

Inhalt

Applocker DefaultRule-Bypass.....	1
Worum geht es?	1
LAB-Umgebung	1
Konfiguration und Test von Applocker	1
Anwendungsausführung ohne Applocker	1
Basis-Konfiguration des Applocker-Services	2
Anwendungsausführung mit Applocker	5
Hacking: Applocker DefaultRuleBypass	5
Konfiguration von Applocker zum Verhindern des Bypasses	6
Test der Applocker DefaultRule-Bypass-Protection.....	10
Bewertung	10

Applocker DefaultRule-Bypass

Worum geht es?

Applocker gehört seit Windows 7 zu den Enterprise-Features. Dahinter steckt ein Dienst im Betriebssystem, der basierend auf einem Regelwerk für jeden einen Benutzer entscheidet, ob dieser eine Anwendung starten darf oder nicht.

Natürlich ist dieser Dienst per Default nicht aktiv. Er muss durch ein vorher erarbeitetes Regelwerk über Gruppenrichtlinien konfiguriert, getestet und aktiviert werden. Bei dem Regeldesign helfen die Standardregeln. Diese schränken Standardbenutzer auf alle Anwendungen in den Programmverzeichnissen C:\Programme und c:\Programme(x86) sowie unter dem Systemverzeichnis C:\Windows ein. Da hier ein Benutzer keine Schreibrechte besitzt, können somit nur Anwendungen ausgeführt werden, die von Administratoren installiert und somit genehmigt wurden.

In meinem WSHoWto zeige ich einen interessanten Bypass zu dem Regelwerk auf – und natürlich auch, wie man diesen schließt. 😊

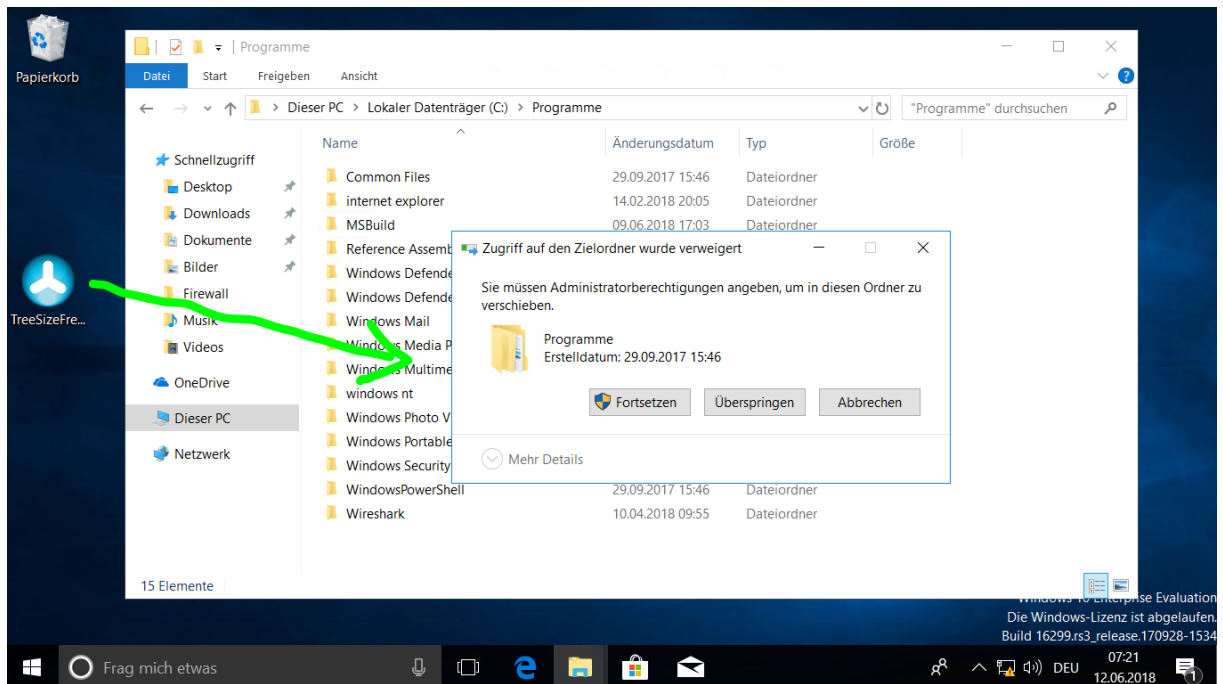
LAB-Umgebung

In meinem steht ein Client mit Windows 10 Enterprise und ein DomainController mit Windows Server 2016 bereit. Mit den Benutzerkonten Tessa.Test (Standardbenutzer) und dem DomainAdministrator versuche ich die Anwendungen zu starten.

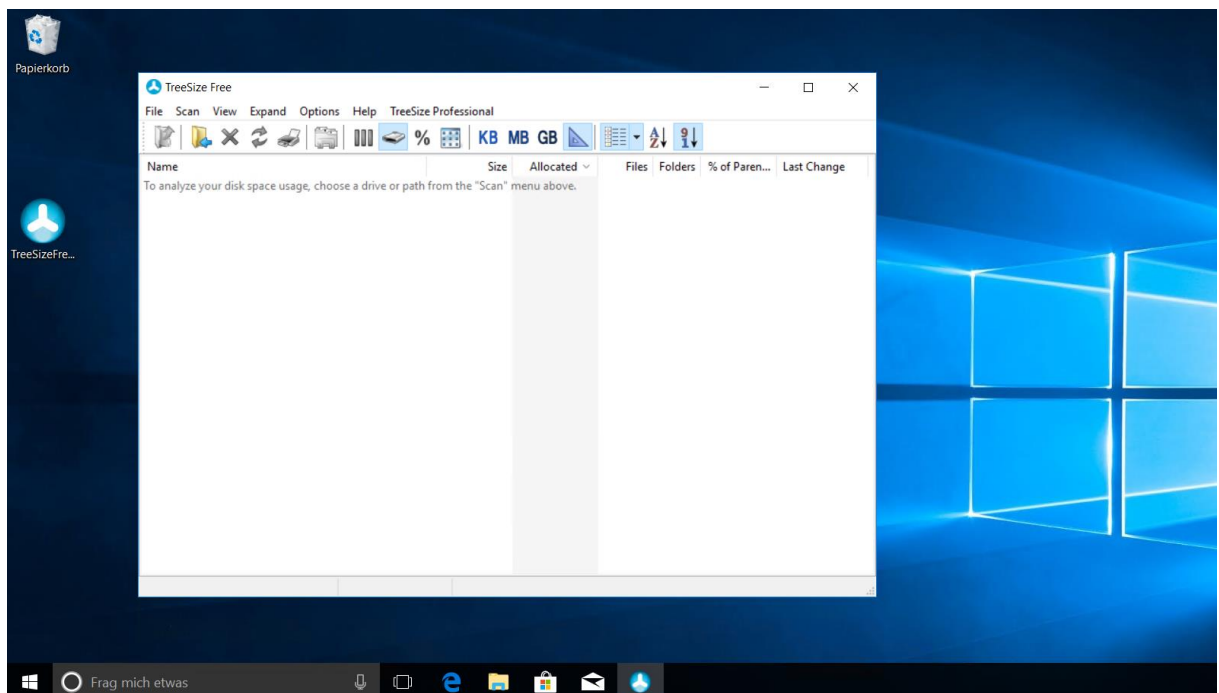
Konfiguration und Test von Applocker

Anwendungsausführung ohne Applocker

Tessa hat eine Anwendung heruntergeladen und möchte diese im Programmverzeichnis speichern. Da sie keine administrativen Rechte besitzt verweigert das Betriebssystem die Ablage:



Viele Anwendungen benötigen aber keine Installation – es handelt sich dabei um PortableApps. Sie lassen sich direkt von dem Verzeichnis aus starten, in dem sie gespeichert sind:

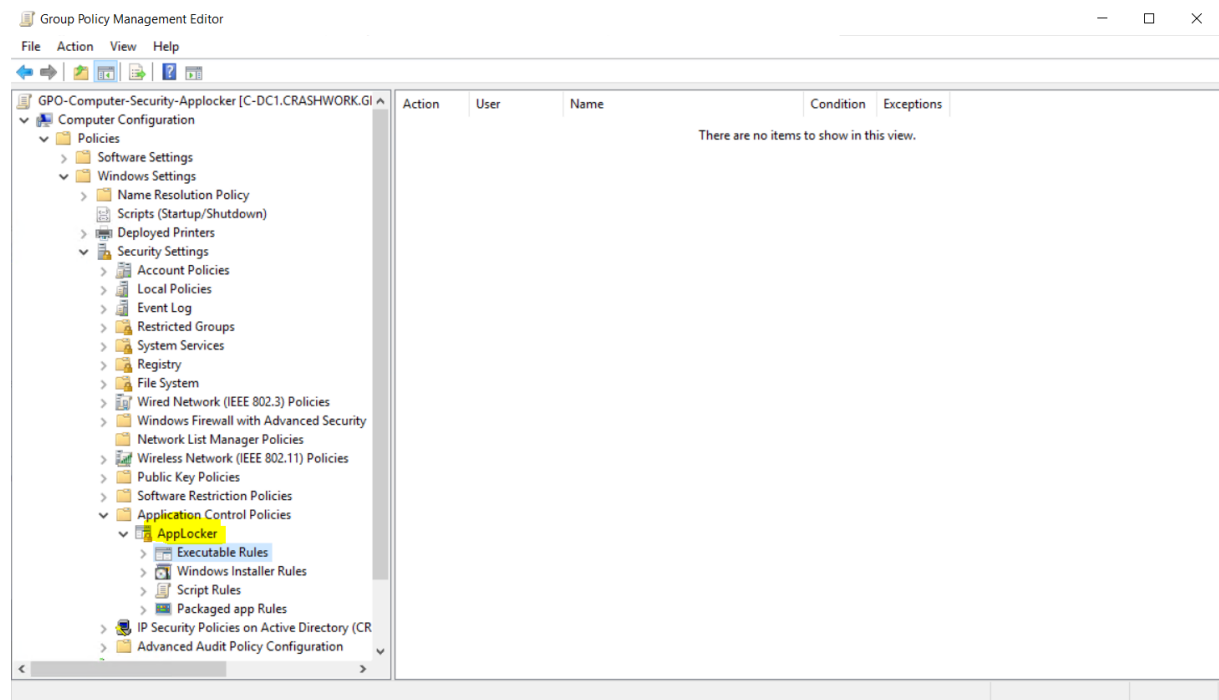


Standardbenutzer können also recht einfach mit so genannten PortableApps eigene Anwendungen in eine gesicherte Infrastruktur einbringen.

Ebenso kann aber auch ein Schadcode im Benutzerkontext den Weg auf die Festplatte finden und sich dort ausführen (verlasst euch mal bitte nicht auf den Virenschanner... 😊)

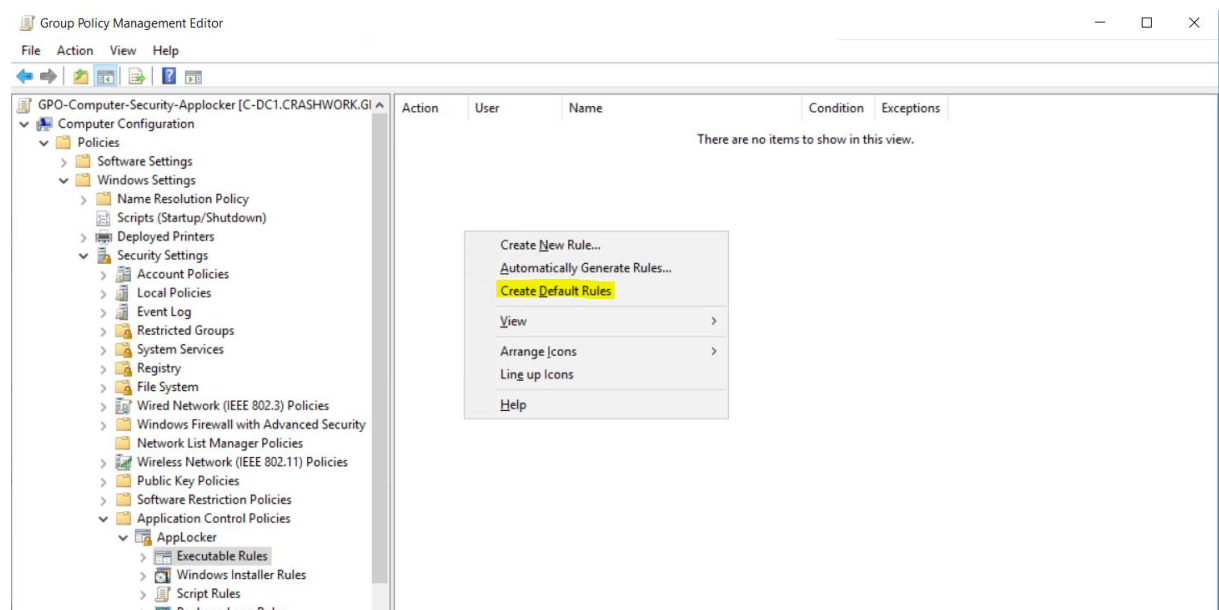
Basis-Konfiguration des Applocker-Services

Mit Applocker kann dies recht einfach verhindert werden. Dafür bietet sich eine zentrale Gruppenrichtlinie auf dem DomainController an:

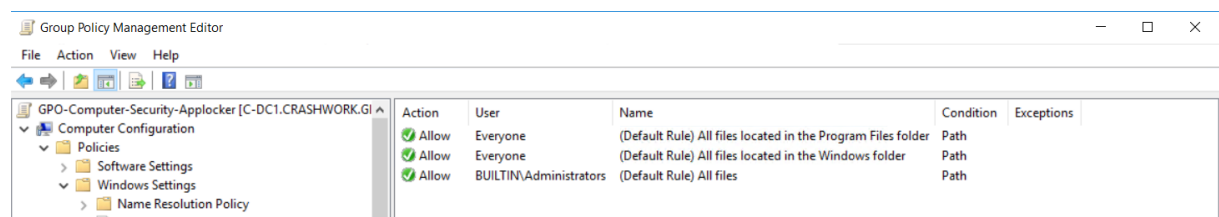


Mit dieser Richtlinie können Regeln für das Setup und die Ausführung von Anwendungen ebenso definiert werden wie Scriptregeln und seit Windows 8 auch Regeln für Windows Store Apps.

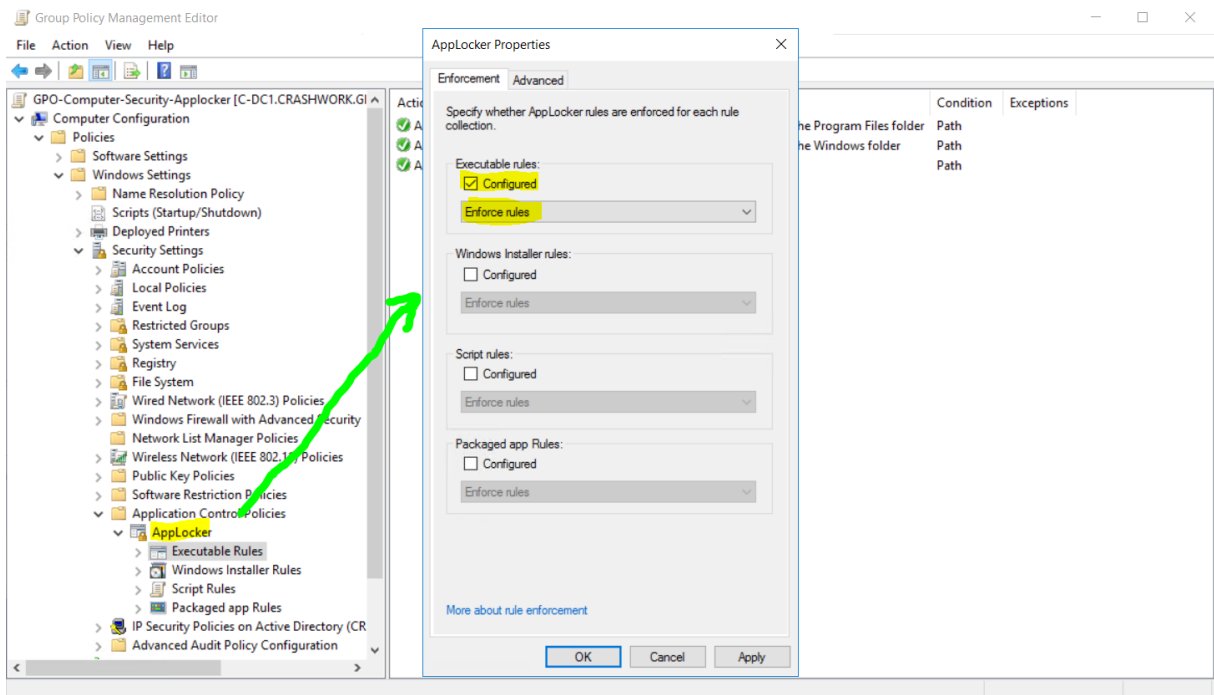
In meinem Fall genügt mir die Steuerung der Executables, da die Installation von Anwendungen für Standardbenutzer nicht möglich ist. Hier gibt es nun die Option der Standardregeln:



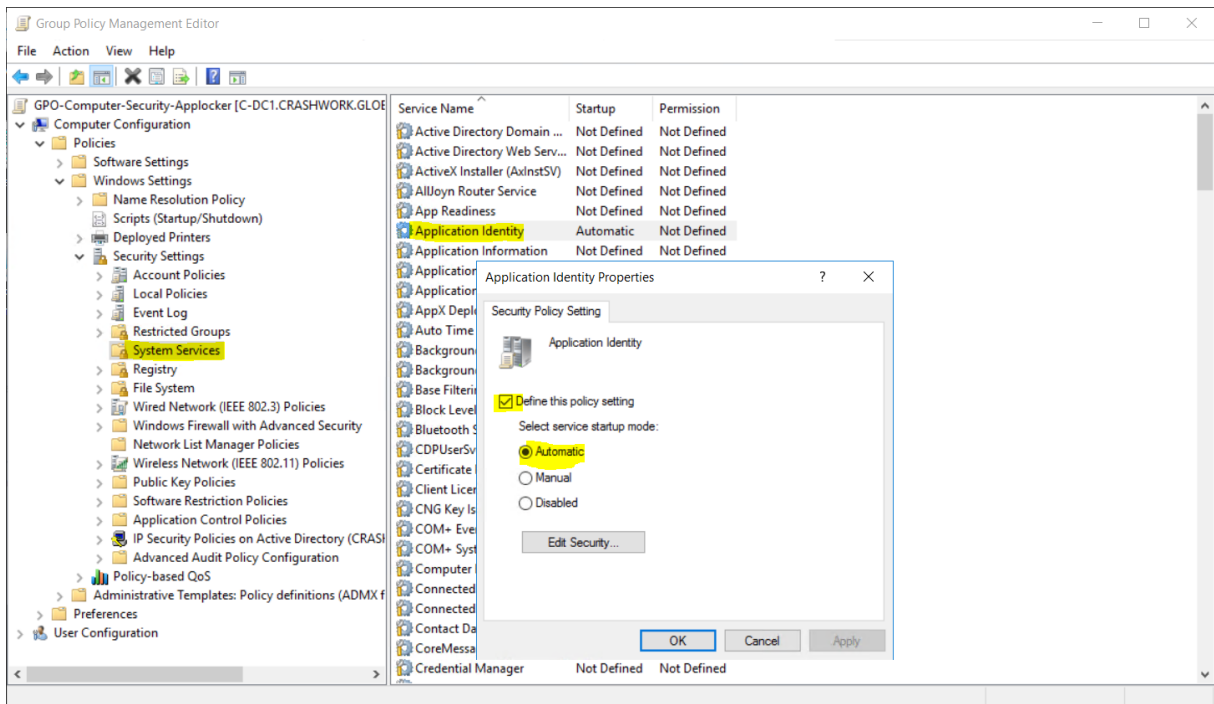
Damit wird folgendes geregelt:



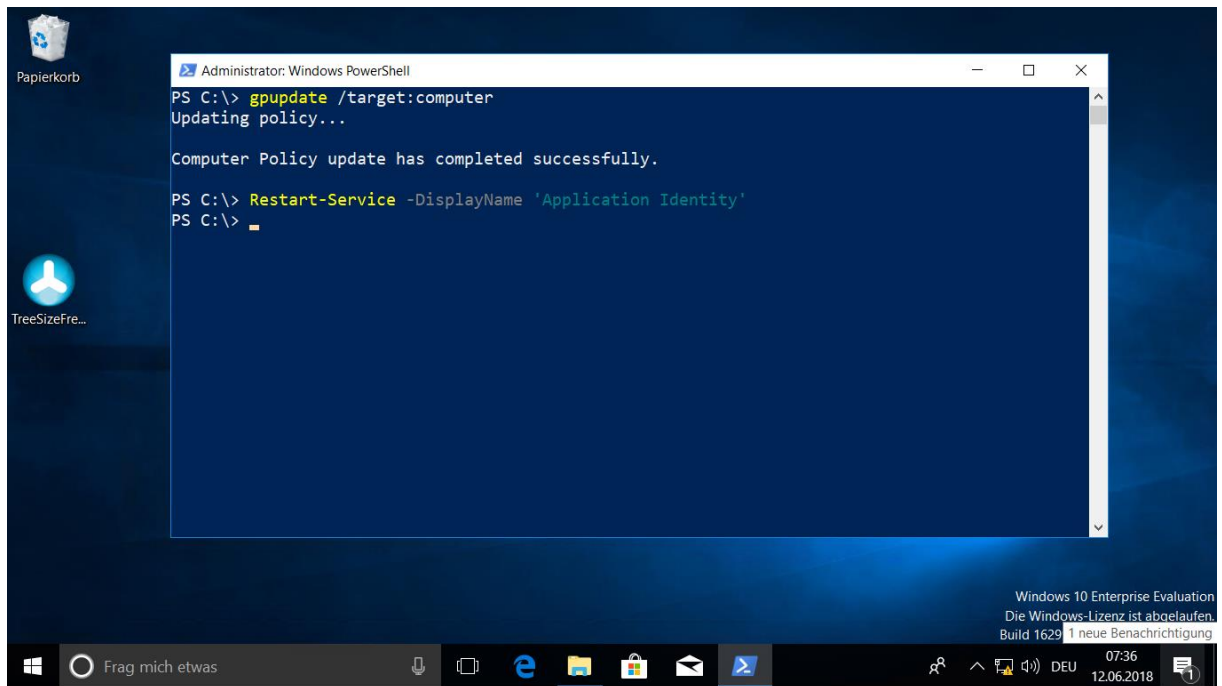
Zum Scharfschalten der Regelwerke ist noch die Einstellung des Betriebsmodus erforderlich. Hier kann AuditOnly oder Enforce gewählt werden:



In der gleichen GPO kann nun noch der Service für AppLocker automatisch gestartet werden:

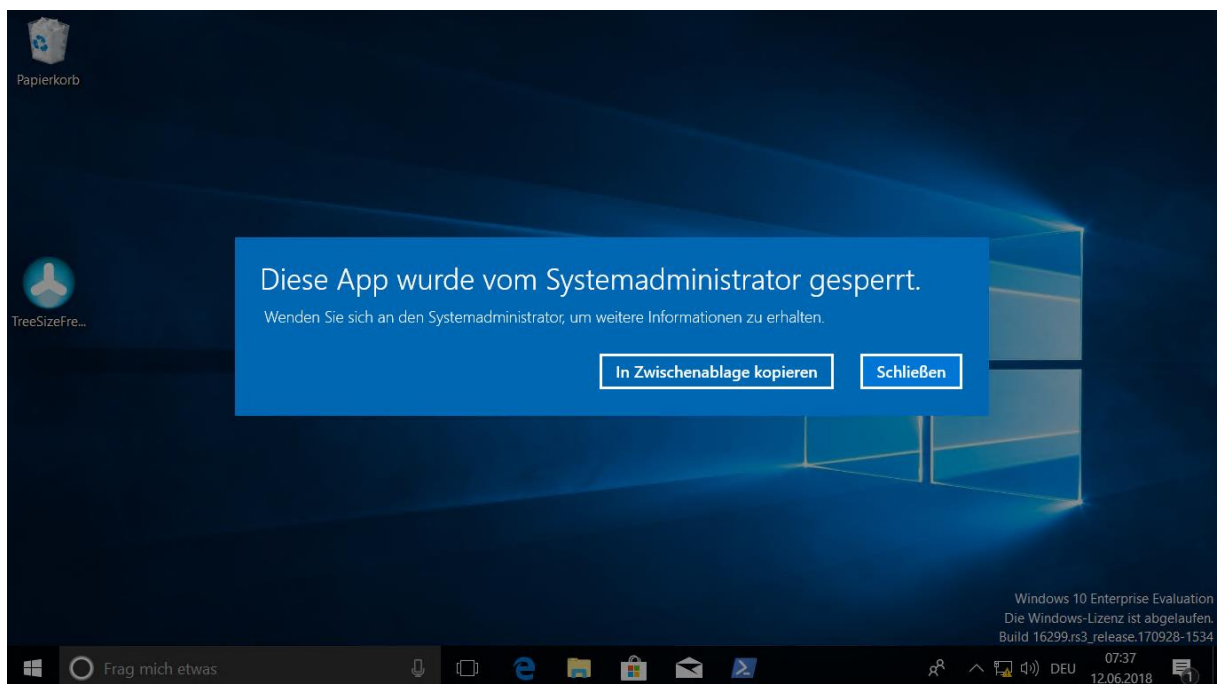


Ein Mappen der GPO und ein gpupdate später sind die Einstellungen am Client angekommen. Ggf. benötigt der Service noch ein/zwei Neustarts. In meinem LAB beschleunige ich das etwas:



Anwendungsausführung mit Applocker

Mit der Basiskonfiguration kann Tessa ihre Anwendung nicht mehr starten, da diese im Pfad c:\Users\Tessa.test\Desktop liegt und für dieses Verzeichnis keine Ausnahme zur Ausführung erstellt wurde. Die Anwendung wird blockiert:



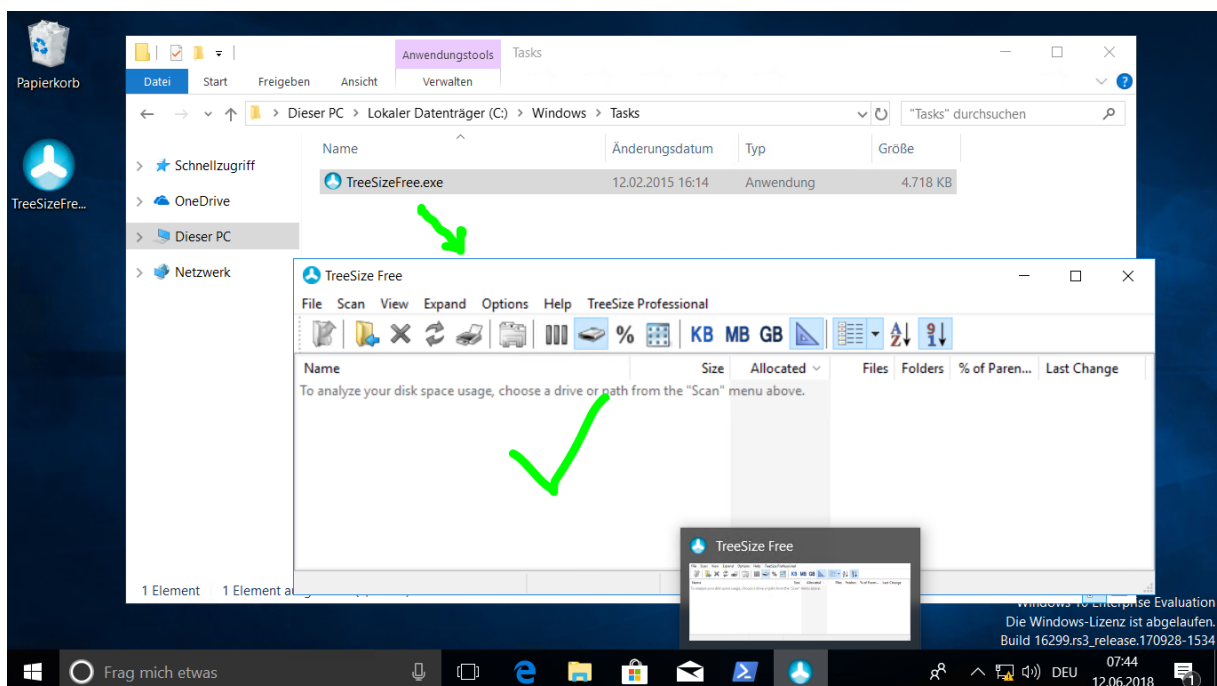
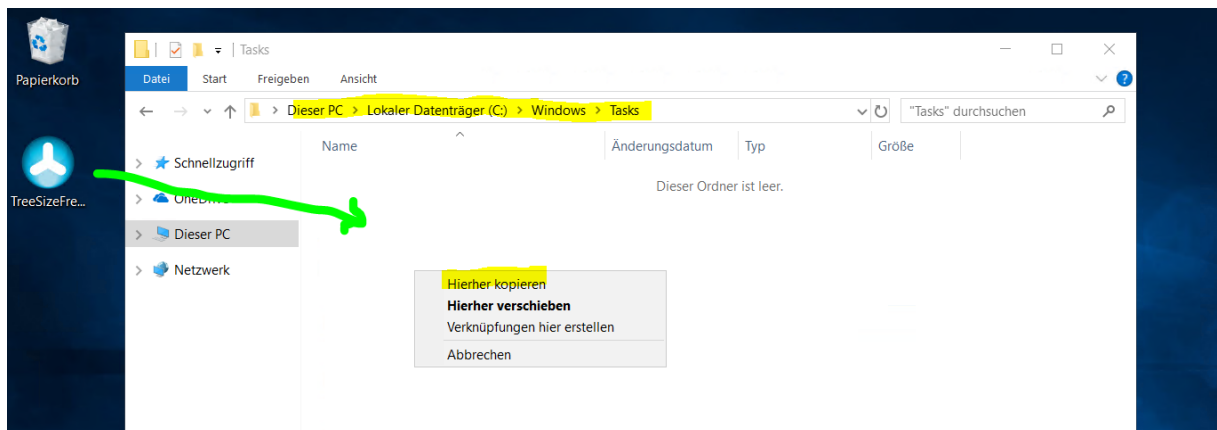
Hacking: Applocker DefaultRuleBypass

Ist der Rechner nun sicher? Na klar, denn Tessa hat in den Verzeichnissen, die für Applocker konfiguriert wurde keine Schreibrechte, da sie kein Administrator ist... Wirklich?

Die Verzeichnisberechtigungen in den Untiefen des Systemlaufwerkes sollten doch eigentlich so strikt sein, dass die Regel „kein Admin kein Schreibrecht“ vorausgesetzt werden kann. Tatsächlich ist es aber (wie üblich aus Kompatibilitätsgründen) leider anders. ☹️

Ein Beispiel: Tessa darf für ihren eigenen Account geplante Aufgaben im Betriebssystem erstellen. Diese lagen früher als Dateien unter c:\Windows\Tasks. Ein Standardbenutzer hat(te) dort also Schreibrechte. Und c:\Windows\Tasks liegt unter

C:\Windows – der Applocker lässt dort also Anwendungsstarts per default zu. Funktioniert das heute mit Windows 10 (1709) immer noch?



... das funktioniert! OK, Tessa weiß das bestimmt nicht. Aber ein Angreifer kann das durchaus wissen und aktiv ausnutzen!

Konfiguration von Applocker zum Verhindern des Bypasses

Nun stellt sich bei mir die Frage: Gibt es noch weitere Verzeichnisse innerhalb der Standardregel-Verzeichnisse von Applocker, in denen ein Standardbenutzer Schreibrechte hat? Mit einem einfachen Script versuche ich über die PowerShell, in jedem Unterverzeichnis als Standardbenutzer eine Datei zu schreiben. Wenn dies gelingt, dann ist das Verzeichnis ein Bypass-Kandidat. Das ist mein Code:

```
trap {'continue'}
$RootFolders = @()
$RootFolders += Get-ChildItem -Path 'C:\' -Directory |
    Where-Object {$_.fullname -notlike 'c:\users*'} |
    Select-Object -ExpandProperty fullname
$RootFolders += 'c:\Programdata'

Get-ChildItem $RootFolders -Directory -Recurse -ErrorAction SilentlyContinue |
    ForEach-Object {
        $Pfad = $_.FullName

        try {
```



```
'test' | out-file -FilePath ($Pfad + '\test.txt')
Write-Host $Pfad -ForegroundColor Green
Remove-Item -Path ($Pfad + '\test.txt') -ErrorAction stop
}
catch { }
}
```

Der erste Block definiert die Hauptverzeichnisse, in denen schreibbare Verzeichnisse gesucht werden sollen. Durch das Try-Catch werden nur Verzeichnispfade gelistet, in denen das Schreiben der test.txt erfolgreich war. ACHTUNG: ggf. hat ein Standardbenutzer ein Schreibrecht aber kein Löschrecht – es können also durchaus Testdateien im System verbleiben

Nach einer kurzen Laufzeit kommt folgendes Ergebnis zustande:

```
C:\Windows\Tasks
C:\Windows\Temp
C:\Windows\tracing
C:\Windows\Registration\CRMLLog
C:\Windows\System32\FxsTmp
C:\Windows\System32\com\dmp
C:\Windows\System32\spool\PRINTERS
C:\Windows\System32\spool\SERVERS
C:\Windows\SysWOW64\FxsTmp
C:\Windows\SysWOW64\com\dmp
C:\Programdata\AntiMalwareTest
C:\Programdata\Microsoft OneDrive
C:\Programdata\USOShared
C:\Programdata\AntiMalwareTest\4_12_2018
C:\Programdata\Microsoft\DeviceSync
C:\Programdata\Microsoft\Crypto\DSS\MachineKeys
C:\Programdata\Microsoft\Crypto\RSA\MachineKeys
C:\Programdata\Microsoft\DataMart\PaidWifi
C:\Programdata\Microsoft\Windows\WER\ReportArchive
C:\Programdata\Microsoft\Windows\WER\ReportQueue
C:\Programdata\Microsoft\Windows\WER\Temp
C:\Programdata\Microsoft\Windows\WER\ReportQueue\NonCritical_Update;_ad95924aceca5298acb0fdb6ed2130143d281e_00000000_0cbf19f4
C:\Programdata\Microsoft\Windows\WER\ReportQueue\NonCritical_Update;_ad95924aceca5298acb0fdb6ed2130143d281e_00000000_0e87e445
C:\Programdata\Microsoft\Windows\WER\ReportQueue\NonCritical_Update;_ad95924aceca5298acb0fdb6ed2130143d281e_00000000_17a4a181
C:\Programdata\Microsoft OneDrive\setup
C:\Programdata\USOShared\Logs
```

Das hat mich überrascht!!! Besonders perfide finde ich den Ordner AntiMalwareTest... 😏 Aber da sind auch Verzeichnisse gelistet, die der Applocker abdeckt – z.B. c:\Programdata. Also erweitere ich meinen Test im Script auf das Erstellen und Starten einer Anwendung in einem Verzeichnis:

```
$CleanupDirs = @()

trap {'continue'}
$RootFolders = @()
$RootFolders += Get-ChildItem -Path 'C:\' -Directory |
    Where-Object {$_.fullname -notlike 'c:\users*'} |
    Select-Object -ExpandProperty fullname
$RootFolders += 'c:\Programdata'

write-host "Verzeichnisse mit Recht 'Schreiben und Ausfuehren' ohne Applocker-Schutz:" -
    ForegroundColor White

Get-ChildItem $RootFolders -Directory -Recurse -ErrorAction SilentlyContinue | ForEach-Object {
    $Pfad = $_.FullName

    try {
        Copy-Item -Path C:\Windows\System32\cmd.exe "$Pfad\ApplockerTest.exe" -ErrorAction Stop

        Start-Process -FilePath "$Pfad\ApplockerTest.exe" -WorkingDirectory $Pfad
        Stop-Process -Name ApplockerTest -ErrorAction Stop

        # Platzhalter für ApplockerPolicyCreator

        $Error.clear()
        Remove-Item -Path "$Pfad\ApplockerTest.exe" -ErrorAction SilentlyContinue
        if ($Error) {
```

```

        $CleanupDirs += $Pfad
    } else {
        Write-Host "    $Pfad" -ForegroundColor Green
    }
}
catch {
    if (Test-Path -Path "$Pfad\ApplockerTest.exe") {
        $Error.clear()
        Remove-Item -Path "$Pfad\ApplockerTest.exe" -ErrorAction SilentlyContinue
        if ($Error) { $CleanupDirs += $Pfad }
    }
}
}
write-host "Verzeichnisse mit Recht 'Erstellen ohne Ausfuehren' (manuell bereinigen):" -
ForegroundColor White
$CleanupDirs | ForEach-Object { Write-Host "    $_" -ForegroundColor Yellow }

```

Das Ergebnis ist nicht mehr ganz so dramatisch. Die grünen Einträge sind fehlerhaft vom Applocker geschützt. Die gelben lassen zwar das Erstellen von Dateien zu – nicht aber die Ausführung (keine Execute-Berechtigung):

```

Verzeichnisse mit Recht 'Schreiben und Ausfuehren' ohne Applocker-Schutz:
C:\Windows\Tasks
C:\Windows\Temp
C:\Windows\System32\Tasks
C:\Windows\System32\Microsoft\Crypto\RSA\Machinekeys
C:\Windows\System32\spool\drivers\color
C:\Windows\System32\spool\drivers\color
C:\Windows\System32\spool\drivers\color
Verzeichnisse mit Recht 'Erstellen ohne Ausfuehren' (manuell bereinigen):
C:\Windows\tracing
C:\ProgramData\Microsoft\User Account Pictures
C:\ProgramData\Microsoft\NetFramework\BreadcrumbStore
PS C:\Users\tessa.test>

```

OK, für alle gefundenen Verzeichnisse kann nun eine Applocker-Deny-Regel erstellt werden. Die Verzeichnisberechtigungen selber würde ich nicht anfassen. Wer weiß, was danach alles nicht mehr funktioniert... Die Policies möchte ich aber nicht von Hand zusammenstellen. Dafür gibt es doch die PowerShell ☺ Im Script gibt es einen Platzhalter-Kommentar. Hier kann folgender Code eingefügt werden, um für die gefundenen Verzeichnisse eine Applocker-Regel im XML-Dateiformat zu generieren:

```

$Filename = $Pfad -replace '\\', '-' -replace ':'

$xml = Get-AppLockerFileInformation -Path "$Pfad\ApplockerTest.exe" |
    New-AppLockerPolicy -RuleType path -User everyone -Xml
$xml = $xml -replace '"Allow"', '"Deny"' -replace "ApplockerTest.exe", '*'
$xml = $xml -replace 'Description=""', 'Description="verhindert Applocker-Bypass"'
$xml | Out-File -FilePath "$XMLVerzeichnis\$Filename.xml"

```

Zusätzlich muss **vor** dem Code das Verzeichnis für die XML-Dateien erstellt werden:

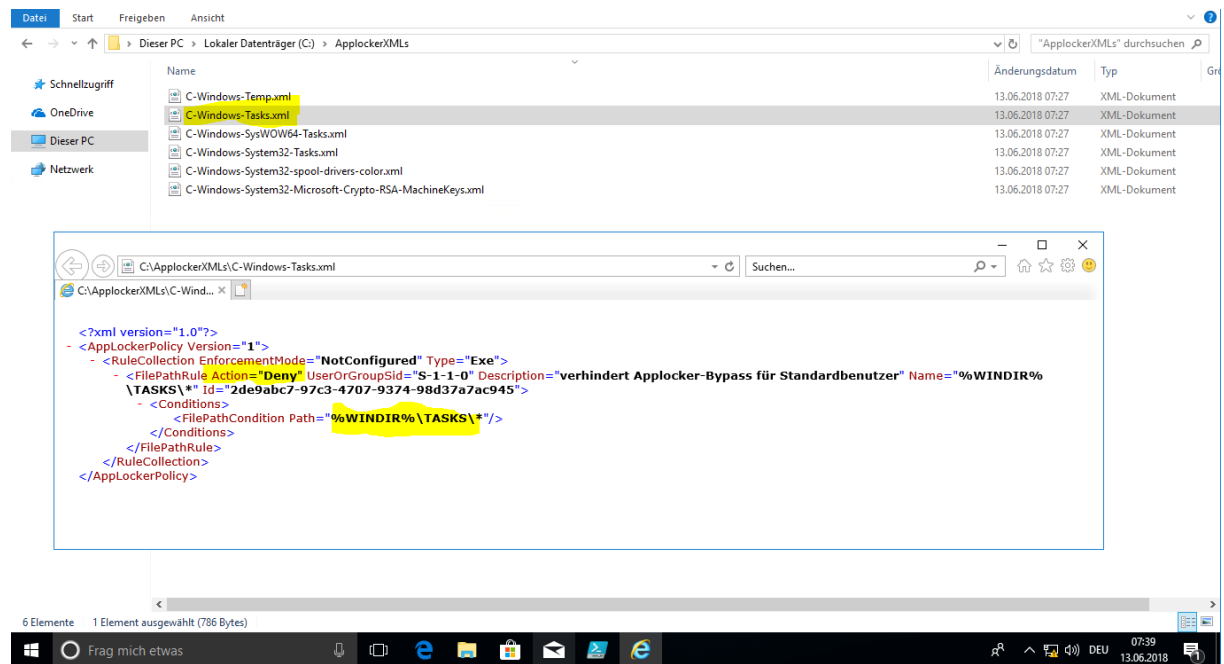
```

$xmlVerzeichnis = "c:\ApplockerXMLs"

if ((Test-Path -Path $xmlVerzeichnis) -eq $false) {
    New-Item -Path $xmlVerzeichnis -ItemType directory | Out-Null
}

```

Dann erstellt der Code zur Laufzeit diese Dateien. Jede Datei ist eine Applocker-DenyRule für ein gefundenes Verzeichnis:



Auf dem DomainController können diese XML-Dateien dann mit der PowerShell direkt in die bereits vorhandene GPO übernommen werden:

```
# Variablen
$BackupDir = "C:\admin\GPOBackup"
$XMLVerzeichnis = "c:\ApplockerXMLs"

# suche und sichere die GPO auf dem PDC
if ((Test-Path -Path $BackupDir) -eq $false) {
    New-Item -Path $BackupDir -ItemType directory | Out-Null
}

$GPO = Get-GPO -All | Where-Object { $_.Displayname -eq 'GPO-Computer-Security-Applocker' }
Backup-GPO -Guid $GPO.id -Path $BackupDir | Out-Null
$GPO = $GPO.path
$PDC = (Get-ADDomain).PDCEmulator

# erstelle für jedes Bypass-XML eine Deny-Ausnahme in der GPO
Get-ChildItem -Path $XMLVerzeichnis | ForEach-Object {
    Set-AppLockerPolicy -XmlPolicy $_.FullName -Ldap "LDAP://$PDC/$GPO" -Merge
}
```

Die GPO ist dann entsprechend erweitert:

Policy	Setting		
Enforce rules of this type	True		
Action	User	Name	Rule Type
Deny	Everyone	%WINDIR%\TASKS*	Path
Deny	Everyone	%WINDIR%\TEMP*	Path
Allow	Everyone	(Default Rule) All files located in the Program Files folder	Path
Allow	Everyone	(Default Rule) All files located in the Windows folder	Path
Deny	Everyone	%SYSTEM32%\Tasks*	Path
Deny	Everyone	%SYSTEM32%\SPOOL\DRIVERS\COLOR*	Path
Deny	Everyone	%SYSTEM32%\MICROSOFT\CRYPTO\RSA\MACHINEKEYS*	Path
Allow	BUILTIN\Administrators	(Default Rule) All files	Path

Achtung: In meinem Script habe ich nicht getestet, ob in den gefundenen Verzeichnissen noch andere Anwendungen liegen, die von Microsoft dort für bestimmte Aufgaben abgelegt wurden. Im realen Leben prüft ihr das bitte genauer!

Test der Applocker DefaultRule-Bypass-Protection

Mit dem Script auf dem Testclient kann nach der Aktualisierung der Gruppenrichtlinie erneut getestet werden:

```
Verzeichnisse mit Recht 'Schreiben und Ausführen' ohne Applocker-Schutz:
Verzeichnisse mit Recht 'Erstellen ohne Ausführen' (manuell bereinigen):
C:\Windows\Temp
C:\Windows\tracing
C:\Windows\System32\spool\drivers\color
C:\Windows\System32\Tasks
C:\ProgramData\Microsoft\User Account Pictures
C:\ProgramData\Microsoft\NetFramework\BreadcrumbStore
PS C:\Users\tessa.test> |
```

Es sind keine Verzeichnisse mit dem Verzeichnisrecht Schreiben+Ausführen UND einer Applocker-AllowRule vorhanden 😊

Bewertung

Applocker ist eine wichtige Schutzkomponente. Aber sie ist nicht unfehlbar. Eine Vielzahl von Angriffsmöglichkeiten wehrt sie für Standardbenutzer ab. Mit einer bewussten Konfiguration kann der Schutz verbessert werden.

Dennoch empfehle ich regelmäßige Audits, ob der konfigurierte Schutz immer noch passt. Vielleicht kommt morgen mit einem Update ein weiteres, ungeschütztes Verzeichnis dazu?