

PSArmoury 1.4 - now with even more armour

cyberstoph.org/posts/2020/02/psarmoury-1.4-now-with-even-more-armour

February 28, 2020

TL;DR;

I recently added some improvements to PSArmoury, which I like to share with you in this post

- Support for BlockDLL process mitigation to protect your armoury
- New config parameter that lets you choose the branch in github
- Simple way to create an armoury from a local file
- Automatic inventory function

Still reading? Great, let's go.

Introducing BlockDLL process mitigation

Thanks to the great C# port of @_RastaMouse it was very easy to implement a process mitigation, that was actually meant to protect processes from malicious injections. By setting a special flag in the extended startupinfo of a new process, we can prevent any non-microsoft DLL from loading into that process. This can prevent certain EDR solutions from detecting well-known tools inside powershell.

To use this with PSArmoury, just create your armoury with the **-EnhancedArmour** switch like shown below.

```
PS C:\Users\cfalta\bin> New-PSArmoury -FromFile .\ADCS.ps1 -Path .\MyArmoury.ps1 -EnhancedArmour
PSArmoury: your armoury contains 1 repositories. Starting to process.
PSArmoury: processing repository LocalRepo
PSArmoury: download complete, starting encryption
PSArmoury: you did not supply a password, so we will generate a random one. You might want to write that down.
PSArmoury: your password is pw_0XK:WUL
PSArmoury: script processing complete, creating armoury. Happy hacking :)
PS C:\Users\cfalta\bin>
```

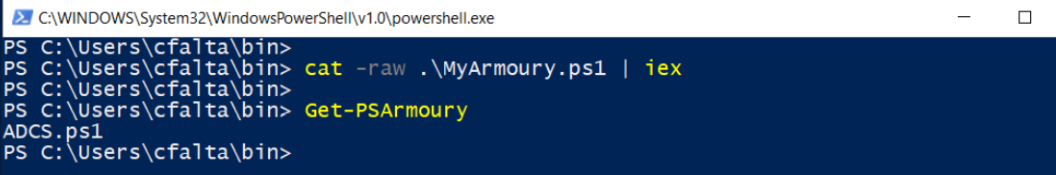
Note that this will change your user experience a bit. If you try to load an enhanced Armoury it will not decrypt it's content but start a new, protected powershell process and ask you to run it in there manually again. This might seem a bit cumbersome at first but the reasons are simple:

- We could also apply the process mitigation policy to the first powershell process we run in (just use Set-ProcessMitigation) but at that point, the process already started and any EDR/AV DLL already runs in that process. Therefore, we need to create a new, clean process.
- There are various ways to automatically run your armoury in the second powershell process, but then again this might raise a red flag itself so I went for the manual approach.

So to use it, just run your armoury as usual and it will guide you through the rest of the process as shown below.

```
PS C:\Users\cfalta\bin> cat -raw .\MyArmoury.ps1 | iex
PSArmoury: Your armoury is set to run with enhanced process mitigation policy. This will block any Non-Microsoft DLLs (e.g. AV) from running inside PowerShell.
PSArmoury: We will now spawn a new, protected PowerShell process. You have to load your armoury manually in there again to continue.
PSArmoury: Press any key to continue...

New PowerShell with PID 5784 started.
PS C:\Users\cfalta\bin>
```



Choose github branch

Every item of type “GitHubRepo” in your configuration file can now contain an attribute called “Branch”. If it exists, PSArmoury will try to use the branch name supplied in this attribute. If it does not exist, it will use the default branch as before. In the sample config you find on github, we use this attribute to download the dev branch of PowerSploit instead of the master.

```
{
  "Name": "PowerSploit",
  "Type": "GitHubRepo",
  "URL": "https://api.github.com/repos/PowerShellMafia/PowerSploit",
  "Branch": "dev",
  "FileInclusionFilter": "*.ps1",
  "FileExclusionFilter": ["*.tests.ps1"]
}
```

Create armoury from file

PSArmoury now offers a simple way to create an armoury from a local file. This comes in handy if you just want to quickly protect one (or many) powershell scripts you have on your disk without the need to create a config file. Just run PSArmoury with the **-FromFile** switch and pass it the path to a file or folder containing powershell scripts. You can omit the **-Config**-switch but note that only files with an extension of ***.ps1** will be included.

```
Windows PowerShell
PS C:\Temp> ls

Directory: C:\Temp

Mode                LastWriteTime         Length Name
----                -
-a-----         2/28/2020   5:06 PM             72 testscript.ps1

PS C:\Temp> New-PSArmoury -FromFile .
PSArmoury: your armoury contains 1 repositories. Starting to process.
PSArmoury: processing repository LocalRepo
PSArmoury: download complete, starting encryption
PSArmoury: you did not supply a password, so we will generate a random one. You might want to write that down.
PSArmoury: your password is Nt4Xsw79w:
PSArmoury: script processing complete, creating armoury. Happy hacking :-)
PS C:\Temp> |
```

Inventory

I occassionally find myself in the situation where I don't remember what scripts I put inside an armoury, especially when it's been some time since I created it. Since everything of relevance is encrypted in the file on disk, it is not so easy to get that information. Therefore every armoury now contains an inventory function called **Get-PSArmoury**, which will print the names of all files included to stdout. This function is built dynamically during creation of the armoury.

```
PS C:\Temp> New-PSArmoury -FromFile .
PSArmoury: your armoury contains 1 repositories. Starting to process.
PSArmoury: processing repository LocalRepo
PSArmoury: download complete, starting encryption
PSArmoury: you did not supply a password, so we will generate a random one. You might want to write that down.
PSArmoury: your password is Nt4Xsw79w:
PSArmoury: script processing complete, creating armoury. Happy hacking :-)
PS C:\Temp>
PS C:\Temp> . .\MyArmoury.ps1
PS C:\Temp> Get-PSArmoury
testscript.ps1
PS C:\Temp> |
```

That's it for now. If you are using PSArmoury and experience any issues or have ideas for new features, I would be happy to hear from you. Just send me a mail or open an issue on github.

Have a nice weekend!
