A Red Teamer's Guide to GPOs and OUs

Published: 2018-04-02 · Archived: 2026-04-06 00:41:32 UTC

Intro

Active Directory is a vast, complicated landscape comprised of users, computers, and groups, and the complex, intertwining permissions and privileges that connect them. The initial release of BloodHound focused on the concept of derivative local admin, then BloodHound 1.3 introduced ACL-based attack paths. Now, with the release of BloodHound 1.5, pentesters and red-teamers can easily find attack paths that include abusing control of Group Policy, and the objects that those Group Policies effectively apply to.

In this blog post, I'll recap how GPO (Group Policy Object) enforcement works, how to use BloodHound to find GPO-control based attack paths, and explain a few ways to execute those attacks.

Prior Work

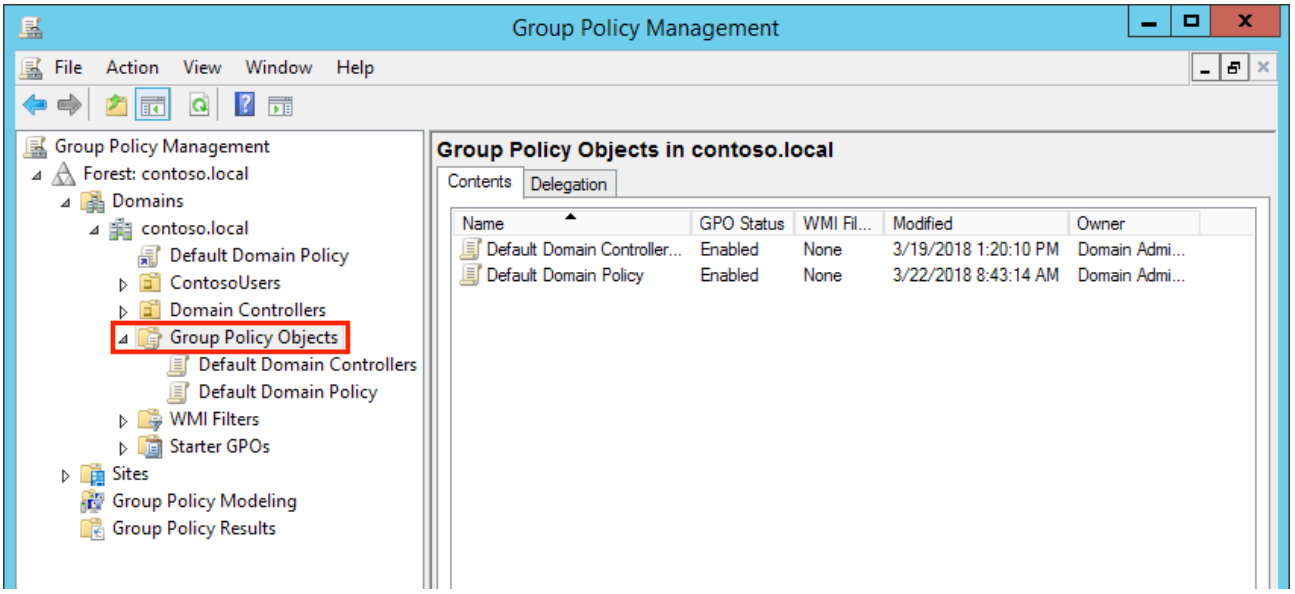
Lucas Bouillot and Emmanuel Gras included GPO control and OU structure in their seminal work, "Chemins de contrôle en environnement Active Directory". They used an attack graph to map which principals could take control of GPOs, and which OUs those GPOs applied to, then chased that down to the objects affected by those GPOs. We learned a lot from Lucas and Emannuel's white paper (in French), and I'd highly recommend you read it as well.

There are several important authors and resources we leaned on when figuring out how GPO works, in no particular order: the Microsoft Group Policy team's posts on TechNet, Sean Metcalf's work at adsecurity.org, 14-time Microsoft MVP "GPO Guy" Darren Mar-Elia, Microsoft's Group Policy functional specification, and last but certainly not least, Will Schroeder's seminal blog post on Abusing GPO Permissions. Special extra thanks to Darren Mar-Elia for answering a lot of my questions about Group Policy. Thanks, Darren! Other resources and references are linked at the bottom of this blog post.

The Moving Parts of Group Policy

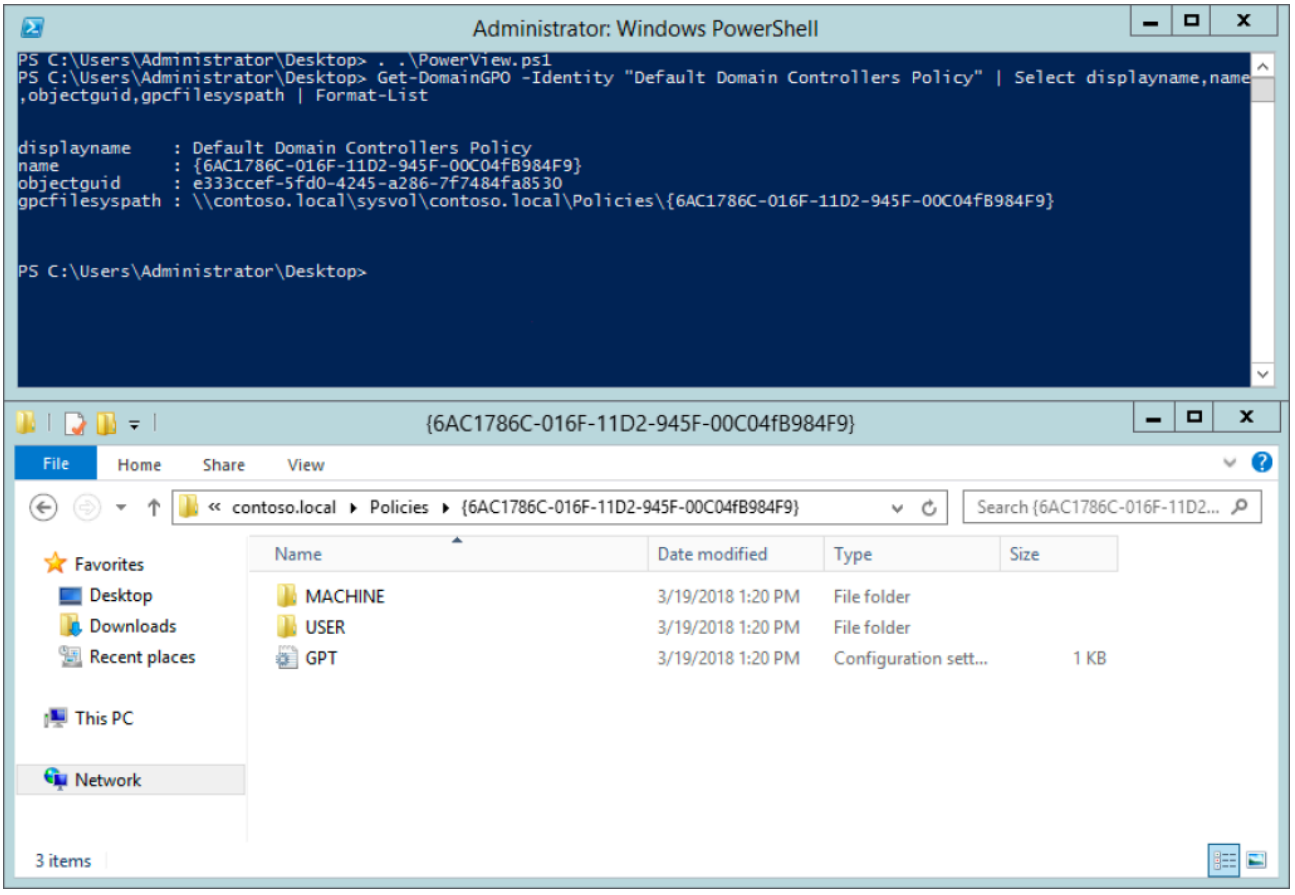
There's no two ways about it: GPO enforcement is a complicated beast with a lot of moving parts. With that said, let's start at the very basics with the vocabulary used in the rest of the post, and build up to explaining how those moving parts interact with one another:

GPO: A Group Policy Object. When an Active Directory domain is first created, two GPOs are created as well: "Default Domain Policy" and "Default Domain Controllers". GPOs contain sets of policies that affect computers and users. For example, you can use a GPO policy to control the Windows desktop background on computers. GPOs are visible in the Group Policy Management GUI here:



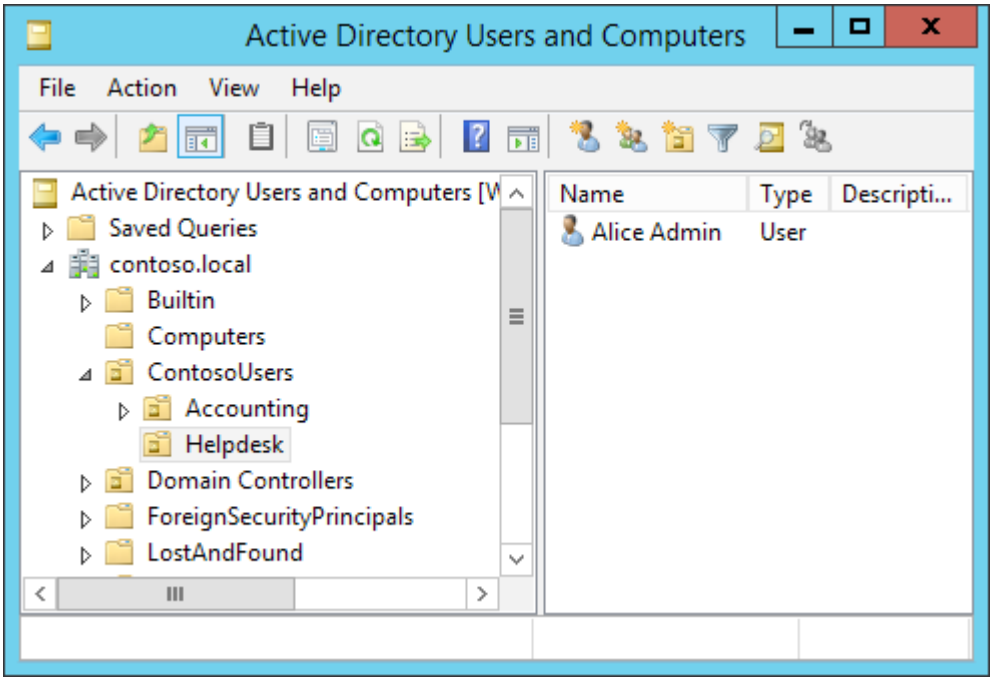
Above: The list of GPOs in our test domain.

Technically, “Default Domain Controllers Policy” is the *display name* of the GPO, while the *name* of the GPO is a GPO curly braced “GUID”. I put “GUID” in quotation marks because this identifier is not actually globally unique. The “Default Domain Controllers Policy” in every Active Directory domain will have the same “name” (read: curly braced GUID): {6AC1786C-016F-11D2-945F-00C04fB984F9}. For this reason, GPOs have an additional parameter called *objectguid*, which actually is globally unique. The policy files for any given GPO reside in the domain SYSVOL at the policy’s *gpcfilesyspath* (ex: \\contoso.local\sysvol\contoso.local\Policies\{6AC1786C-016F-11D2-945F-00C04fB984F9}).



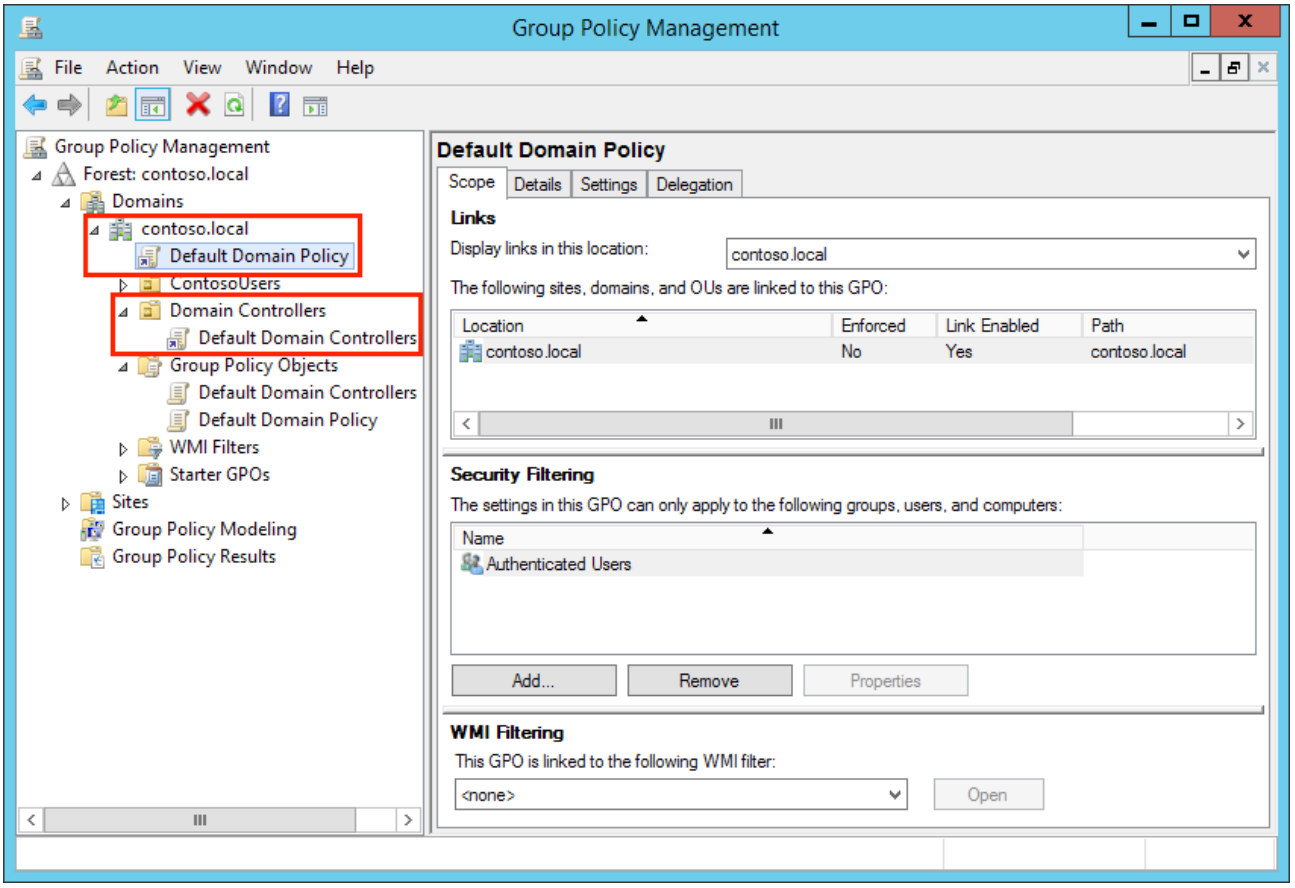
Above: The relevant properties of the "Default Domain Controllers Policy" GPO, and that GPO's policy files location in the SYSVOL.

OU: An Organizational Unit. According to Microsoft's TechNet, OUs are "general-purpose container[s] that can be used to group most other object classes together for administrative purposes". Basically, OUs are containers that you place principals (users, groups, and computers) into. Organizations will commonly use OUs to organize principals based on department and/or geographic location. Additionally, OUs can of course by nested within other OUs. This usually results in a relatively complex OU tree structure within a domain, which can be difficult to navigate without first being very familiar with the tree. You can see OUs in the ADUC (Active Directory Users and Computers) GUI. In the below screenshot, "ContosoUsers" is a child OU of the CONTOSO.LOCAL domain, "Helpdesk" is a child OU within the "ContosoUsers" OU, and "Alice Admin" is a child user of the "Helpdesk" OU:



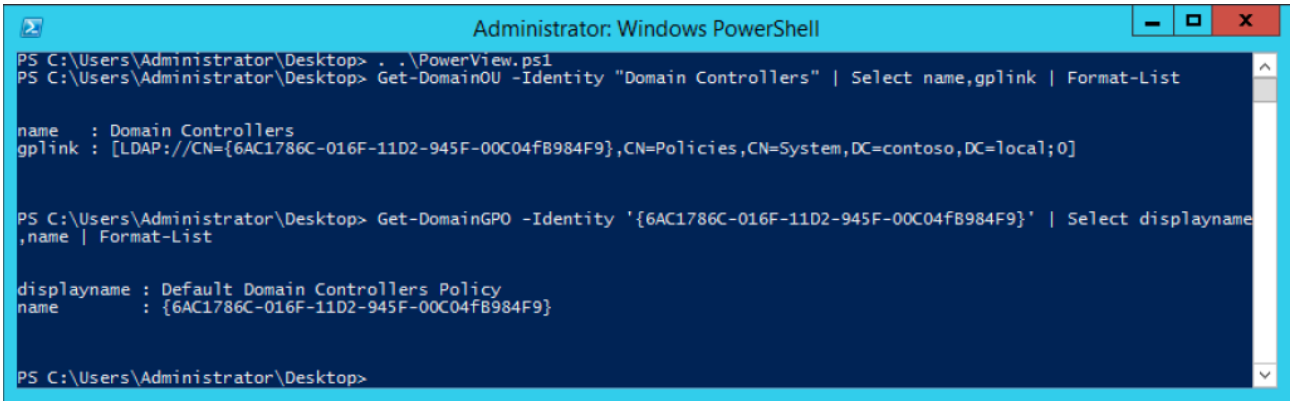
Above: The Alice Admin user within the OU tree.

GpLink: A Group Policy Link. GPOs can be “linked” to domains, sites, and OUs. By default, a GPO that is linked to an OU will apply to the child objects of that OU. For example, the “Default Domain Policy” GPO is linked, by default, to the domain object, while the “Default Domain Controllers Policy” is linked, by default, to the Domain Controllers OU. In the below screenshot, you can see that if we expand the “contoso.local” domain and the “Domain Controllers” OU, the GPOs linked to those objects appear below them:



Above: The "Default Domain Policy" is linked to the domain "contoso.local". The "Default Domain Controllers" policy is linked to the "Domain Controllers" OU.

GpLinks are stored on the objects the GPO is linked to, on the attribute called "gplink". The format of the "gplink" attribute value is [<Distinguished name of the GPO>;<0 if the link is not enforced, 1 if the link is enforced>]. You can easily enumerate those links with PowerView as in the example below:



Above: The "Default Domain Controllers Policy" GPO is linked to the "Domain Controllers" OU, and is not enforced.

Those three pieces — GPOs, OUs, and GpLinks — comprise the major moving parts we're working with. It's important to know those three pieces well before understanding GPO enforcement logic and how to use BloodHound to find attack paths, so make sure you feel confident with those before continuing on. One last note:

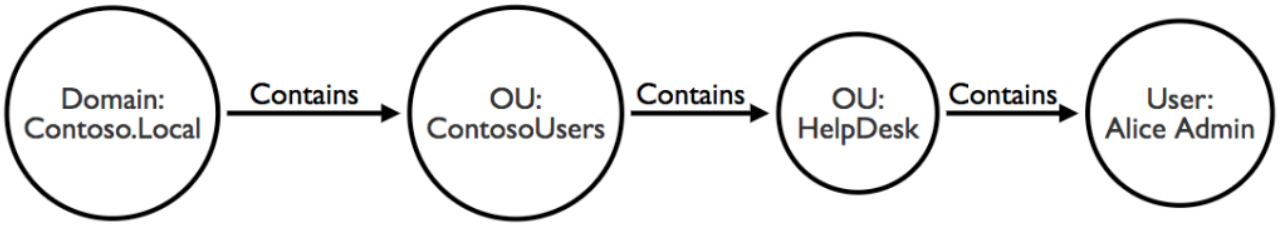
GPOs can also be linked to sites, but at this time we're not including that due to complications site memberships and collection challenges.

GPO Enforcement Logic

Now that you know the basic moving parts, let's look more closely at how they connect. GPO enforcement logic, very briefly, works like this:

- GpLinks can be *enforced*, or not.
- OUs can *block inheritance*, or not.
- If a GpLink is *enforced*, the associated GPO will apply to the linked OU and all child objects, regardless of whether any OU in that tree *blocks inheritance*.
- If a GpLink is **not** *enforced*, the associated GPO will apply to the linked OU and all child objects, **unless** any OU within that tree *blocks inheritance*.

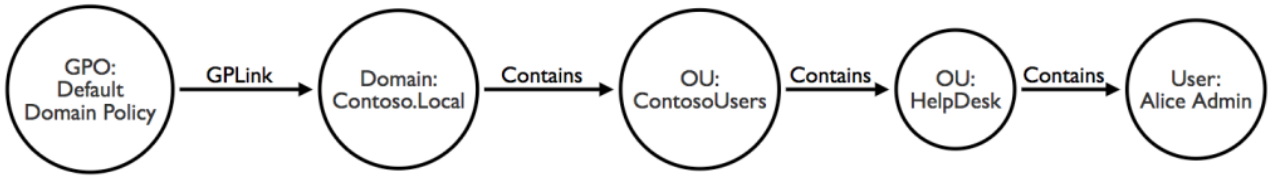
There are further complications on top of this, which we'll get to later on. First though, let's visualize the above rules regarding GpLink enforcement and OUs blocking inheritance. Recall earlier I had a user called Alice Admin within a HelpDesk OU. Instead of looking at that in ADUC, though, let's start to think about this as a graph:



Above: Alice Admin within the domain/OU tree.

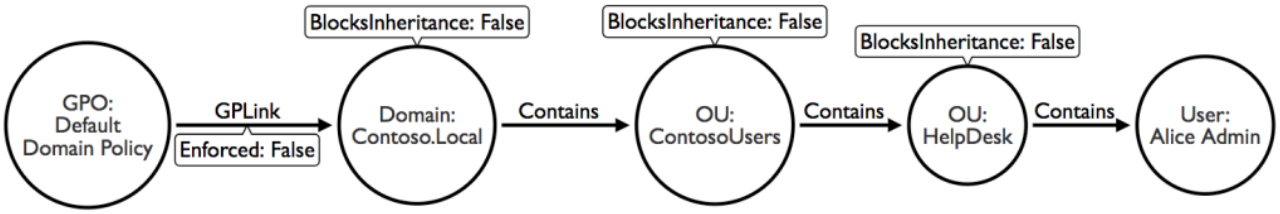
The domain object, Contoso.Local, is a container object. It contains the OU called ContosoUsers. The OU ContosoUsers contains the OU HelpDesk. Finally, the OU HelpDesk contains the user Alice Admin.

Now, let's add our Default Domain Policy GPO into the mix. Recall from earlier that in my test domain, that GPO is linked to the domain object:

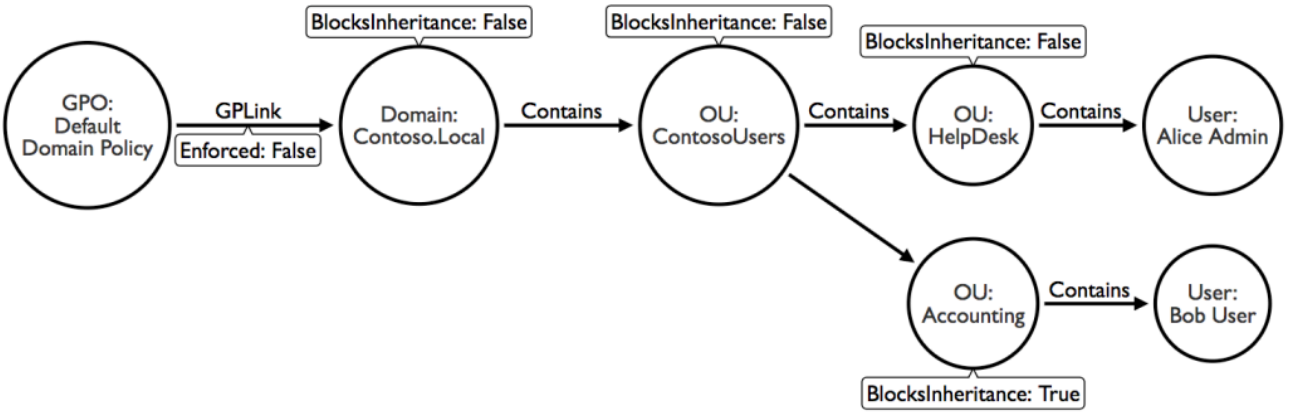


Above: The "Default Domain Policy" GPO is linked to the domain object.

Now, in default circumstances, you can simply read from left to right to figure out that the Default Domain Policy will apply to the user Alice Admin. The "default circumstance" here is that the GpLink relationship is not *enforced*, and that none of the containers in this path *block inheritance*. Let's add that information to the above graph:



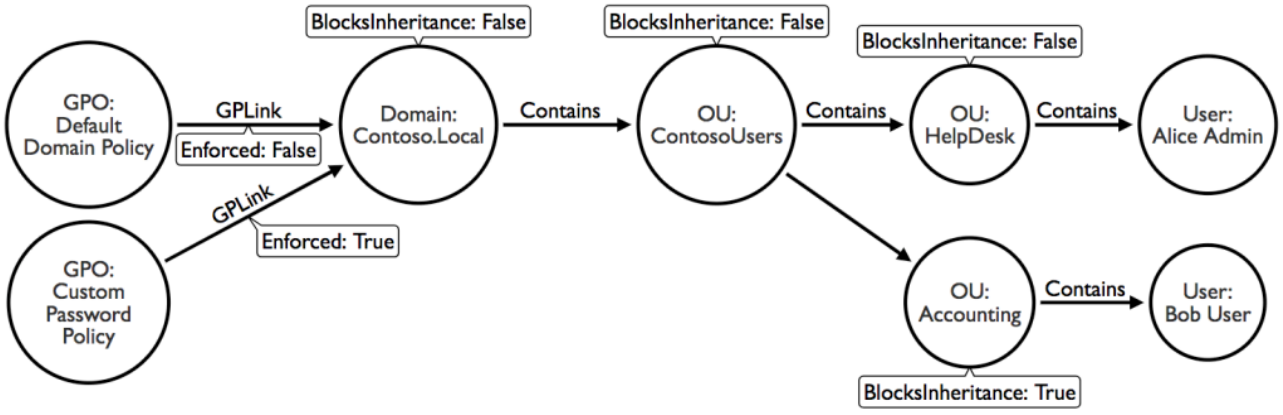
In this circumstance, it doesn't matter that the GpLink edge is *not enforced*, as none of the OUs *block inheritance*. In our test domain, we have another OU under ContosoUsers called "Accounting", with one user in that OU: Bob User. For example's sake, we'll say that the Accounting OU **does** *block inheritance.* Let's add that to our existing graph:



Again, we can see that the Default Domain Policy GPO is linked to the domain object, and Bob User is contained within the OU tree under the domain object; however, because the OU "Accounting" *blocks inheritance*, **and** because the GpLink edge is **not** enforced, the Default Domain Policy **will not** apply to Bob User.

Still with me? You'd be forgiven for being slightly confused at this point, but don't worry, it gets worse!

Let's add another GPO to the mix and link it to the domain object as well, except this time we will enforce the GpLink:



Our new GPO called "Custom Password Policy" is linked to the domain object, which again contains the entire OU tree under it. Now, because the GPLink is *enforced*, this policy will apply to all child objects in the OU tree,

regardless of whether any of those OUs *block inheritance*. This means that the "Custom Password Policy" GPO **will** apply to both "Alice Admin" **and** "Bob User", despite the "Accounting" OU *blocking inheritance*.

In our experience, this information is going to cover 95%+ of situations you'll run into in real enterprise networks; however, there are three more things to know about, which **may** impact you when abusing GPO control paths during your pentests and red team assessments: WMI filtering, security filtering, and Group Policy link order and precedence.

- WMI filtering allows administrators to further limit which computers and users a GPO will apply to, based on whether a certain WMI query returns True or False. For example, when a computer is processing group policy, it may run a WMI query that checks if the operating system is Windows 7, and only apply the group policy if that query returns true. See Darren Mar-Elia's excellent blog post for further details.
- Security filtering allows administrators to further limit which principals a GPO will apply to. Administrators can limit the GPO to apply to specific computers, users, or the members of a specific security group. By default, every GPO applies to the "Authenticated Users" principal, which includes any principal that successfully authenticates to the domain. For more details, see this post on the TechGenix site.
- Group Policy link order dictates which Group Policy "wins" in the event of conflicting, non-merging policies. Imagine you have two "Password Policy" GPOs: one that requires users to change their password every 30 days, and one that requires users to change their password every 60 days. Whichever policy is higher in the precedence order is the policy that will "win". The group policy client enforces this "win" condition by processing policies in **reverse order of precedence**, so the **highest** precedence policy is processed **last,** and "wins". Luckily, you don't need to worry about this for almost every abuse primitive. For more information, check out this blog post.

Like I said above, our experience has been that in real enterprise networks, you won't need to worry about WMI filtering, security filtering, or GpLink order in 95% or more of the situations you run into, but I mention them so you know where to start troubleshooting if your abuse actions aren't working. We may try to roll those three items into the BloodHound interface in the future. In the meantime, make sure your target computer and user objects won't be filtered out by WMI or security filters, or attempt to push an evil group policy that will be overruled by a higher precedence policy.

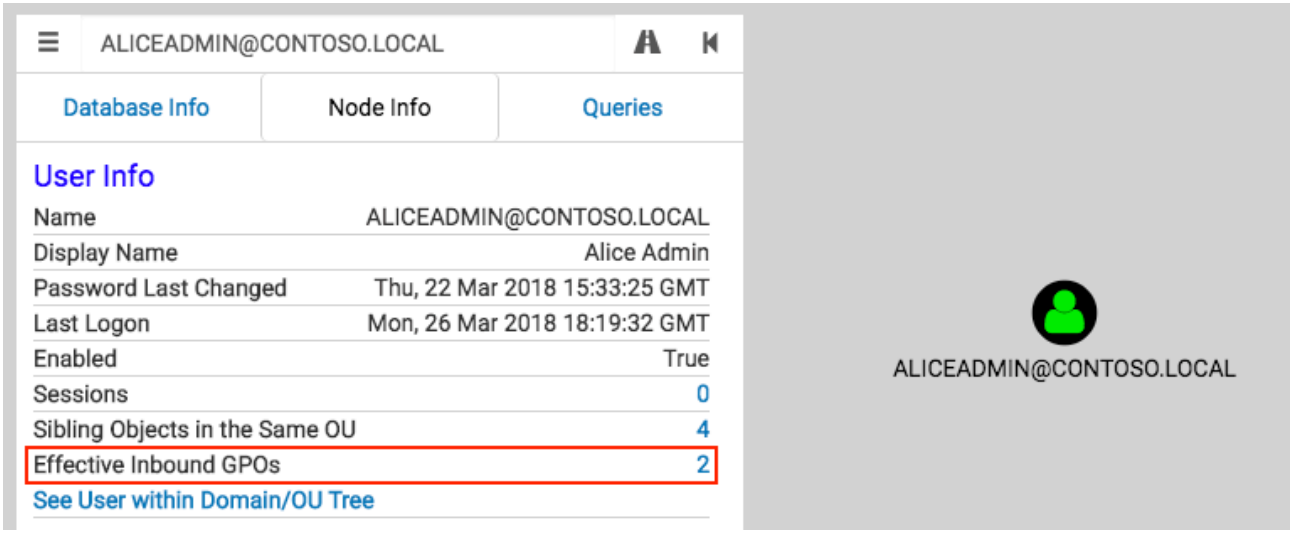
Analysis with BloodHound

First, make sure you are running at least BloodHound 1.5.1. Second, do your standard SharpHound collection like you always have, but this time either do the "All" or "Containers" and "ACL" collection methods, which will collect GPO ACLs and OU structure for you:

```
C:\> SharpHound.exe -c All
```

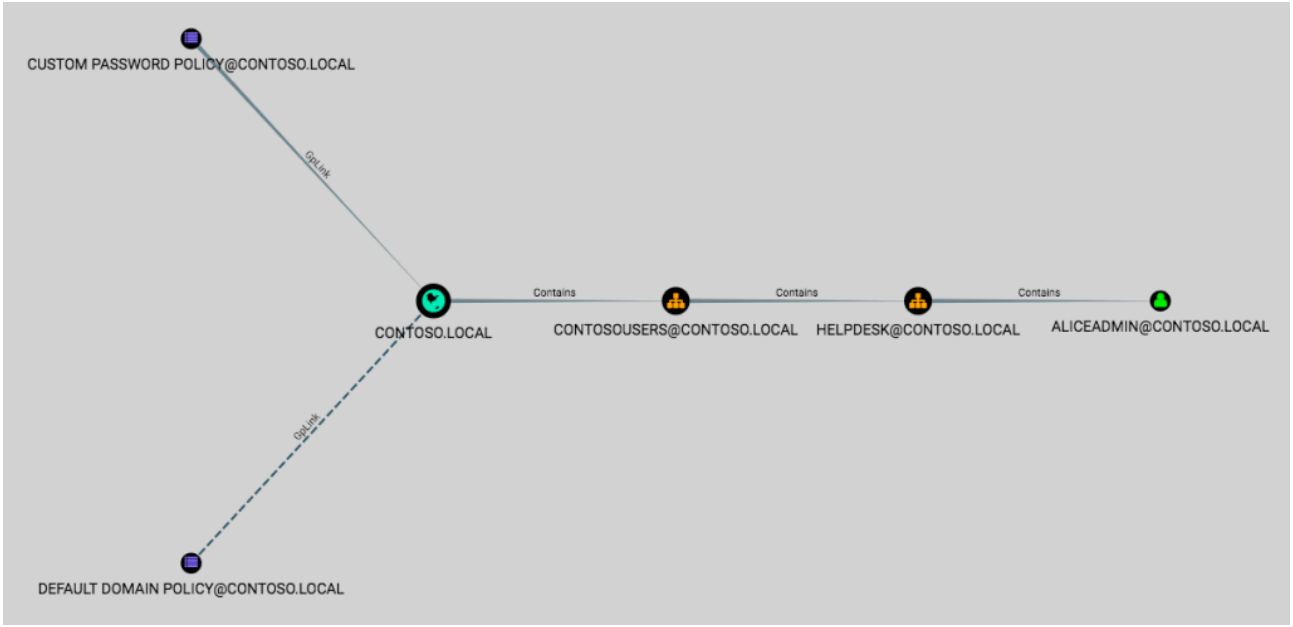
Then, import the resulting acls.csv, container_gplinks.csv, and container_structure.csv through the BloodHound interface like normal. Now you're ready to start analyzing outbound and inbound GPO control against objects.

For example, let's take a look at our "Alice Admin" user. If we search for this user, then click on the user node, you'll see some new information in the user tab, including "Effective Inbound GPOs":



Above: Two GPOs apply to Alice Admin.

The Cypher query that generates this number does the GpLink enforcement and OU blocking inheritance logic **for you**, so you don't need to worry about working that out yourself. Simply click on the number "2", in this instance, to visualize the GPOs that apply to "Alice Admin":

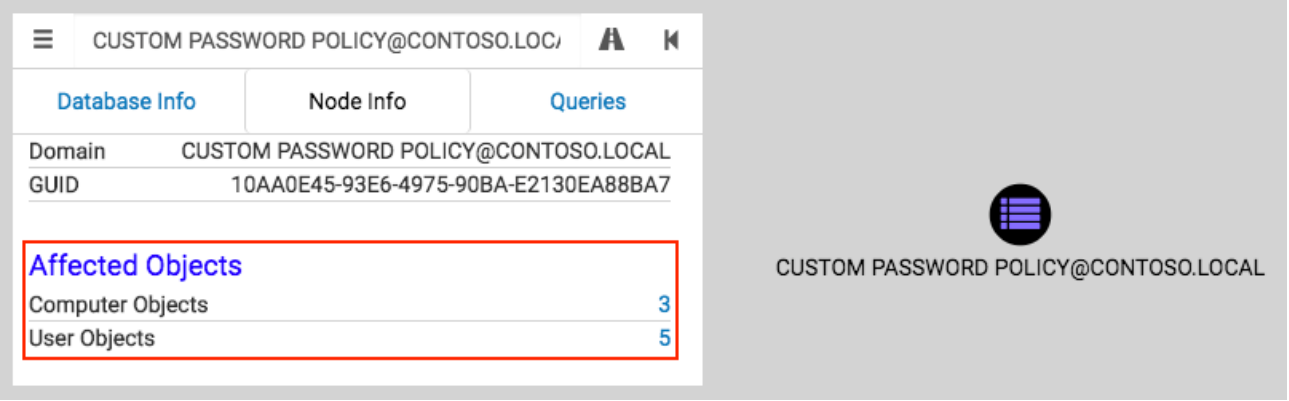


Above: How the two GPOs apply to Alice Admin.

Notice the edge connecting "Default Domain Policy" to the "Contoso.Local" domain is *dotted*. This means that this GPO is **not enforced**; however, all of the "Contains" edges are *solid*, meaning that none of those containers block inheritance. Recall from earlier that unenforced GpLinks will only be affected by OUs that block inheritance, so in this case, the Default Domain Policy **still applies** to Alice Admin.

Also note that the edge connecting “Customer Password Policy” to the “Contoso.Local” domain is *solid.* This means that this GPO **is enforced**, and will therefore apply to all children objects regardless of whether any subsequent containers block inheritance.

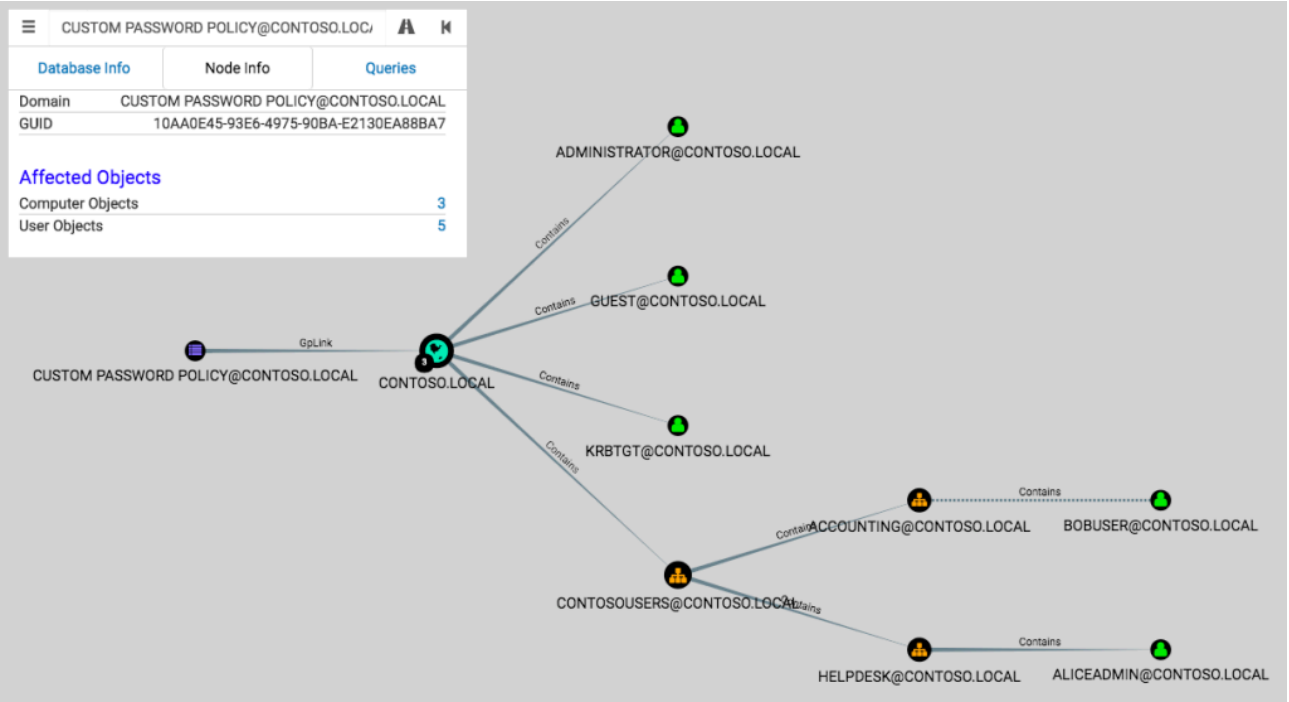
We can also see the flip side of this — what objects does any given GPO effectively apply to? First, let’s check out the Custom Password Policy GPO:



Above: The Custom Password Policy GPO applies to 3 computers and 5 users.

Reminder: GPOs can only apply to users and computers, not security groups.

By clicking on the numbers, you can render the objects affected by this GPO, and how the GPO applies to those objects. If we click the “5” next to “User Objects”, we get this graph:

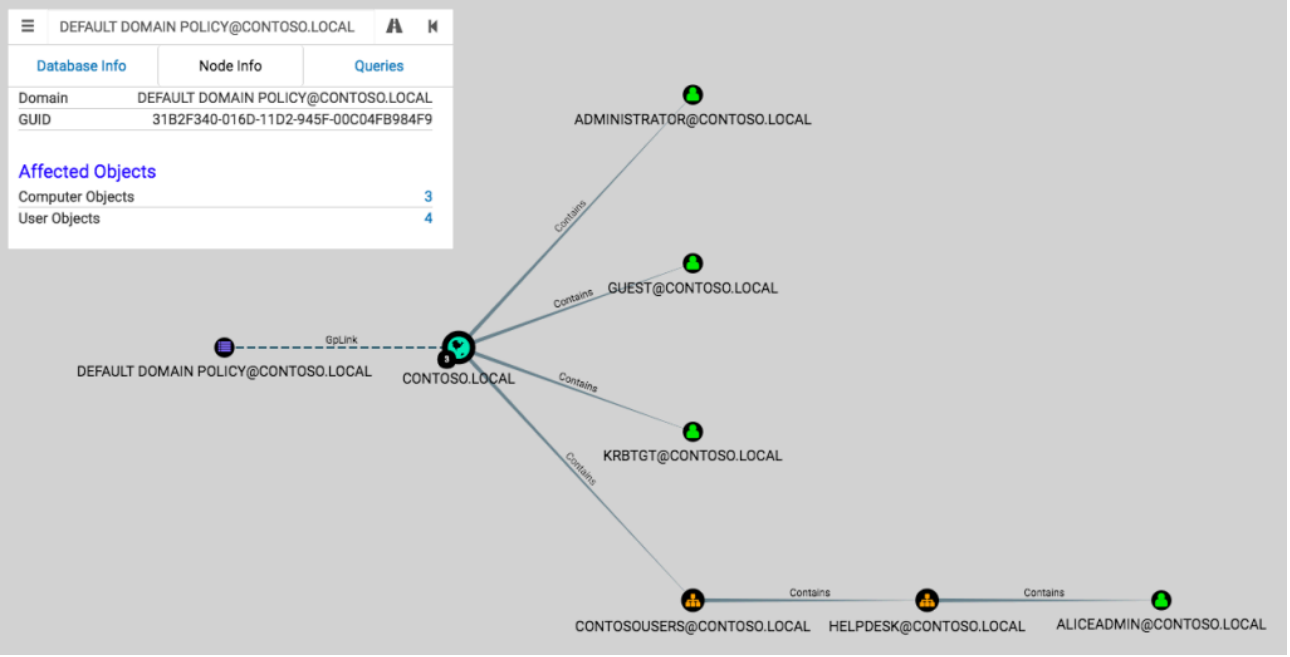


Above: How the Customer Password Policy GPO applies to user objects.

There are two important things to point out here: again, the edge connecting the “Custom Password Policy” GPO to the “Contoso.Local” domain object is solid, meaning this GPO **is enforced.** Second, notice the edge connecting

the “Accounting” OU to the “Bob User” user is *dotted*, indicating the “Accounting” OU **blocks inheritance.** But, because the “Custom Password Policy” GPO is **enforced**, the OU blocking inheritance **doesn’t matter**, and will be applied to the “Bob User” user anyway.

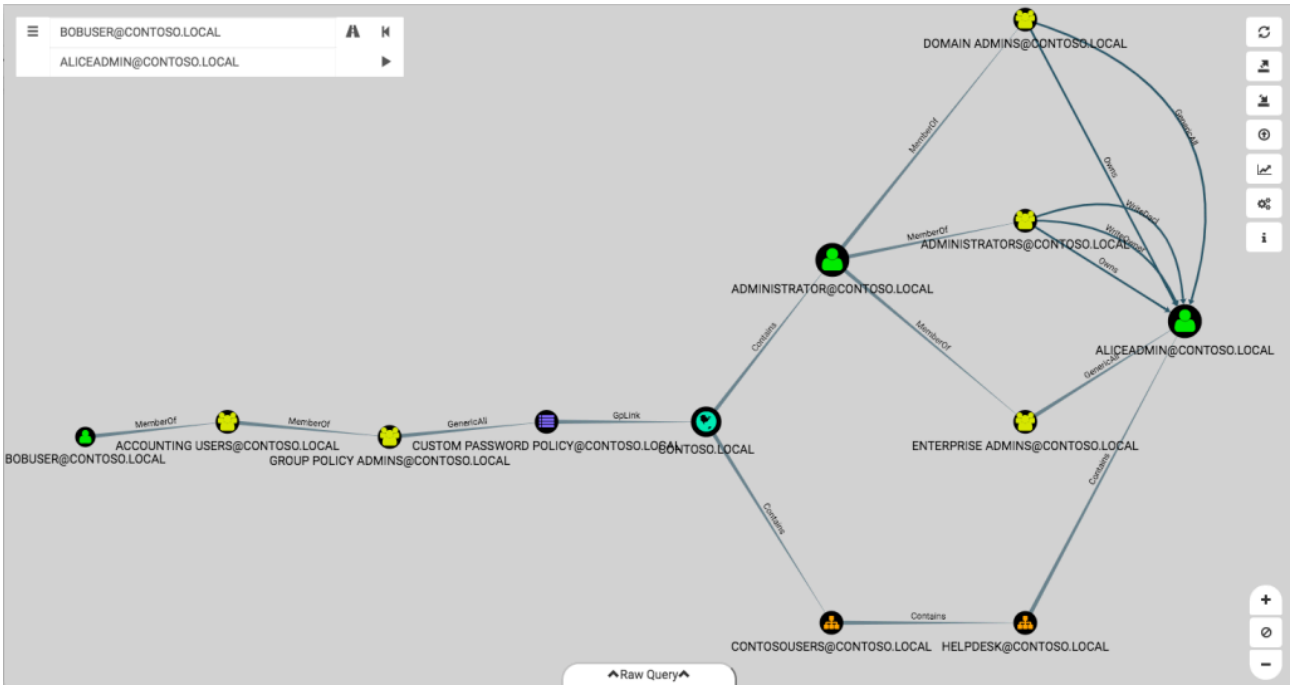
Compare the above graph to the graph we get if we do the same for the “Default Domain Policy”:



Above: The users affected by the “Default Domain Policy” GPO.

Notice how the “Bob User” user is no longer there? That’s because the “Default Domain Policy” GPO is **not enforced**. Because the “Accounting” OU **blocks inheritance**, that GPO will **not** apply to the “Bob User” user.

Alright, let’s put it all together and see if we can find an attack path from “Bob User” to “Alice Admin”. In the BloodHound search bar, click the path finding icon, then select your source node and target node. Hit enter, and BloodHound will find and render an attack path, if one exists:



Above: The attack path from "Bob User" to "Alice Admin".

Reading this graph from left to right, we can see that "Bob User" is in a group called "Accounting", which is part of a group called "Group Policy Admins" (believe me when I say crazier things have happened in the wild, and remember this is a contrived example :). The "Group Policy Admins" group has, as you would imagine, full control of the "Custom Password Policy" GPO. That GPO is then linked to the "Contoso.Local" domain. From here we have a couple options – push an evil policy down to the "Administrator" user and take over "Alice Admin" with an ACL based attack or just push an evil policy down directly to the "Alice Admin" user.

Abusing GPO Control

Finally, the most important part of this entire topic: how to actually take over computers and users with control over the GPOs that affect those users. For a bit of background and inspiration, read Will's excellent blog post on abusing GPO rights, which contains information about the first proof-of-concept GPO abuse cmdlet that I'm aware of, New-GPOImmediateTask.

When people say "you can do anything with GPO", they really mean it: you can do **anything** with GPO. Will and I put together this list of abuses against computers, including the policy location and abuse, just to give you a few ideas:

- Policy Location: Computer Configuration\Preferences\Control Panel Settings\Folder Options
- Abuse: Create/alter file type associations, register DDE actions with those associations.

- Policy Location: Computer Configuration\Preferences\Control Panel Settings\Local Users and Groups
- Abuse: Add new local admin account.

- Policy Location: Computer Configuration\Preferences\Control Panel Settings\Scheduled Tasks
- Abuse: Deploy a new evil scheduled task (ie: PowerShell download cradle).

- Policy Location: Computer Configuration\Preferences\Control Panel Settings\Services
- Abuse: Create and configure new evil services.
- Policy Location: Computer Configuration\Preferences\Windows Settings\Files
- Abuse: Affected computers will download a file from the domain controller.

- Policy Location: Computer Configuration\Preferences\Windows Settings\INI Files
- Abuse: Update existing INI files.

- Policy Location: Computer Configuration\Preferences\Windows Settings\Registry
- Abuse: Update specific registry keys. Very useful for disabling security mechanisms, or triggering code execution in any number of ways.

- Policy Location: Computer Configuration\Preferences\Windows Settings\Shortcuts
- Abuse: Deploy a new evil shortcut.

- Policy Location: Computer Configuration\Policies\Software Settings\Software installation
- Abuse: Deploy an evil MSI. The MSI must be available to the GP client via a network share.

- Policy Location: Computer Configuration\Policies\Windows Settings\Scripts (startup/shutdown)
- Abuse: Configure and deploy evil startup scripts. Can run scripts out of GPO directory, can also run PowerShell commands with arguments

- Policy Location: Computer Configuration\Policies\Windows Settings\Security Settings\Local Policies\Audit Policy
- Abuse: Modify local audit settings. Useful for evading detection.

- Policy Location: Computer Configuration\Policies\Windows Settings\Security Settings\Local Policies\User Rights Assignment\
- Abuse: Grant a user the right to logon via RDP, grant a user SeDebugPrivilege, grant a user the right to load device drivers, grant a user seTakeOwnershipPrivilege. Basically, take over the remote computer without ever being an administrator on it.

- Policy Location: Computer Configuration\Policies\Windows Settings\Security Settings\Registry
- Abuse: Alter DACLs on registry keys, grant yourself an extremely hard to find backdoor on the system.

- Policy Location: Computer Configuration\Policies\Windows Settings\Security Settings\Windows Firewall
- Abuse: Manage the Windows firewall. Open up ports if they're blocked.

- Policy Location: Computer Configuration\Preferences\Windows Settings\Environment
- Abuse: Add UNC path for DLL side loading.

- Policy Location: Computer Configuration\Preferences\Windows Settings\Files
- Abuse: Copy a file from a remote UNC path.

So, that's all well and good, but how do we actually take these actions? Currently, you've got two options: download and install the Group Policy Management Console and use the GPMC GUI to modify the relevant GPO or manually craft the relevant policy file and correctly modify the GPO and gpt.ini file.

As an example, let's say you want to push a new immediate scheduled task to a computer or user. My current understanding (which is definitely subject to correction), based on testing and the Microsoft Group Policy Preferences functional spec, follows:

Whenever a group policy client (user or computer) checks for updated group policy, they will go through several steps to collect and apply Group Policy to themselves. The client will check whether the remote version of the GPO is greater than the locally cached version of that GPO (unless gpupdate /force is used). The remote version of the GPO is stored in two locations:

1. As an integer value for the versionNumber attribute on the Group Policy Object itself.
2. As the same integer in the GPT.INI file, located at \\<domain.com>\Policies\<gpo name>\GPT.ini. Note that the "name" of the GPO is **not** the display name. For instance, the "name" for the Default Domain Policy is {6AC1786C-016F-11D2-945F-00C04fB984F9}.

If the remote GPO version number is greater than the locally cached version, the group policy client will continue, analyzing which policies and/or preferences it needs to search for in the relevant SYSVOL directory. For Group Policy preferences (which scheduled tasks fall under), the group policy client will check to see which Client-Side Extensions (CSEs) exist as part of the "gPCMachineExtensionNames" and "gPCUserExtensionNames" attributes. According to the Microsoft Group Policy Preferences functional spec, CSE GUIDs "enable a specific client-side extension on the Group Policy client to be associated with policy data that is stored in the logical and physical components of a Group Policy Object (GPO) on the Group Policy server, for that particular extension." The CSE GUIDs for Immediate Scheduled tasks, as they would be stored in the "gPCMachineExtensionNames" attribute, are:

```
[{00000000-0000-0000-0000-000000000000}{79F92669-4224-476C-9C5C-6EFB4D87DF4A}{CAB54552-DEEA-4691-817
```

And in a slightly more readable format:

```
[
    {00000000-0000-0000-0000-000000000000}
    {79F92669-4224-476C-9C5C-6EFB4D87DF4A}
    {CAB54552-DEEA-4691-817E-ED4A4D1AFC72}
]
[
    {AADCED64-746C-4633-A97C-D61349046527}
    {CAB54552-DEEA-4691-817E-ED4A4D1AFC72}
]
```

This translates to the following:

```
[
    {Core GPO Engine}
    {Preference Tool CSE GUID Local users and groups}
    {Preference Tool CSE GUID Scheduled Tasks}
```

```
]
[
    {Preference CSE GUID Scheduled Tasks}
    {Preference Tool CSE GUID Scheduled Tasks}
]
```

Once the group policy client understands that there are some scheduled tasks that apply to it, it will search for a file in the GP directory called ScheduledTasks.xml. That file exists in a predictable location:

```
\\<domain.com>\sysvol\<domain.com>\Policies\<gpo-name>\Machine\Preferences\ScheduledTasks.xml
```

Finally, the group policy client will parse the ScheduledTasks.xml and register the task locally.

That's how the process works, as I understand it. There is still a lot of work to be done on crafting scripts to automate the GPO abuse process, as installing GPMC is rarely a great option while on a red team assessment. **If ever there were a call to arms, this is it**: we'll continue working on creating scripts that reliably automate GPO control abuse, but are equally as excited to see what people in the community can come up with as well.

Conclusion

As Rohan mentioned in his post, BloodHound 1.5 represents a pretty big milestone for the BloodHound project. By adding in GPOs and OU structure, we're greatly increasing the scope of Active Directory attack surface you can easily map out with BloodHound. In a future blog post, I'll focus more on the defensive side of things, showing how defenders can use BloodHound to analyze and reduce the attack surface in AD now that we're tracking GPOs and OU structure.

BloodHound is available free and open source on GitHub at https://github.com/BloodHoundAD/BloodHound
You can join us on Slack at the official BloodHound Gang Slack here: https://bloodhoundgang.herokuapp.com/

Also published on Medium.
