GPO Permissions/GPO Files

Description

A Group Policy Object (GPO) is a virtual collection of policy settings that has a unique name. `GPOs` are the most widely used configuration management tool in Active Directory. Each GPO contains a collection of zero or more policy settings. They are linked to an `Organizational Unit` in the AD structure for their settings to be applied to objects that reside in the OU or any child OU of the one to which the GPO is linked. GPOs can be restricted to which objects they apply by specifying, for example, an AD group (by default, it applies to Authenticated Users) or a WMI filter (e.g., apply only to Windows 10 machines).

When we create a new GPO, only Domain admins (and similar privileged roles) can modify it. However, within environments, we will encounter different delegations that allow less privileged accounts to perform edits on the GPOs; this is where the problem lies. Many organizations have GPOs that can modify 'Authenticated Users' or 'Domain Users', which entails that any compromised user will allow the attacker to alter these GPOs. Modifications can include additions of start-up scripts or a scheduled task to execute a file, for example. This access will allow an adversary to compromise all computer objects in the OUs that the vulnerable GPOs are linked to.

Similarly, administrators perform software installation via GPOs or configure start-up scripts located on network shares. If the network share is misconfigured, an adversary may be able to replace the file to be executed by the system with a malicious one. The GPO may have no misconfigurations in these scenarios, just misconfigured NTFS permissions on the files deployed.

Attack

No attack walkthrough is available here - it is a simple GPO edit or file replacement.

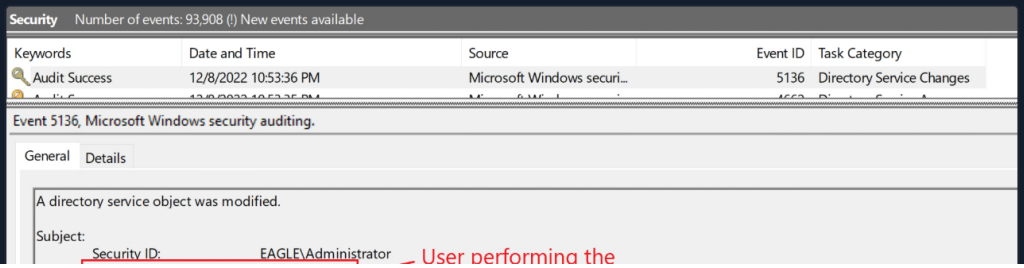
Prevention

One way to prevent this attack is to lock down the GPO permissions to be modified by a particular group of users only or by a specific account, as this will significantly limit the ability of who can edit the GPO or change its permissions (as opposed to everybody in Domain admins, which in some organizations can easily be more than 50). Similarly, never deploy files stored in network locations so that many users can modify the share permissions.

We should also review the permissions of GPOs actively and regularly, with the option of automating a task that runs hourly and alerts if any deviations from the expected permissions are detected.

Detection

Fortunately, it is straightforward to detect when a GPO is modified. If Directory Service Changes auditing is enabled, then the event ID `5136` will be generated:

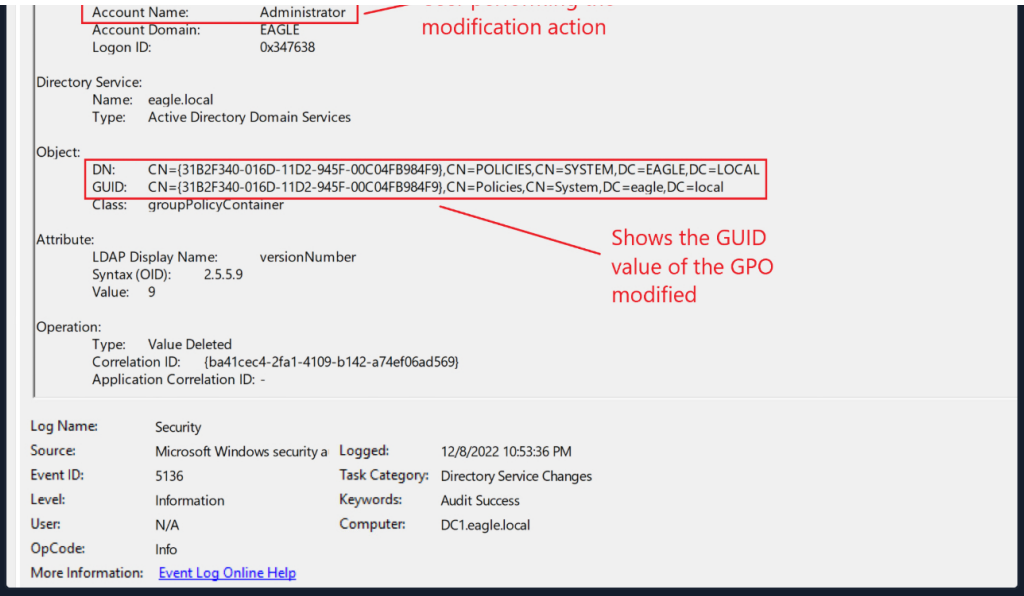


My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left



From a defensive point of view, if a user who is `not` expected to have the right to modify a GPO suddenly appears here, then a red flag should be raised.

Honeypot

A common thought is that because of the easy detection methods of these attacks, it is worth having a misconfigured GPO in the environment for threat agents to abuse; this is also true for a deployed file as they can be continuously monitored for any change to the file (e.g., constantly checking if the hash value of the file has not changed). However, implementing these techniques is only recommended if an organization is mature and proactive in responding to high/critical vulnerabilities; this is because if, in the future, an escalation path is discovered via some GPO modification, unless it is possible to mitigate it in real-time, the `trap` backfires to become the weakest point.

However, when implementing a honeypot using a misconfigured GPO, consider the following:

- `GPO is linked to non-critical servers only.`
- `Continuous automation is in place for monitoring modifications of GPO. - - If the GPO file is modified, we will disable the user performing the modification immediately.`
- `The GPO should be automatically unlinked from all locations if a modification is detected.`

Consider the following script to demonstrate how PowerShell can automate this. In our case, the honeypot GPO is identified by a GUID value, and the action desired is to disable the account(s) associated with this change. The reason for potentially multiple accounts is that we will execute the script every 15 minutes as a scheduled task. So, if numerous compromised users were used to modify the GPO in this time frame, we will disable them all instantly. The script has a commented-out section that can be used for sending an email as an alert, but for a PoC, we will display the output on the command line:

Code: powershell

```powershell
# Define filter for the last 15 minutes
$TimeSpan = (Get-Date) - (New-TimeSpan -Minutes 15)

# Search for event ID 5136 (GPO modified) in the past 15 minutes
$Logs = Get-WinEvent -FilterHashtable @{LogName='Security';id=5136;StartTime=$TimeSpan} -ErrorActio
Where-Object {$_.Properties[8].Value -match "CN={73C66DBB-81DA-44D8-BDEF-20BA2C27056D},CN=POLICIES,


if($Logs){
    $emailBody = "Honeypot GPO '73C66DBB-81DA-44D8-BDEF-20BA2C27056D' was modified`r`n"
    $disabledUsers = @()
    ForEach($log in $logs){
        If(((Get-ADUser -identity $log.Properties[3].Value).Enabled -eq $true) -and ($log.Propertie
            Disable-ADAccount -Identity $log.Properties[3].Value
            $emailBody = $emailBody + "Disabled user " + $log.Properties[3].Value + "`r`n"
```

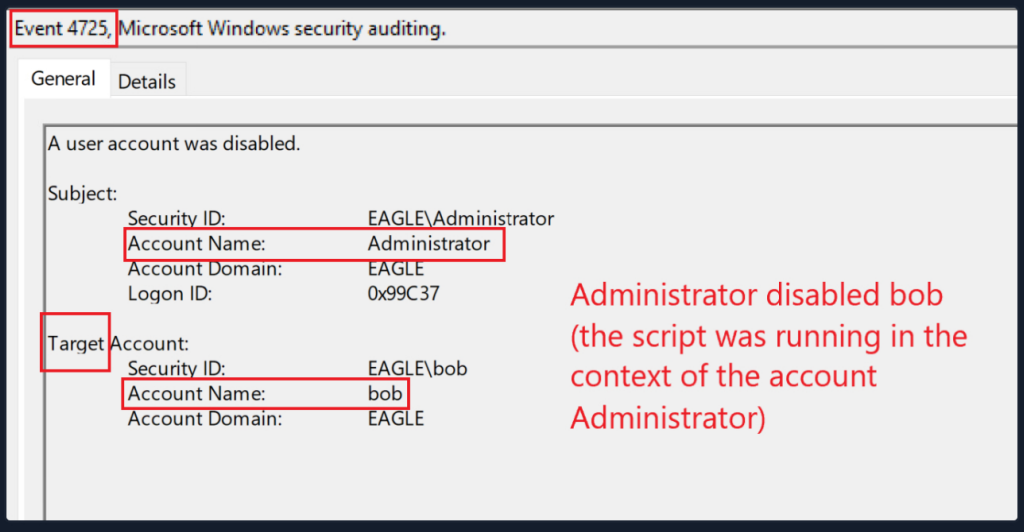
```
            $disabledUsers += $log.Properties[3].Value
        }
    }
    # Send an alert via email - complete the command below
    # Send-MailMessage
    $emailBody
}
```

We will see the following output (or email body if configured) if the script detects that the honeypot GPO was modified:

GPO Permissions/GPO Files

```
PS C:\scripts> # Define filter for the last 15 minutes
$TimeSpan = (Get-Date) - (New-TimeSpan -Minutes 15)

# Search for event ID 5136 (GPO modified) in the past 15 minutes
$Logs = Get-WinEvent -FilterHashtable @{LogName='Security';id=5136;StartTime=$TimeSpan} -ErrorActio
Where-Object {$_.Properties[8].Value -match "CN={73C66DBB-81DA-44D8-BDEF-20BA2C27056D},CN=POLICIES,


if($Logs){
    $emailBody = "Honeypot GPO '73C66DBB-81DA-44D8-BDEF-20BA2C27056D' was modified`r`n"
    $disabledUsers = @()
    ForEach($log in $logs){
        # Write-Host "User performing the modification is " $log.Properties[3].Value
        If(((Get-ADUser -identity $log.Properties[3].Value).Enabled -eq $true) -and ($log.Propertie
            Disable-ADAccount -Identity $log.Properties[3].Value
            $emailBody = $emailBody + "Disabled user " + $log.Properties[3].Value + "`r`n"
            $disabledUsers += $log.Properties[3].Value
        }
    }
    # Send an alert via email
    # Send-MailMessage
    $emailBody
}

Honeypot GPO '73C66DBB-81DA-44D8-BDEF-20BA2C27056D' was modified
Disabled user bob


PS C:\scripts>
```

As we can see above, the user `bob` was detected modifying our honeypot GPO and is, therefore, disabled. Disabling the user will then create an event with ID `4725`:



VPN Servers

⚠ **Warning:** Each time you "Switch", your connection keys are regenerated and you must re-download your VPN connection file.

All VM instances associated with the old VPN Server will be terminated when switching to a new VPN server.

Existing PwnBox instances will automatically switch to the new VPN server.

US Academy 3 — Medium Load

PROTOCOL

UDP 1337 TCP 443

DOWNLOAD VPN CONNECTION FILE

Connect to Pwnbox

Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK 160ms

Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Cheat Sheet

Download VPN Connection File

Target(s): Click here to spawn the target system!

RDP to with user "bob" and password "Slavi123"

+ 0 From WS001 RDP again into DC1 (172.16.18.3) as 'htb-student:HTB_@cademy_stdnt!' and abuse GPO directly. Once completed type DONE as the answer

DONE

Submit

Previous Next

Mark Complete & Next