A Day in the Life of a Windows Sysadmin

This Challenge assignment builds on the Group Policy Objectives activities from the previous class. You will create domain-hardening GPOs and revisit some PowerShell fundamentals.

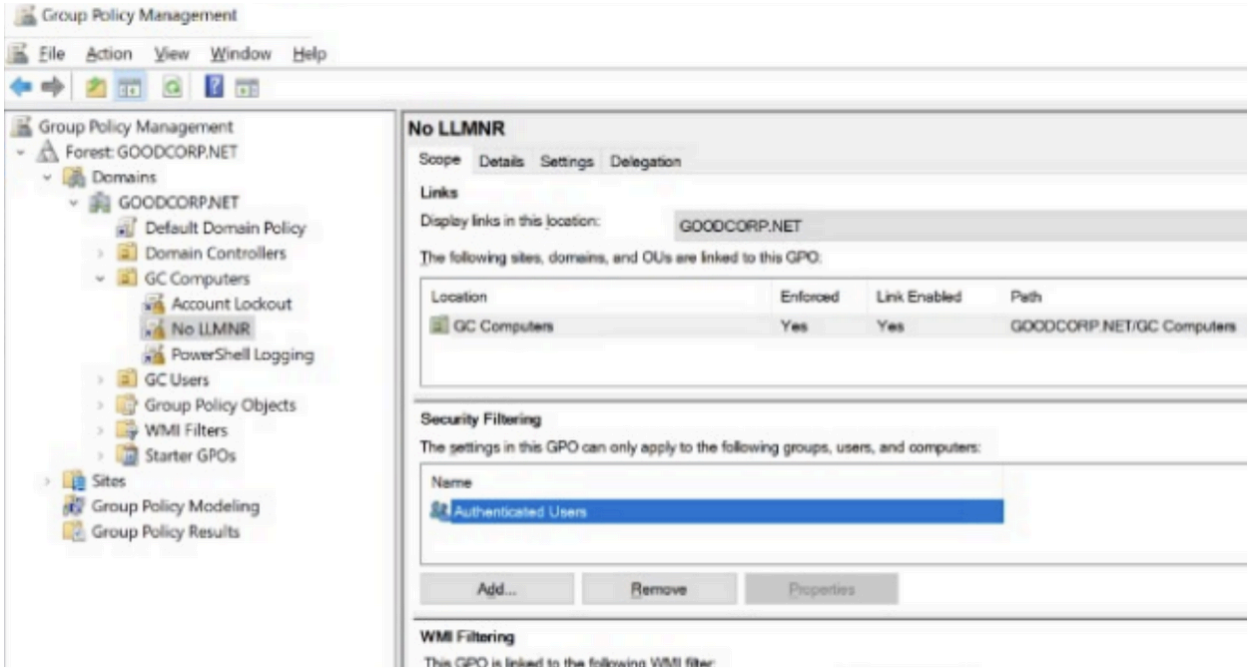
Task 1: Create a GPO—Disable Local Link Multicast Name Resolution (LLMNR)

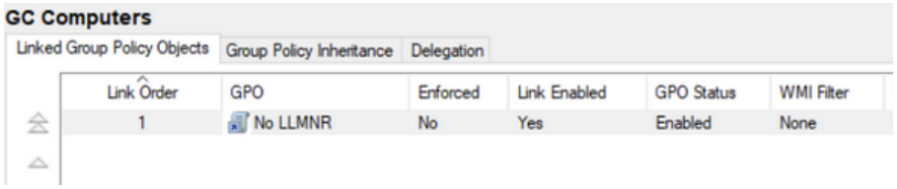
Local Link Multicast Name Resolution (LLMNR) is a vulnerability, so you will disable it on your Windows 10 machine (via the `GC Computers` OU).

A few notes about LLMNR:

- LLMNR is a protocol used as a backup (not an alternative) for DNS in Windows.
- When Windows cannot find a local address (e.g., the location of a file server), it uses LLMNR to send out a broadcast across the network asking if any device knows the address.
- LLMNR's vulnerability is that it accepts any response as authentic, allowing attackers to poison or spoof LLMNR responses, forcing devices to authenticate to them.
- An LLMNR-enabled Windows machine may automatically trust responses from anyone in the network.

- **Deliverable for Task 1:** A screenshot of all the GPOs created for this assignment. To find these, launch the Group Policy Management tool, select Group Policy Objects, and take a screenshot of the GPOs you've created. Name the screenshot file `GPOs`.





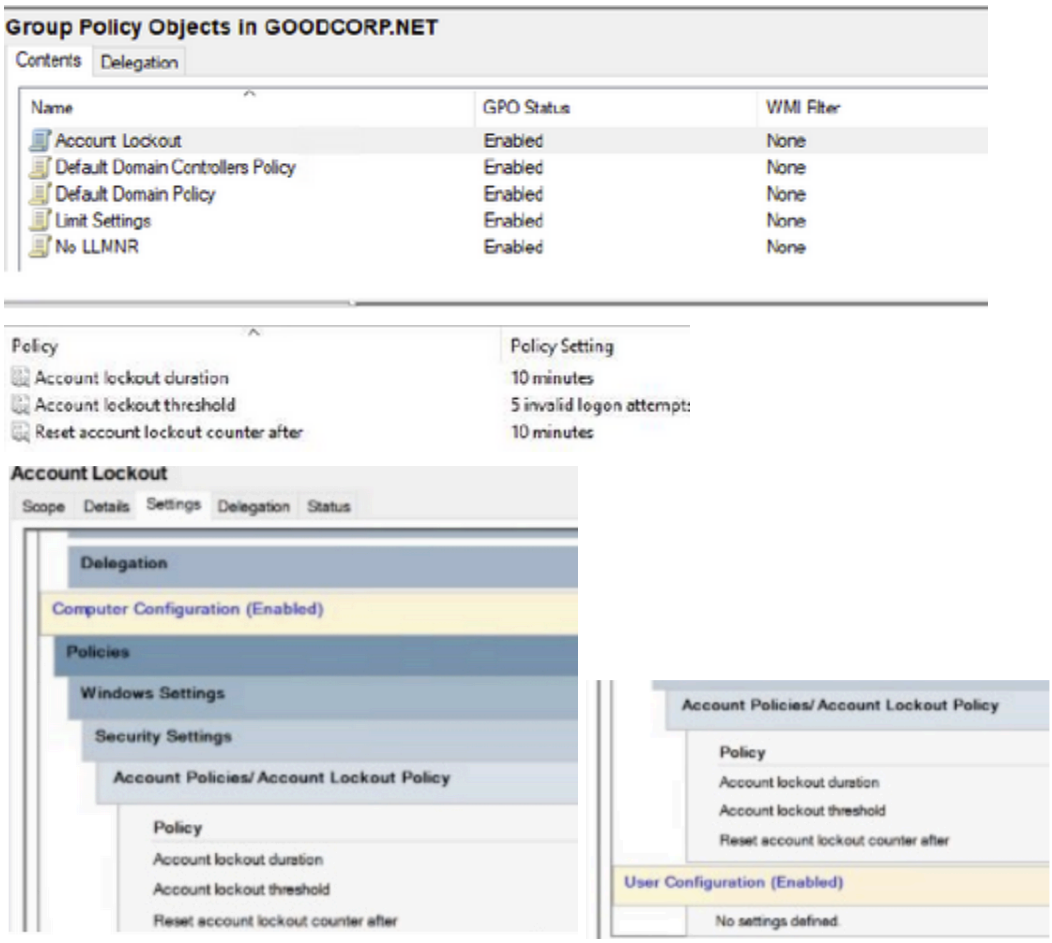
Task 2: Create a GPO—Account Lockout

For security and compliance reasons, the CIO needs you to implement an account lockout policy on your Windows workstation. An account lockout disables access to an account for a set period of time after a specific number of failed login attempts. This policy defends against brute-force attacks, in which attackers can enter a million passwords in just a few minutes.

Account lockouts have some important considerations. Read about these in the following documentation: Microsoft Security Guidance: Configuring Account Lockout

Links to an external site.. You only need to read the "Account Lockout Tradeoffs" and "Baseline Selection" sections.

- **Deliverable for Task 2:** A screenshot of the different `Account Lockout` policies in Group Policy Management Editor. It should show the three values you set under the Policy and Policy Setting columns. Name the screenshot file `Account-Lockout-Policies`.

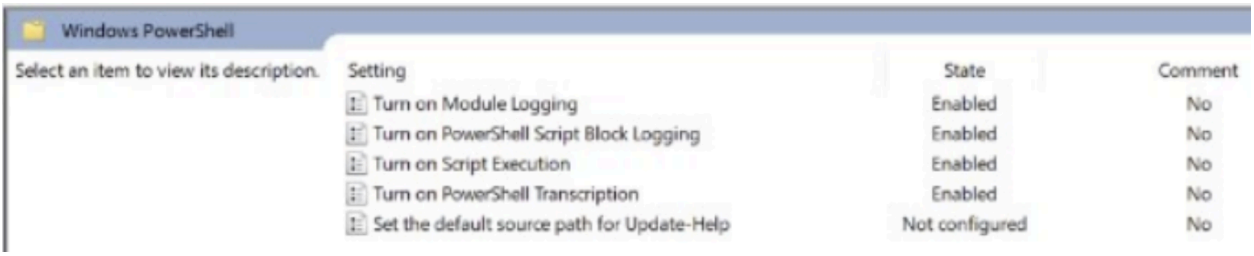


Task 3: Create a GPO—Enabling Verbose PowerShell Logging and Transcription

As mentioned in a previous lesson, PowerShell is often used as a living off the land Links to an external site. hacker tool. This means:

    - Once a hacker gains access to a Windows machine, they will leverage built-in tools, such as PowerShell and `wmic`, as much as possible to achieve their goals while trying to stay under the radar.
- So why not just completely disable PowerShell?
    - Many security tools and system administration management operations, such as workstation provisioning, require heavy use of PowerShell to set up machines.
    - Best practices for enabling or disabling PowerShell are debated. This often leads to the solution of allowing only certain applications to run. These setups require a heavy amount of configuration using tools such as AppLocker
- Links to an external site..
- For this reason, we'll use a PowerShell practice that is recommended regardless of whether PowerShell is enabled or disabled: enabling enhanced PowerShell logging and visibility through verbosity.
- This type of policy is important for tools like SIEM and for forensics operations, as it helps combat obfuscated PowerShell payloads.

- **Deliverable for Task 3:** A screenshot of the different `Windows PowerShell` policies within the Group Policy Management Editor. Four of these should be enabled. Name the screenshot file `Windows-PowerShell-Policies`.



Task 4: Create a Script—Enumerate Access Control Lists

Before we create a script, let's review Access Control Lists

Links to an external site..

- In Windows, access to files and directories are managed by Access Control Lists (ACLs). These identify which entities (known as security principals), such as users and groups, can access which resources. ACLs use security identifiers to manage which principals can access which resources.

- While you don't need to know the specific components within ACLs for this task, you do need to know how to use the `Get-Acl` PowerShell cmdlet to retrieve them. View Get-Acl documentation here
- Links to an external site..

Familiarize yourself with the basics of `Get-Acls`:

- `Get-Acl` without any parameters or arguments will return the security descriptors of the directory you're currently in.
- `Get-Acl <filename>` will return the specific file's ACL. We'll need to use this for our task.

- **Deliverable for Task 4:** A copy of your `enum_acls.ps1` script.

```
$directory = Get-ChildItem  .\

foreach ($item in $directory) {

        Get-Acl  $item

}
```

- **Deliverable for Bonus Task 5:** A screenshot of the contents of one of your transcribed PowerShell logs (named `PowerShell-logs`) or a copy of one of the logs.
    - Start time:
        - Username: GOODCORP\sysadmin
        - RunAs User: GOODCORP\sysadmin
-