCSI.EXE

November 21, 2016 by [enigma0x3](#)

Over the past few weeks, I have had the pleasure to work side-by-side with Matt Graeber ([@mattifestation](#)) and Casey Smith ([@subtee](#)) researching Device Guard user mode code integrity (UMCI) bypasses. If you aren't familiar with Device Guard, you can read more about it here: <https://technet.microsoft.com/en-us/itpro/windows/keep-secure/device-guard-deployment-guide>. In short, Device Guard UMCI prevents unsigned binaries from executing, restricts the Windows Scripting Host, and it places PowerShell in Constrained Language mode.

After discovering an Application Whitelist bypass using dnx.exe, I decided to take a look at what other potential tools existed that allowed for execution of arbitrary C#, which led me to this blog post by Microsoft: <https://blogs.msdn.microsoft.com/visualstudio/2011/10/19/introducing-the-microsoft-roslyn-ctp/>

I then started looking for a Microsoft compiled version, which led me to the "[Microsoft Roslyn CTP](#)" download page. Unfortunately, this requires Visual Studio 2012 and the VS2012 SDK

installed, but those are freely available.

It is stated in that blog that the Microsoft Roslyn CTP contains a new binary called “rcsi.exe”. This particular binary is one of the first attempts at integrating Roslyn on the backend (it is now nicely integrated into Visual Studio 15 Update 1, which interfaces with csi.exe). More information on Roslyn can be found here: <https://github.com/dotnet/roslyn>

Recently, my co-worker Casey Smith (@subtee) blogged about [bypassing application whitelisting using csi.exe](#). The difference between these bypasses is csi.exe is interactive and rcsi.exe is not. Luckily for us, you can utilize the introduction of [C# Scripting](#) with rcsi.exe to execute unsigned code.

In a Device Guard scenario, rcsi.exe is allowed to execute as it is a Microsoft signed binary that can be installed along side of Visual Studio 2012. In order to execute rcsi.exe on a Device Guard system (assuming it isn't already installed), you will need to gather rcsi.exe and its required dependencies (there are only 2 of them), and transport everything to your target (this is an exercise left up to the reader).

With everything required now on our target host, we can now start down the path of bypassing Device Guard's UMCI. Since rcsi.exe allows for executing C# scripts, we can use it to execute arbitrary, unsigned C# code by passing the binary our own csx script.

For example, we can create a csx file and add whatever C# code we want. Keep in mind that it uses Roslyn, so you don't need to add classes. To demonstrate the execution of unsigned code, we can keep things simple:

```
bypass.csx
1 using System;
2 Console.WriteLine("Hello, I am unsigned C# code running inside rcsi.exe!");
3 Console.ReadLine();
```

Now that we have our C# script created, we can execute our C# using rcsi.exe by simply passing our csx to it. This is done on a PC running Device Guard:

```
Command Prompt - rcsi.exe bypass.csx

C:\Users\Matt\Desktop\Roslyn Bypass>rcsi.exe bypass.csx
Hello, I am unsigned C# code running inside rcsi.exe!
```

As you can see above, our unsigned C# successfully executed and is running inside of rcsi.exe.

Fortunately, these “misplaced trust” bypasses can be mitigated via code integrity policy FilePublisher file rules. You can read up on creating these mitigation rules here:

<http://www.exploit-monday.com/2016/09/using-device-guard-to-mitigate-against.html>

You can find a comprehensive bypass mitigation policy here:

<https://github.com/mattifestation/DeviceGuardBypassMitigationRules>

If you want to know more about “Misplaced Trust” bypasses, you can find [@subtee](#)’s awesome Bluehat presentation slides here: <https://github.com/subTee/BlueHat2016>

Cheers!
Matt Nelson

SHARE THIS:



Be the first to like this.

RELATED

Bypassing Application
Whitelisting By Using dnx.exe

UMCI Bypass Using
PSWorkflowUtility: CVE-2017-
0215

Defeating Device Guard: A look
into CVE-2017-0007
With 1 comment

Bookmark the [permalink](#).

LEAVE A REPLY

Enter your comment here...

ARCHIVES

- [January 2018](#)
- [November 2017](#)
- [October 2017](#)
- [September 2017](#)
- [August 2017](#)
- [July 2017](#)
- [April 2017](#)
- [March 2017](#)
- [January 2017](#)
- [November 2016](#)
- [August 2016](#)
- [July 2016](#)
- [May 2016](#)
- [March 2016](#)
- [February 2016](#)
- [January 2016](#)
- [October 2015](#)
- [August 2015](#)
- [April 2015](#)
- [March 2015](#)
- [January 2015](#)
- [October 2014](#)
- [July 2014](#)
- [June 2014](#)

RECENT POSTS

- [Reviving DDE: Using OneNote and Excel for Code Execution](#)
- [Lateral Movement Using Outlook's CreateObject Method and DotNetToJScript](#)
- [A Look at CVE-2017-8715: Bypassing CVE-2017-0218 using PowerShell Module Manifests](#)
- [UMCI Bypass Using PSWorkflowUtility: CVE-2017-0215](#)
- [Lateral Movement using Excel.Application and DCOM](#)

CATEGORIES

- [Uncategorized](#)

RECENT COMMENTS



Soc on [Defeating Device Guard: A](#)

"Fileless..." on ["Fileless" UAC Byp...](#)

"Fileless..." on [Bypassing UAC using App P...](#)



Windows 10 UAC Looph... on [Bypa...](#)
UAC using App P...

NexusLogger: A New C... on ["Filele...](#)
UAC Byp...

META

- [Register](#)
- [Log in](#)
- [Entries RSS](#)
- [Comments RSS](#)
- [WordPress.com](#)

- [March 2014](#)
- [January 2014](#)

[Blog at WordPress.com.](#)