

# the Generic bypasses and the famous ones

## Placing files in writable directories

---

### Bypass :

Put the files in folders where you have write access, and change the permissions to allow execution. If you can't run the file because execution is blocked, you can either break inheritance or create a hard link.

#### #command

Montre tous les dossiers/fichiers sur `C:\` (et sous-dossiers) où le groupe/utilisateur "hajar" a des droits d'écriture, en listant uniquement ceux où l'accès est accordé.

```
C:\Users\hajar\Downloads\AccessChk>.\accesschk.exe "hajar" C:\ -wsu
```

```
C:\Windows\Tasks\*  
C:\Windows\Tracing\*  
C:\Windows\servicing\Sessions\*  
C:\Windows\Registration\CRMLog\*  
C:\Windows\ServiceProfiles\LocalService\AppData\
```






```
Local\Microsoft\Dlna\DeviceIcon\*
C:\Windows\System32\Microsoft\Crypto\RSA\Machine
Keys
C:\Windows\SysWOW64\FxsTmp\*
C:\Windows\SysWOW64\Tasks\*
C:\Windows\SysWOW64\com\dmp\*
C:\Windows\Logs\OpsMgrTrace\*
C:\Windows\servicing\Packages\*
C:\Windows\Temp\*
C:\Windows\System32\Tasks_Migrated\*
C:\Windows\System32\FxsTmp\*
C:\Windows\System32\Tasks\*
C:\Windows\System32\com\dmp\*
C:\Windows\System32\Drivers\DriverData\*
C:\Windows\System32\spool\PRINTERS\*
C:\Windows\System32\spool\SERVERS\*
C:\Windows\System32\spool\drivers\color\*
C:\Windows\System32\Tasks\Microsoft\Windows\Sync
Center
C:\Windows\SysWOW64\Tasks\Microsoft\Windows\Sync
Center\*
C:\Windows\SysWOW64\Tasks\Microsoft\Windows\PLA\
System\*
```

## Mitigation:

### AppLocker Rules:

Block execution from these directories using path-based Deny rules for Executable, Script, Windows Installer ,Packaged App ,and DLL types

Applies to:

-  Executable Files
-  Scripts
-  Windows Installer Files
-  DLL Files
-  Packaged apps (if installed there)

## User writeable files

---

### Bypass :

These are user-writable files that can be modified by the first user who logs in.

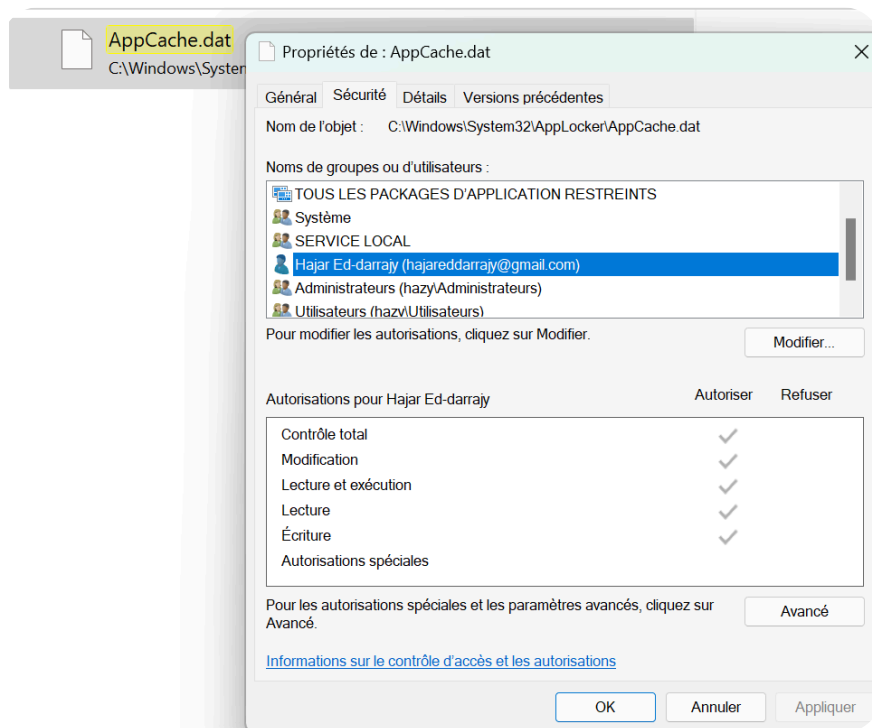
That user will have full access to them, and they can potentially be used to bypass AppLocker protections.

For example, in the folder:

`C:\Windows\System32\AppLocker`

The following files are writable:

- `AppCache.dat`
- `AppCache.dat.LOG1`
- `AppCache.dat.LOG2`



## Mitigation:

### AppLocker Rules:

the rule that should be set to fix this issue would be :  
deny rules blocking execution from the  
C:\Windows\System32\AppLocker path, and also  
remember to block alternate data stream from the  
same folder.

C:\logs\\*:\*

%USERPROFILE%\Downloads\\*:\*

wildcard :

C:\Windows\System32\AppLocker\\*:\*

Applies to:

- ☒ Executable Files
- ☒ DLL Files

-  Scripts

# PowerShell Version 2

---

## Bypass :

If PowerShell v2 is enabled, attackers bypass AppLocker since v2 does not respect security restrictions like constrained language mode:

- Elle ne respecte pas les règles modernes de sécurité
- Elle n'active pas le mode restreint
- Elle désactive aussi la journalisation (pas de logs !)

## Mitigation :

- Disable PowerShell v2 via Windows Features

👉 how :

Aller dans Ajouter/Supprimer des fonctionnalités Windows > Décoche "Windows PowerShell 2.0"  
or via PowerShell Command (as Admin):

```
Disable-WindowsOptionalFeature -Online -  
FeatureName MicrosoftWindowsPowerShellV2 -  
norestart
```

- allow only signed PowerShell scripts using AppLocker Script Rules, while blocking all unsigned scripts

👉 how :

in the script rules under the publisher type(it's an allow rule) Browse to a signed script file (you can get a legit `.ps1` from Microsoft or sign one yourself)

Select:

- All scripts signed by this publisher
- Use current file version or any version depending on your needs

and in the same time block any unsigned script same process but this time with the path rules block this path

\*.\*

Applies to:

-  Scripts

## VSTO Add-ins (Office) – .vsto files

Bypass :

[#briefly](#)

Attackers use Visual Studio Tools for Office

(VSTO) to install malicious add-ins that execute automatically.

VSTO (Visual Studio Tools for Office) est un outil de Microsoft permet de créer des extensions pour pour les applications Office comme Word ou Excel. qui peuvent être installées sans droits administrateur et s'exécutent automatiquement à l'ouverture de l'application. Cette fonctionnalité peut être exploitée par un attaquant pour contourner AppLocker, car ces add-ins ne sont pas bloqués par défaut. Un attaquant peut ainsi créer un add-in malveillant, le faire installer par la victime, et exécuter du code à chaque lancement d'Office, offrant une persistance discrète. Pour se protéger, il faut restreindre les add-ins aux éditeurs approuvés, désactiver ceux non signés, renforcer les règles AppLocker, et sensibiliser les utilisateurs.

ref:<https://bohops.com/2018/01/31/vsto-the-payload-installer-that-probably-defeats-your-application-whitelisting-rules/>

## Mitigation

💡 rule : Consider adding VSTO files to Application Whitelisting policies.

Chemins typiques identifiés par Microsoft

Voici les chemins connus utilisés par ClickOnce (mécanisme d'installation des VSTO) :



```
%LOCALAPPDATA%\Apps\2.0\  
%USERPROFILE%\AppData\Local\assembly
```

```
%USERPROFILE%\AppData\Local\Apps\2.0\
```

Ce sont les répertoires par défaut où les add-ins VSTO sont installés lorsqu'un utilisateur clique sur un fichier

```
.vsto
```

Applies to:

-  Executable Files (the add-ins can launch code)
-  Packaged apps and installers

## NTFS Alternate Data Streams (ADS)

---

Bypass :

Attackers can bypass AppLocker using Alternate Data Streams (ADS), a feature in Windows that allows hiding malicious code inside files or folders without being noticed. Even if a folder is blocked by AppLocker, the hidden stream (like `folder:file.exe`) can still be executed. To prevent this, you need to add specific AppLocker rules that block not just the folder, but also any ADS inside it (e.g., using `logs:*`). This ensures AppLocker blocks all possible hidden executions in writable directories.

Mitigation:



<https://hitco.at/blog/howto-prevent-bypassing-applocker-using-alternate-data-streams/>

add to the path blocked :

```
*:*
```

Applies to:

- ☒ Executable Files
- ☒ Scripts
- ☒ DLL Files

## File Hash and DLL Hijacking

---

Bypass :

see this pdf : Thesis\_applocker\_bypas


Même si un exécutable est autorisé par hash, il peut être contourné si ce programme a une faille de DLL hijacking, car AppLocker ne bloque pas toujours les DLLs chargées. Il suffit d'injecter une DLL piégée pour exécuter du code malveillant.

## Mitigation

AppLocker Rules:

- Crée une règle DENY pour les DLLs chargées depuis des dossiers sensibles  
Exemples de chemins à bloquer :  
in DLL Rules

```
%TEMP%\*.dll  
%APPDATA%\*.dll  
C:\Users\*\AppData\Local\Temp\*.dll  
C:\ProgramData\*.dll(si modifiable)
```

- Block specific executables known to allow DLL injection:  
 Microsoft recommend blocking those binaries :  
ref : <https://learn.microsoft.com/en-us/windows/security/application-security/application-control-for-business/design/applications-that-can-bypass-appcontrol>

```
addinprocess.exe  
addinprocess32.exe  
addinutil.exe  
aspnet_compiler.exe  
atBroker.exe  
bash.exe  
bginfo.exe  
cdb.exe  
cmstp.exe  
control.exe  
cscript.exe  
csi.exe
```

dbghost.exe  
dbgsvc.exe  
dbgsvr.exe  
dfsvc.exe  
dnx.exe  
dotnet.exe  
forfiles.exe  
fsi.exe  
fsiAnyCpu.exe  
ie4unit.exe  
ieexec.exe  
installutil.exe  
infdefaultinstall.exe  
kd.exe  
kill.exe  
lxcrun.exe  
lxssmanager.dll  
mavinject.exe  
manage-bde.wsf  
Microsoft.Build.dll  
Microsoft.Workflow.Compiler.exe  
msbuild.exe  
msbuild.dll  
msdeploy.exe  
msdt.exe  
mshta.exe  
msiexec.exe  
msxsl.exe  
ntkd.exe  
ntsd.exe  
odbcconf.exe  
presentationhost.exe

powershellcustomhost.exe  
pubprn.vbs  
rcsi.exe  
regsvr32.exe  
regsvcs.exe  
regasm.exe  
rsi.exe  
rundll32.exe  
runscripthelper.exe  
slmgr.vbs  
syncappvpublishingserver.exe  
system.management.automation.dll  
te.exe  
texttransform.exe  
tracker.exe  
visualuiaverifynative.exe  
webclnt.dll  
wfc.exe  
windbg.exe  
winrm.vbs  
winword.exe  
wmic.exe  
wscript.exe  
wsl.exe  
wslconfig.exe  
wslhost.exe  
xwizard.exe

powershell.exe

Applies to:

-  DLL Files

# Reflective PE Injection

---

## Bypass :

ref:thesis applocker bypass pdf

ref:<https://www.hacking-tutorial.com/hacking-tutorial/how-to-bypass-windows-applocker/#sthash.iOhozZSl.dpbs>

<https://oddvar.moe/2019/05/29/a-small-discovery-about-applocker/>

<https://github.com/api0cradle/UltimateAppLockerBypassList/blob/master/Generic-AppLockerbypasses.md#placing-files-in-writeable-paths>

[#briefly](#)

Using PowerShell (e.g., PowerSploit), attackers load executables directly into memory, bypassing disk-based rules.

En Details:

Another method is by loading the executable in memory and launching it by jumping to its entry point. Since there is no execution path, the Applocker rule will not be triggered.

Given evil.exe as the malicious executable, it needs to be stored in a PowerShell variable:

```
$ByteArray =  
[System.IO.File]::ReadAllBytes("C:\PoC\evil.exe")  
;
```

Next, Invoke-ReflectivePEInjection function implemented in the PowerSploit framework

(["https://github.com/PowerShellMafia/PowerSploit/tree/master/CodeExecution"](https://github.com/PowerShellMafia/PowerSploit/tree/master/CodeExecution), Github) can be used to load it in memory and jump to that point.

```
Invoke-expression(Get-Content .\Invoke-ReflectivePEInjection.ps1 |out-string) Invoke-ReflectivePEInjection -PEBytes $ByteArray
```

Another way to achieve this is via CL\_LoadAssembly.ps1, which is a script that can be found in Windows systems that provide LoadAssemblyFromPath which can be used to load assemblies from .exe files.

## Mitigation :

- Interdire tous les scripts PowerShell sauf ceux signés par une autorité de confiance. en bloquant \*.ps1 , \*.psm1 , \*.psd1 et en autorisant juste les scripts signes

hadi drtha i idid allow only  
the ps1 scripts signed  
lakhriin psm1 et psd1 i can;t

- Forcer l'exécution de PowerShell en mode restreint

```
Set-Item -Path  
"HKLM:\SOFTWARE\Microsoft\PowerShell\1\ShellIds\  
Microsoft.PowerShell" -Name "LanguageMode" -  
Value "ConstrainedLanguage"
```

- Et bloquer : `powershell_ise.exe` `powershell.exe` avec des paramètres non sécurisés
- Bloquer ou surveiller PowerSploit on Surveille les appels à : `Invoke-ReflectivePEInjection` , `Invoke-Expression` et les Scripts stockés localement avec `Reflective` ou `PEInjection` dans le nom

hadi drtha

- Désactiver ou restreindre l'accès à .NET et aux scripts d'injection on essay de Restreindre les appels à `System.Reflection` et `System.IO` depuis PowerShell
- Règles "DLL Rules" \_(option avancée mais efficace) Bloquer le chargement de DLLs non signées ou non approuvées (utile contre DLL sideloading utilisé en complément)

AppLocker

Activer les règles pour `powershell.exe` , `powershell_ise.exe` ,  
`System32\WindowsPowerShell\v1.0\powershell.exe`



**Invoke-**

**`ReflectivePEInjection.ps1`**

— c'est quoi exactement ?

C'est un script PowerShell malveillant (ou post-exploitation) qui vient du framework PowerSploit, très utilisé par les attaquants ou les pentesters.

Il sert à faire une chose précise :  
Charger un fichier exécutable ( **.exe** )  
directement en mémoire, et l'exécuter sans laisser de trace sur le disque.

drt hada i blocked invoke-reflectivePEInjection.ps1  
with hash rule

had I script ma3rftoch

```
C:\Windows\diagnostics\system\AERO\CL_Invocation  
.ps1
```

Applies to:

-  Scripts

## LOLBAS Executables

---

Bypass :

Attackers abuse living-off-the-land binaries (LOLBAS) to execute code indirectly.

### Mitigation:

Create AppLocker rules to deny execution of known LOLBAS tools unless explicitly needed



- Applies to:
  -  Executable Files

## Recommendation

---

Utiliser des comptes utilisateurs non privilégiés

Les systèmes sur lesquels AppLocker est déployé doivent être utilisés avec un compte utilisateur standard. Les utilisateurs ne doivent en aucun cas disposer de privilèges d'administration (locaux ou distants).

Prêter une attention particulière aux dossiers accessibles en écriture

Lorsqu'une règle de type Chemin d'accès est utilisée, le dossier spécifié ainsi que ses sous-dossiers ne doivent être accessibles en écriture qu'aux administrateurs et aux entités SYSTEM

Compléter la règle par défaut créée pour C:\Windows

La règle par défaut créée pour C:\Windows doit être complétée par l'ajout d'exceptions pour les sous-répertoires accessibles en écriture par les utilisateurs.

Compléter la règle par défaut créée pour C:\Windows

La règle par défaut créée pour C:\Windows doit être complétée par l'ajout d'exceptions pour les sous-répertoires accessibles en écriture par les utilisateurs.

Une méthode alternative conseillée consiste à remplacer la règle par défaut par une règle de type Éditeur n'autorisant que les composants signés par Microsoft.

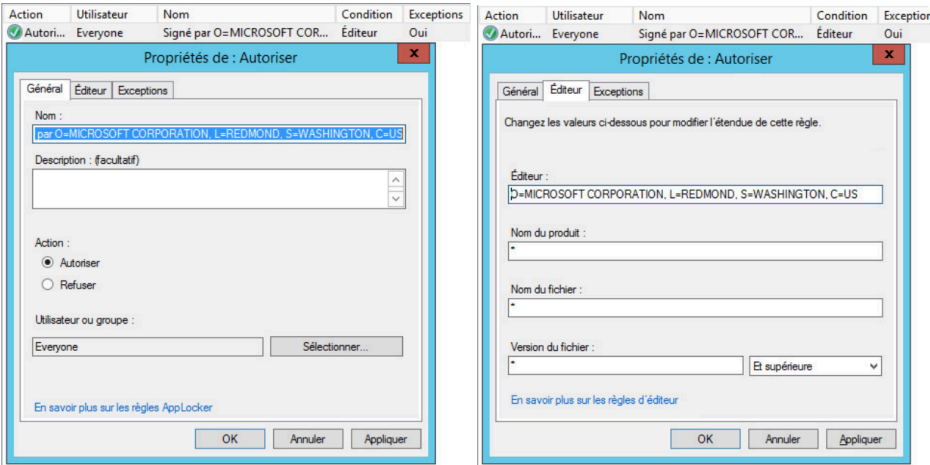


FIGURE 5 – Règle d'autorisation des exécutables de l'éditeur Microsoft

## Remplacer la règle créée par défaut pour C:\Program Files

Il est recommandé de supprimer la règle par défaut concernant C:\Program Files et de la remplacer par des règles autorisant explicitement chaque application de cet emplacement à s'exécuter.

## Créer les règles pour les scripts et les installeurs

### Refuser l'exécution de scripts depuis C:\Windows

Concernant les applications universelles de Microsoft, il est conseillé de les autoriser par deux règles de type Éditeur (une pour CN=Microsoft Corporation et une pour CN=Microsoft Operating System) puis d'ajouter des exclusions afin de bloquer spécifiquement les applications universelles de Microsoft qui ne sont pas souhaitées. De cette manière, si de nouvelles fonctionnalités du système sont fournies par des

applications universelles suite à des mises à jour de Windows, elles ne seront pas bloquées.

### Créer une exception pour l'exécutable InstallUtil.exe

Pour éviter le contournement trivial d'AppLocker, il est recommandé de créer une exception de type Éditeur pour interdire l'exécution du programme InstallUtil.exe du Microsoft Framework 4.0 par un utilisateur non privilégié.

Ces deux exceptions (regsvr32.exe et InstallUtil.exe) doivent être ajoutées à la règle qui autorise les exécutables de l'éditeur Microsoft , ou celle qui autorise l'emplacement %WINDIR%

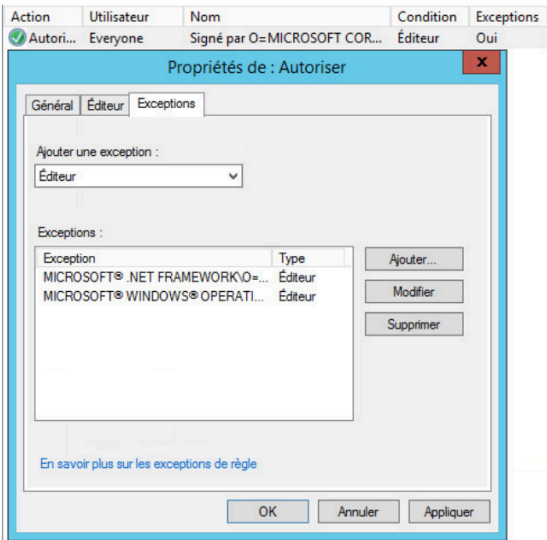


FIGURE 8 – Exceptions à la règle d'autorisation des exécutables Microsoft

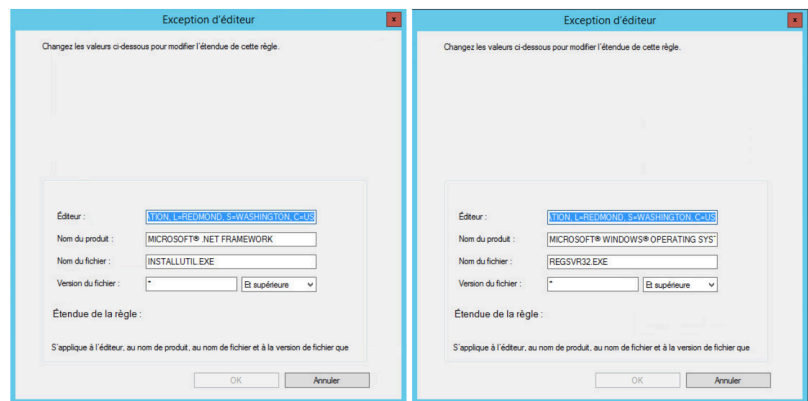


FIGURE 9 – Exceptions `InstallUtil.exe` et `regsvr32.exe` à la règle d'autorisation des exécutables Microsoft

Les composants ActiveX peuvent permettre d'instancier certains programmes dont l'exécution devrait être bloquée par AppLocker. Il est par exemple possible de lancer une session RDP en insérant le contrôle Microsoft RDP client dans un document Word. Seules des règles sur les bibliothèques permettent alors de bloquer ces moyens de contournement.

## Bloquer les bibliothèques qui permettraient d'instancier des programmes indésirables

Pour les programmes dont l'exécution est bloquée par AppLocker, si d'autres composants logiciels permettent de les instancier, ils doivent aussi être bloqués par

des règles sur les bibliothèques associées.

Les règles AppLocker autorisent ou empêchent le lancement des programmes, mais n'exercent aucun contrôle sur le comportement des programmes une fois lancés. En utilisant des fonctions avec des paramètres spéciaux <sup>3</sup>, un programme peut lancer des exécutables ou des bibliothèques sans que les règles ne s'appliquent. En particulier, les modules d'extension (greffons, plugins, etc.) de certains logiciels peuvent être exploités par un attaquant pour contourner les restrictions d'Applocker.

## Évaluer la sécurité des programmes autorisés

Les programmes autorisés à s'exécuter doivent avoir fait l'objet de vérifications pour s'assurer qu'ils ne comportent pas de fonctionnalités susceptibles de permettre un contournement d'Applocker.

Les utilisateurs ne doivent pas être autorisés à installer des modules d'extension.

Restreindre les possibilités d'exécution des codes interprétés par les programmes autorisés

Les logiciels offrant des possibilités de scripting doivent être paramétrés de façon à restreindre les possibilités d'exécution des codes interprétés. Dans le cas d'une suite bureautique, il est recommandé de restreindre l'exécution aux seules macros signées et de confiance, voire d'empêcher l'utilisation de l'éditeur de macros pour les utilisateurs qui n'en ont pas l'utilité.

Désactiver la fonction NTVDM

Si aucune application 16 bits n'est utilisée, désactiver la fonction NTVDM avec l'éditeur de stratégie de groupe .

*La fonction peut être désactivée en se rendant dans l'arborescence suivante : Configuration ordinateur → Modèles d'administration → Composants Windows →*

*Compatibilité des applications 16 bits → Empêcher l'accès aux applications 16 bits.*

---

To defend against brute force attacks in AppLocker, where malware tries to execute itself in multiple directories until it finds an allowed path, you can implement custom AppLocker rules that minimize this attack surface. Here's how:

---

Les règles créées pour les *packaged apps* (comme les applis du Microsoft Store) et leurs installateurs doivent obligatoirement utiliser des conditions basées sur l'éditeur (publisher).

Pourquoi ? Parce que Windows n'autorise pas les packaged apps non signées, donc on est sûr qu'elles ont un éditeur vérifiable.

---