Setting Up and Securing an Active Directory Lab

Authors :

ELLAILI Daoud
DAOUCHE Oualid

04 July 2025

Contents

1 Lab Environment Setup and Configuration **3**
- 1.1 Lab Preparation . . . 3
- 1.2 Network Switch and Router Configuration . . . 3
- 1.3 Active Directory Server (Windows Server) . . . 4
  - 1.3.1 Step 1: Install Windows Server . . . 4
  - 1.3.2 Step 2: Update and Basic Configurations . . . 4
  - 1.3.3 Step 3: Prepare for AD Role . . . 4
- 1.4 Splunk Server . . . 5
  - 1.4.1 Step 1: Install Ubuntu Server . . . 5
  - 1.4.2 Step 2: Splunk Initial Configuration . . . 5
- 1.5 Windows 10 Target Machine . . . 6
  - 1.5.1 Step 1: Install Windows 10 . . . 6
  - 1.5.2 Step 2: Basic Configuration . . . 6
- 1.6 Kali Linux Attacker Machine . . . 7
  - 1.6.1 Step 1: Install Kali . . . 7
  - 1.6.2 Step 2: Update and Configure Kali . . . 7

2 Active Directory Deployment **8**
- 2.1 Promoting the Server to Domain Controller . . . 8
- 2.2 Creating AD Users, Groups, and OUs (Automated with `Setup-ADLab.ps1`) . . . 9
- 2.3 Joining the Windows 10 Client to the Domain . . . 10
- 2.4 Implementing Group Policies . . . 10

3 Monitoring and Logging with Splunk **15**
- 3.1 Installing Splunk Universal Forwarder on Windows Hosts . . . 15
- 3.2 Monitoring Critical Events and Alerts in Splunk . . . 18

4 Attack Simulation with Kali Linux **20**
- 4.1 Reconnaissance with BloodHound (Active Directory Mapping) . . . 20
- 4.2 Credential Dumping with Mimikatz . . . 22

5 Security Hardening Best Practices **26**

6 Threat Detection and Response with Splunk **28**
- 6.1 Detecting Active Directory Attacks in Logs . . . 28
- 6.2 Incident Response in the Lab . . . 32
- 6.3 Using Splunk for Continuous Monitoring . . . 32

Introduction

This project walks through building a secure Active Directory (AD) lab environment with the following components: an AD **Domain Controller** (Windows Server), a **Windows 10 client** (domain-joined workstation), a **Splunk server** (for log aggregation), a **Kali Linux attacker** machine, and a **network switch/router** to interconnect them. We cover installation and configuration of each, deploying Active Directory (users, groups, Group Policies), setting up monitoring with Splunk, simulating attacks (Mimikatz, BloodHound, CrackMapExec), and implementing security hardening (LAPS, auditing, firewall, SIEM alerts).

Chapter 1

Lab Environment Setup and Configuration

1.1 Lab Preparation

Before configuring services, We need to prepare the lab environment. A virtualized environement is used with the following architecture

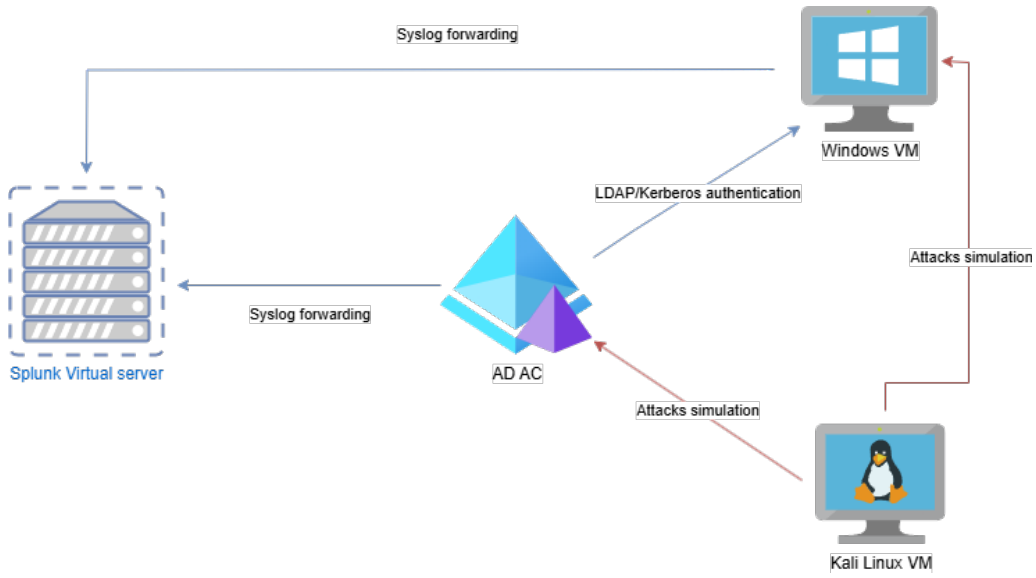

Figure 1.1: Lab architecture.

1.2 Network Switch and Router Configuration

- **Network Design**: Connecting all lab machines to a common network using an isolated subnet `192.168.1.0/24` in VMware.
- **IP Configuration**: Assigning the domain controller IP `192.168.1.2/24`, the Splunk server `192.168.1.10/24`, the Windows 10 client `192.168.1.5/24`, and the Kali machine `192.168.1.111/24`, and the default gateway as `192.168.1.1/24`.

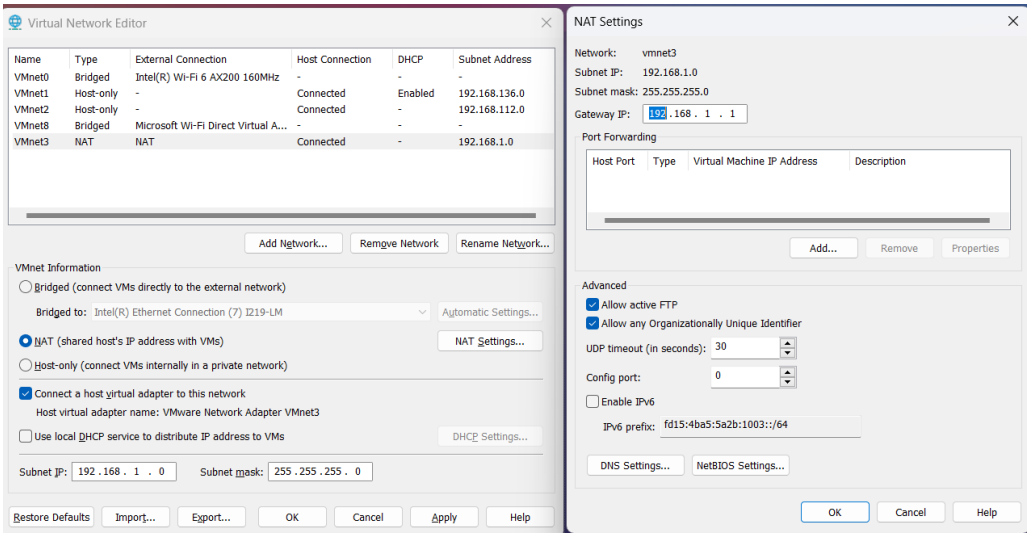

Figure 1.2: Router/NAT Configuration.

- **Router/NAT**: For internet access in the lab (for updates or downloads), We use a NAT interface or a virtual router.**We Ensure the router blocks inbound traffic from the internet** – the lab is isolated except for intentional updates. Only allow necessary outbound traffic (e.g. to download patches or tools).
- **Connectivity**: We Verify all machines can ping each other on the lab network. The switch (virtual or physical) allow free communication between the VMs.

1.3 Active Directory Server (Windows Server)

1.3.1 Step 1: Install Windows Server

We set up a VM for the domain controller, using a Windows Server 2019ISO. Allocate sufficient resources (e.g. 2–4 CPU, 4–8GB RAM, 40GB disk). After install, we **assign a static IP** on the lab network and configure DNS to point to itself.

1.3.2 Step 2: Update and Basic Configurations

After reboot, We log in as Administrator. Install the latest Windows updates and patches. Updating ensures security vulnerabilities are patched before proceeding. Also, We disable any extraneous services and set a strong local Administrator password. At this stage, **set the machine's time zone** and verify the clock is correct (time sync is important in AD for Kerberos).

1.3.3 Step 3: Prepare for AD Role

We install the **Active Directory Domain Services (AD DS)** role.

1.4 Splunk Server

The Splunk server will collect and index logs from the Windows machines. We used **Ubuntu Linux** for the Splunk server.

1.4.1 Step 1: Install Ubuntu Server

We Create a VM for Splunk (Ubuntu 20.04+ LTS). Allocate resources (2 CPU, 4GB RAM, 50GB disk). After that we precede to install splunk and configure it.

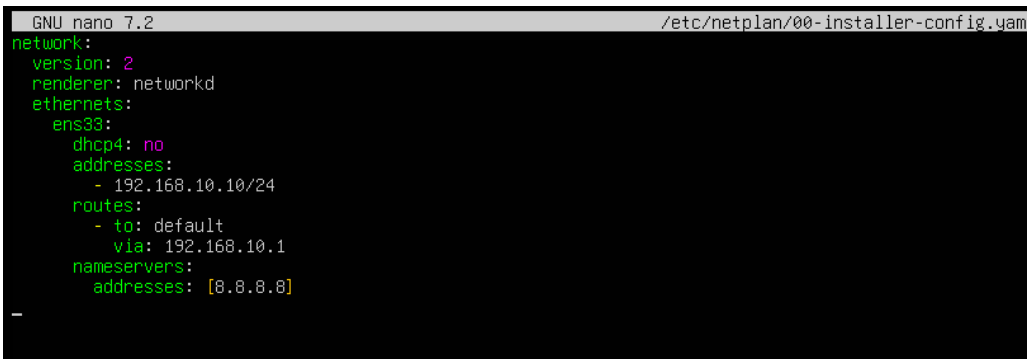

Figure 1.3: Ubuntu configuration.

After installation, we create a Splunk admin user and the following command

```
sudo /opt/splunk/bin/splunk start --accept-license
```

Now we create a username and password for the Splunk web UI. Then, we enable Splunk to start at boot:

```
sudo /opt/splunk/bin/splunk enable boot-start
```

Now Splunk is running on this server. By default, the **Splunk web interface is on port 8000** and the **indexing (receiver) port is 9997** (we will confirm receiver setup later).

1.4.2 Step 2: Splunk Initial Configuration

We access the Splunk Web interface. Go to **Settings** → **Server Settings** → **General Settings** and ensure the host name/IP is correct. Next, We enable Splunk to receive data from forwarders: go to **Settings** → **Forwarding and receiving** → **Configure Receiving** and **add a new TCP port 9997**. This opens port 9997 for incoming logs from Universal Forwarders. Also, We create an index for Windows logs that we name it "endpoint" with default settings. This index will store Windows Event logs from the clients.

Splunk server setup is now complete. We Leave the Splunk server running and ensure port 8000 (web) and 9997 (receiving) are open/accessible to the lab network.

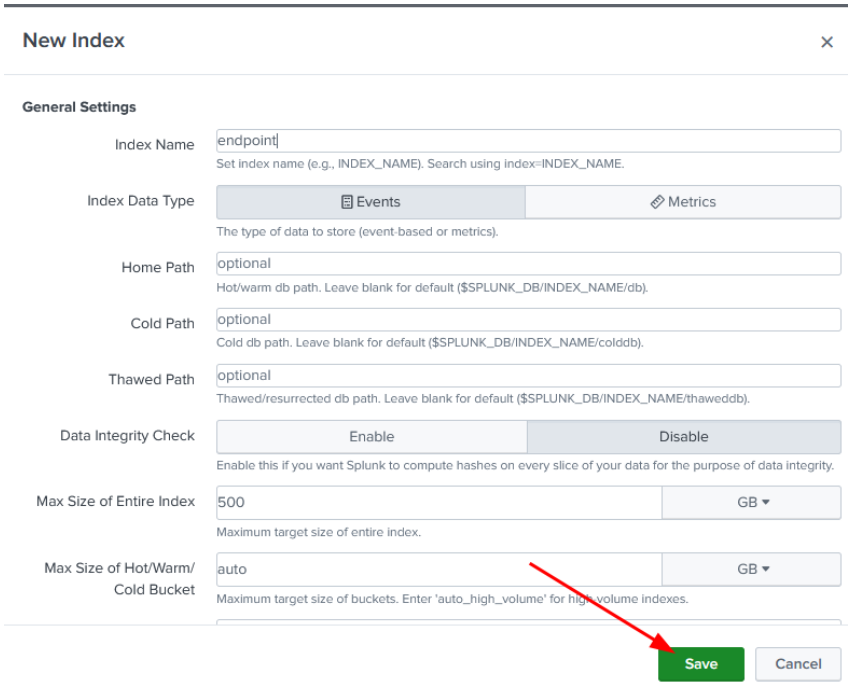

Figure 1.4: Splunk index configuration.

1.5 Windows 10 Target Machine

The Windows 10 machine will simulate a client workstation in the domain (a "user" machine). It will later be joined to AD and used for both normal activity and as a target for attacks.

1.5.1 Step 1: Install Windows 10

We Create a VM for the client. Allocate ∼2 CPU, 4GB RAM, and ∼40GB disk. We Use a Windows 10 ISO.We temporary Create a local user for setup, but we'll use domain accounts later.

1.5.2 Step 2: Basic Configuration

We **Rename the PC** to a meaningful name, e.g. "Victim-PC". Next, assign a **static IP** on the lab network. We use the following configuration: ip of the host `192.168.1.5/24`, gateway `192.168.1.1`, DNS `192.168.1.2`.

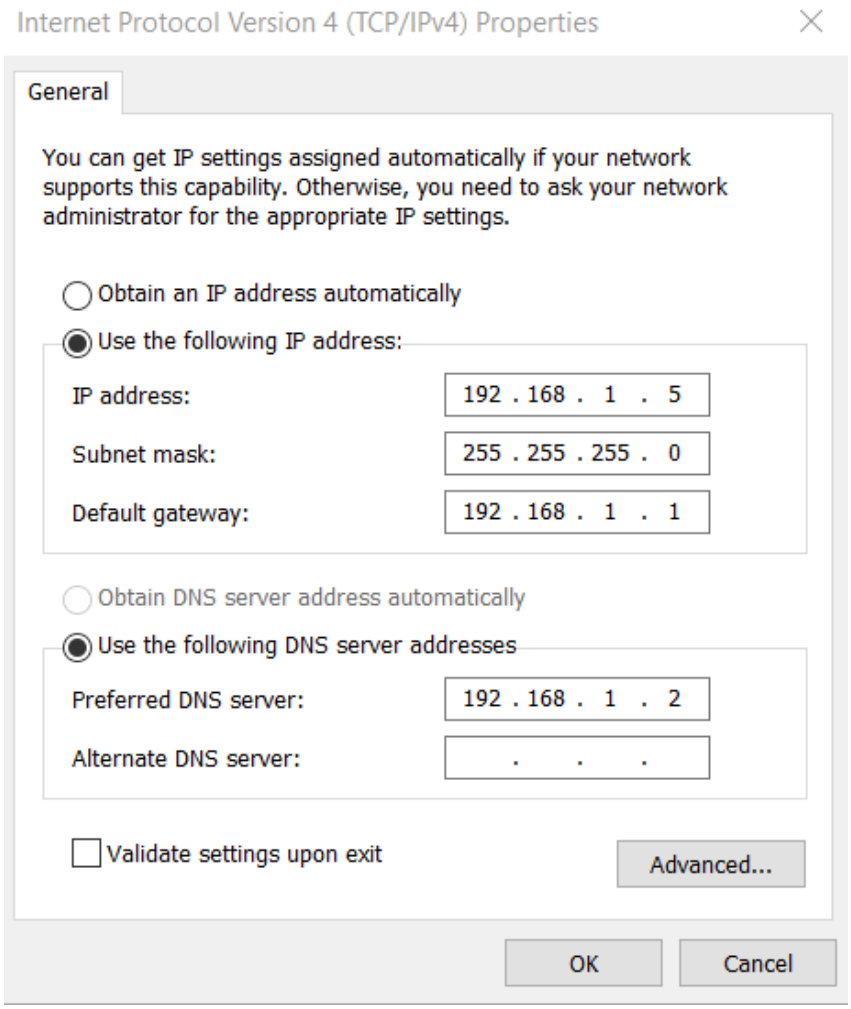

Figure 1.5: Victim-PC configuration.

1.6 Kali Linux Attacker Machine

The Kali machine will be used to simulate an attacker's perspective. It will *not* join the AD domain, but will reside in the same network to perform reconnaissance and attacks.

1.6.1 Step 1: Install Kali

We Create a VM and install Kali Linux. Allocate ∼2 CPU, 4GB RAM, 20GB+ disk. After installation, We ensure the Kali VM is on the same lab network attaching its NIC to the lab's switch/NAT network.

1.6.2 Step 2: Update and Configure Kali

We Run `sudo apt update && sudo apt upgrade` to ensure Kali is up-to-date with the latest tools. Most hacking tools we need (like nmap, BloodHound, CrackMapExec, Mimikatz, etc.) are included or available via package or GitHub. We will use Kali primarily for *offensive* tools, so no special configuration is needed in terms of roles.

Chapter 2

Active Directory Deployment

This section covers promoting the Windows Server to a Domain Controller, creating an AD domain, and provisioning users, groups, and Group Policies. We will create a new forest/domain (since this is a lab). Active Directory Domain Services (AD DS) will allow the Windows 10 machine to join a domain and enable centralized management.

2.1 Promoting the Server to Domain Controller

Step 1: Configure AD DS

On **DC01**,we open **Server Manager**, and launches the AD DS Configuration Wizard. We'll use `samssar.local` as the AD domain.

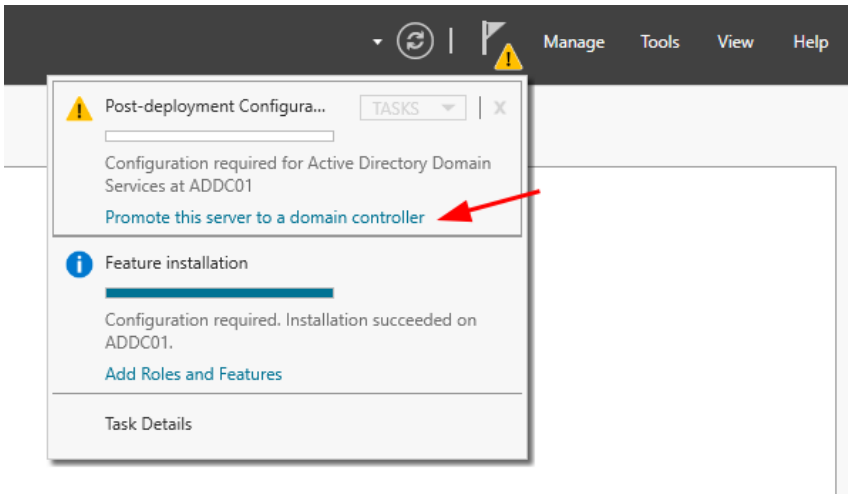

Figure 2.1: Promote the server to the domain controller.

Step 2: Set DSRM Password

In the wizard, We set the **Directory Services Restore Mode (DSRM) password** (this is used for AD restore in safe mode). We then Choose a strong password and note it down. Continue with the defaults: the wizard will automatically install DNS Server as part of AD.

After that We Proceed through the wizard. Ignoring any DNS delegation warning (since this is a new DNS zone). The NetBIOS domain name will auto-fill (e.g. "LAB"). We Accept defaults for database and SYSVOL locations. Finally the promotion is complete.

Step 3: Post-Promotion Tasks

After reboot, the server is now a **Domain Controller (DC)** for `samssar.local`. We Log in using the domain Administrator account (which is the same credentials as the old local Administrator, but now under the domain).

2.2 Creating AD Users, Groups, and OUs (Automated with `Setup-ADLab.ps1`)

Rather than using the graphical Active Directory Users and Computers (ADUC) console, every object in our brokerage domain `Samssar.local` is provisioned automatically by the PowerShell script `Setup-ADLab.ps1`. This approach ensures repeatability, lets us version–control the configuration, and avoids manual errors.

Step 1: Create Organizational Units (OUs)

The script issues a series of `New-ADOrganizationalUnit` cmdlets:

```
New-ADOrganizationalUnit -Name "Brokerage_Management" -Path "DC=Samssar,DC=local"
New-ADOrganizationalUnit -Name "Brokers"              -Path "DC=Samssar,DC=local"
New-ADOrganizationalUnit -Name "Clients"              -Path "DC=Samssar,DC=local"
New-ADOrganizationalUnit -Name "IT_Security"          -Path "DC=Samssar,DC=local"
New-ADOrganizationalUnit -Name "Finance"              -Path "DC=Samssar,DC=local"
```

This mirrors the company's structure (*Brokers*, *Clients*, *Finance*, etc.) and replaces the default Users container, allowing targeted Group Policy application and cleaner delegation.

Step 2: Create User Accounts

Domain users are generated with `New-ADUser`, each supplied a strong initial password stored in the variable `$AdminPassword`:

```
New-ADUser -Name "Hassan El Amrani" -SamAccountName "hamrani" -Path "OU=Brokers,DC=Sa
New-ADUser -Name "Siham Bouzid"    -SamAccountName "sbouzid" -Path "OU=Brokers,DC=Sam
New-ADUser -Name "Omar Jebli"      -SamAccountName "ojebli"  -Path "OU=Clients,DC=Sam
New-ADUser -Name "Salma Bennani"   -SamAccountName "sbennani"-Path "OU=Finance,DC=Sam
New-ADUser -Name "Adil Fassi"      -SamAccountName "afassi"  -Path "OU=IT_Security,DC
```

At least one *privileged* (`hamrani`, senior broker) and several *non-privileged* accounts are included to model everyday users versus administrative roles.

Step 3: Create Groups and Assign Membership

Security groups are defined with `New-ADGroup` and populated via `Add-ADGroupMember`:

```
New-ADGroup -Name "Senior_Brokers" -Path "OU=Brokers,DC=Samssar,DC=local" -GroupScope
Add-ADGroupMember -Identity "Senior_Brokers" -Members "hamrani"

New-ADGroup -Name "Junior_Brokers" ...
Add-ADGroupMember -Identity "Junior_Brokers" -Members "sbouzid"

New-ADGroup -Name "VIP_Clients" ...
Add-ADGroupMember -Identity "VIP_Clients" -Members "ojebli"
```

These groups later drive access control lists (ACLs) and tailored Group Policies.

Step 4: Domain Security Baseline

The same script also:

- Links a custom GPO (`Brokerage Security Policy`) enforcing UAC and hard password rules.
- Sets a domain password policy: complexity enabled, `MinPasswordLength = 10`, lockout after five failures for 15 minutes.

Because every action is scripted, rebuilding the lab—or cloning it for additional testing—requires only rerunning `Setup-ADLab.ps1` on a fresh Windows Server, with no GUI clicks at all.

2.3 Joining the Windows 10 Client to the Domain

Now that AD is ready with user accounts, We join the Windows 10 machine to the domain

Now we have an AD domain with at least one DC and one domain-joined client. The user accounts and groups are set up. Next, we configure **Group Policies** to enforce security settings in the domain.

2.4 Implementing Group Policies

Active Directory Group Policy allows us to enforce configuration and security settings across computers and users in the domain. We will set up some essential policies for our lab:

Step 1: Default Domain Policy – Password Policy

By default, AD has a password policy (minimum length 7, complexity required, etc.). We don't cahnge it but to change it we start by opening **Group Policy Management** (GPMC) from Server Manager → Tools. Expand Forest → Domains → `samssar.local`. Right-click **Default Domain Policy** and choose Edit. Navigate to **Computer Configuration** → **Policies** → **Windows Settings** → **Security Settings** → **Account**

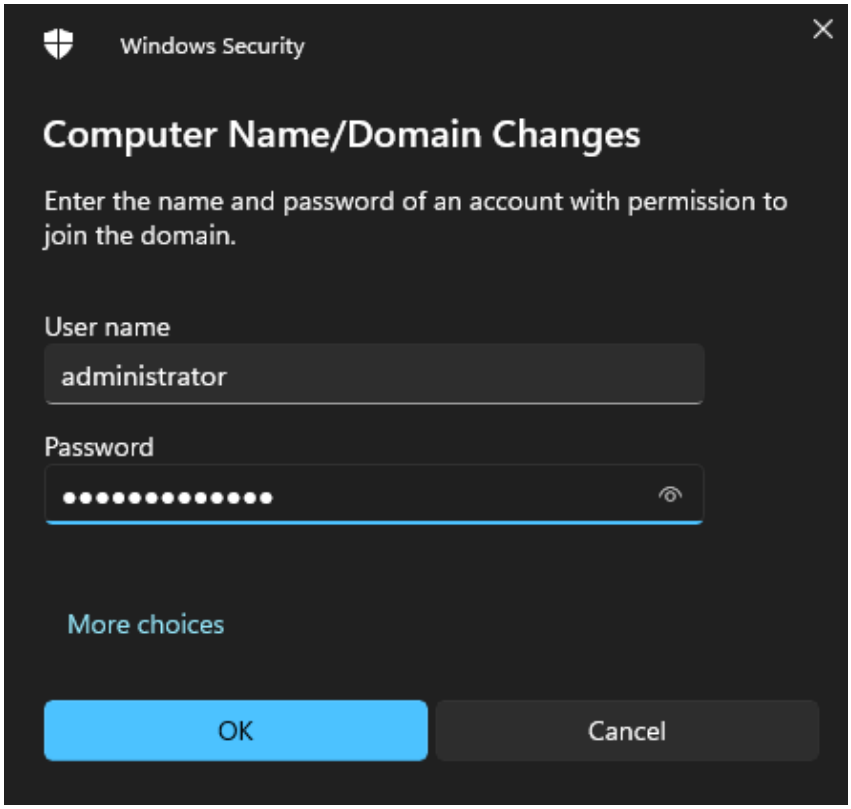

Figure 2.2: Joining Victim-PC to the Domain.

Policies → Password Policy. Here we can adjust settings like **Minimum password length, Maximum password age, and Password complexity requirements**. For a lab, we might *lower* requirements for convenience (e.g. minimum length 4, no complexity). We set **Account Lockout Policy** as well (lockout after 5 bad attempts, for 15 minutes) to mitigate brute force.

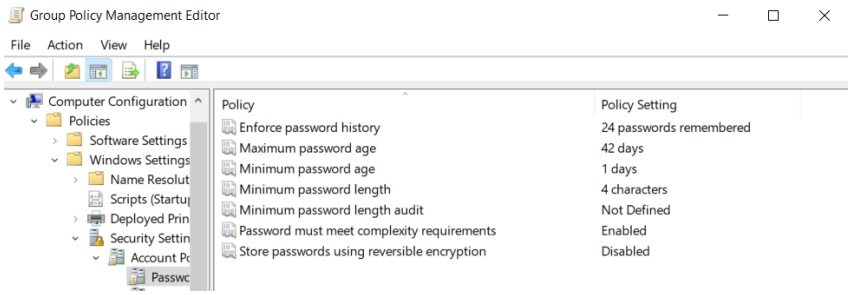

Figure 2.3: Password policy.

Step 2: Create a GPO for Advanced Auditing

To detect attacks and have logs in Splunk, we need detailed Windows auditing. Instead of using the basic audit settings, We use **Advanced Audit Policy Configuration** (introduced in Win7/2008+). We start by Creating a new GPO named **"Audit Settings Policy"**, and link it at the domain. In the GPO, We go to **Computer Configuration → Policies → Windows Settings → Security Settings → Advanced Audit Policy**

Configuration → Audit Policies. Enable auditing for the following categories (set to "Success and Failure" where appropriate):

- **Logon/Logoff**: Audit Logon (Success+Failure), Audit Account Lockout (Success).
- **Account Management**: Audit User Account Management (Success+Failure) – this logs events like user creation (Event ID 4720), group membership changes (4728), password resets, etc.
- **DS Access**: Audit Directory Service Access and Changes (to catch AD object changes).
- **Privilege Use**: Audit Sensitive Privilege Use (Success+Failure) – e.g. events when a user assigns privileges.
- **Object Access**: Audit File Share, Audit Registry, etc., if we want to detect file access (less critical for our focus).
- **Policy Change**: Audit Audit Policy Change (Success+Failure) – logs changes to audit policies themselves (Event 4719: system audit policy changed).
- **Account Logon**: Audit Kerberos Authentication Service and Ticket Operations (to catch Kerberoasting or golden ticket usage).
- **Detailed Tracking**: Audit Process Creation (Success) – if not using Sysmon, enabling this will log process start events (4688) which can be useful to catch malicious processes.

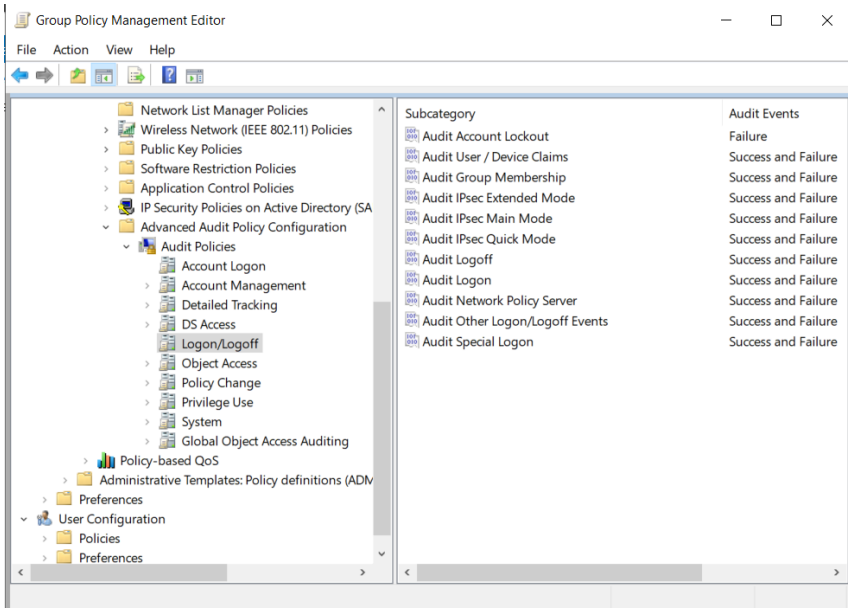

Figure 2.4: Configure a GPO for Logon/Logoff.

Enabling these will generate a lot of events, but in a lab it's fine. Microsoft recommends focusing on **privilege use and critical events** to avoid overload. These audit settings will ensure our DC and clients log the events we need to detect attacks.

We apply the GPO and run `gpupdate /force` on DC and client to apply immediately.

Step 3: Deploying LAPS

Microsoft **Local Administrator Password Solution (LAPS)** is a crucial hardening tool. LAPS ensures each domain-joined computer's local Administrator account has a unique, randomized password that is regularly rotated and stored securely in AD. This prevents attacks where one local admin password unlocks many machines. In a lab, implementing LAPS is highly educational:

- **Install LAPS software** on the domain controller (management tools) and on domain computers (client-side extension). Download LAPS (Microsoft's MSI) and install the management tools on DC and the client component on the Win10 PC.
- **Extend the AD Schema for LAPS** to add attributes for the password storage. On the DC, import the LAPS PowerShell module and run `Update-LapsADSchema`.
- **Set permissions** so that computers can update their password attribute and designated admins can read it. E.g., grant "SELF" permission for each computer to update its own `ms-MCS-AdmPwd` attribute, and create a security group (like "LAPS Readers") that has read access to those attributes.
- **Group Policy configuration**: Create a GPO for LAPS (or use an existing one). Under Computer Configuration → Administrative Templates (after installing LAPS .ADMX) → LAPS, enable "Password Settings: Do not allow password expiration time longer than required by policy" and set the desired password complexity/length. Also enable "Enable local admin password management". This GPO will instruct domain computers to randomize their Administrator password periodically (e.g. every 30 days).
- After a gpupdate, each computer will generate a random password for the built-in Administrator and store it in AD (attribute `ms-MCS-AdmPwd`). we can test this: on the DC, open ADUC, enable "Advanced Features" (View menu), find the computer object, and in its attributes we'll see `ms-MCS-AdmPwd` with a value (the password). Use the LAPS UI or PowerShell to retrieve passwords as an admin.

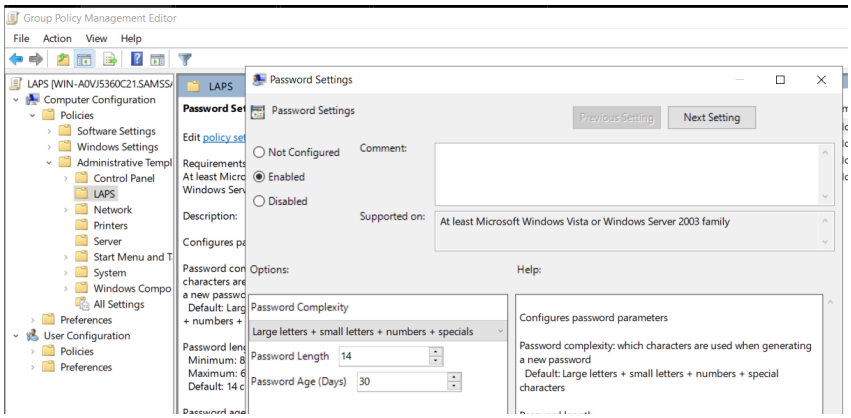

Figure 2.5: Password Settings in LAPS.

Implementing LAPS significantly improves security by removing the risk of lateral movement via shared local creds. Even in a small lab, it's worth doing to learn the process.

At this point, AD is set up with a domain, users, computers, and policies enforcing a baseline of security (password policy, auditing, LAPS, firewall, etc.). We will now integrate Splunk to collect logs from the DC and the client.

Chapter 3

Monitoring and Logging with Splunk

Centralized logging and monitoring are critical for security. In our lab, we use **Splunk** as a SIEM to aggregate Windows event logs and detect suspicious activities. We will install the Splunk **Universal Forwarder** on the Windows machines to send their logs to the Splunk server, then configure Splunk to index and alert on those logs.

3.1 Installing Splunk Universal Forwarder on Windows Hosts

Splunk Universal Forwarder (UF) is a lightweight agent that sends logs to the Splunk indexer. We need to install it on both the domain controller (DC01) and the Windows 10 client.

Step 1: Download Splunk UF

On each Windows machine (DC and client), download the Splunk Universal Forwarder installer from Splunk's site. (It's available under Splunk > Free Trials and Downloads > Splunk Universal Forwarder for Windows 64-bit MSI). The file will be something like `splunkforwarder-<version>-x64.msi`.

Step 2: Install Splunk UF

Run the MSI installer on each machine. During installation, accept the license and choose **Installation Options**:

- **Install for Everyone** (all users) and as a system service (default).
- Opt for **"Enable forwarding to Splunk Enterprise"** (on-premises) when prompted.
- It will ask for the **Splunk server (Indexer) IP and port** – enter the IP of our Splunk server (e.g. 192.168.1.10) and port **9997** (the receiving port we enabled).
- Set the **administration credentials** for the forwarder (it will create a local Splunk user on the machine). we can use something simple or the same across hosts for lab ease. This is not the same as the Splunk Enterprise credentials – it's for the forwarder service if we need to manage it.

- No need to configure as deployment client for this lab.
- Complete the installation.

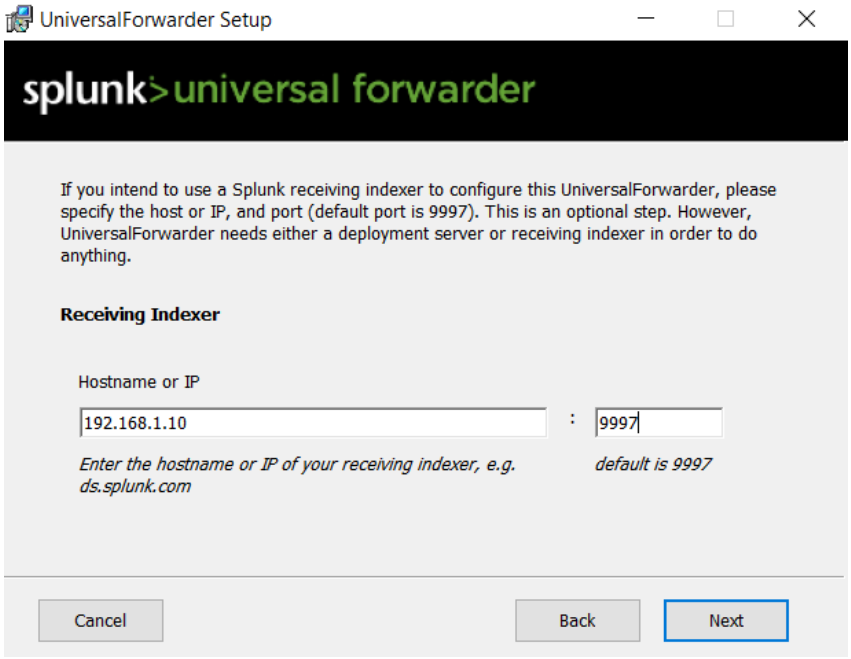

Figure 3.1: Splunk UF Configuaration.

Once installed, the Splunk Forwarder service will run in the background.

Step 3: Configure Inputs (Logs to send)

By default, the forwarder doesn't send all Windows logs until we configure inputs. We want to send key Windows Event Logs and potentially Sysmon logs to Splunk. We'll do this by creating an **inputs.conf** on each forwarder:

1. On the Windows machine, open a PowerShell or cmd **as Administrator**.
2. Navigate to the UF's directory, for example: `C:\Program Files\SplunkUniversalForwarder\etc\s`
3. Create a new file **inputs.conf** in that `local` folder (do not edit the default one in `...system\default`). we can use Notepad (runing Notepad as admin so we can save in that folder).
4. In inputs.conf, specify the logs to collect. For example, to collect the main Event Logs and Sysmon, use:

```
[WinEventLog://Application]
index = endpoint
disabled = false

[WinEventLog://Security]
index = endpoint
```

```
disabled = false

[WinEventLog://System]
index = endpoint
disabled = false

[WinEventLog://Microsoft-Windows-Sysmon/Operational]
index = endpoint
disabled = false
renderXml = true
```

This config tells the forwarder to send Application, Security, System, and Sysmon Operational log channels to the index named "endpoint" on Splunk. Ensure the `index` name matches the one we created on Splunk (we used "endpoint"). The `renderXml = true` for Sysmon makes events detailed. If we didn't install Sysmon, we can omit that stanza or install Sysmon next (see below).

Save the file. Then **restart the Splunk Forwarder service** to apply changes (via `services.msc` or `splunk restart` command). For example, run `net stop SplunkForwarder` and `net start SplunkForwarder` or use the Services control panel to restart "SplunkForwarder" service.

Repeat this on both the DC and the Win10 client. Alternatively, we could have done this during install via command line or used Splunk Deployment Server for mass config, but editing inputs.conf manually is fine for a couple of hosts.

Step 4: Install Sysmon (Optional but Recommended)

Sysinternals Sysmon is a tool that provides detailed system event logs (process creation, file changes, registry, network connections, etc.), which are very useful for security monitoring. In our lab, installing Sysmon on the DC and client will enhance visibility (especially to detect things like Mimikatz or suspicious processes). To install:

- Download Sysmon (Sysmon64.exe) from Microsoft Sysinternals.
- On each machine, run an admin command prompt: `Sysmon64.exe -i -accepteula -h md5` (and optionally specify a configuration file for filtering if we have one – many community Sysmon configs exist, but for lab we can use default which logs most events).
- This will install Sysmon as a service and start logging to **Event Viewer → Applications and Services Logs → Microsoft → Windows → Sysmon → Operational**.
- We already included that channel in our inputs.conf. After installing Sysmon, the forwarder will start sending those events too. For example, Sysmon logs process access events (Event ID 10) that can catch techniques like Mimikatz accessing LSASS.

Step 5: Verify Logs in Splunk

Now go back to the Splunk server UI (`http://192.168.1.10:8000`). Search for data from these forwarders. we can run a Splunk search like: `index=endpoint host="DC01"` and time = last 15 minutes. we see events coming in from the domain controller. Check

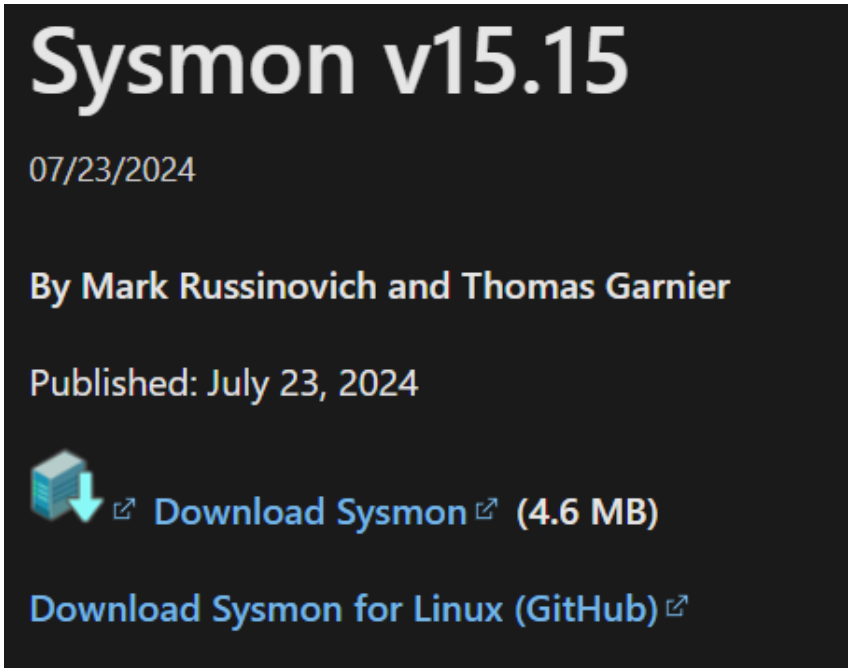

Figure 3.2: Install Sysmon .

`index=endpoint host="Victim-PC"` (or whatever the client hostname is) for the workstation logs. If nothing appears, troubleshoot: ensure the forwarder service is running and the port 9997 is open. we can also check the UF logs under `\SplunkUniversalForwarder\var\log\splu` for connection messages.

In Splunk, it helps to install the **Splunk Add-on for Windows** (which knows how to parse Windows events) and the **Splunk Security Essentials** app (for use-cases and sample detections). These are optional, but in a lab SIEM, they provide pre-built knowledge. Since our focus is just getting logs in and creating custom alerts, we can proceed without them if desired.

At this point, Splunk is ingesting events from both machines. Events like user logons, group changes, etc., that we configured auditing for will be collected. We will later use Splunk searches to detect malicious patterns.

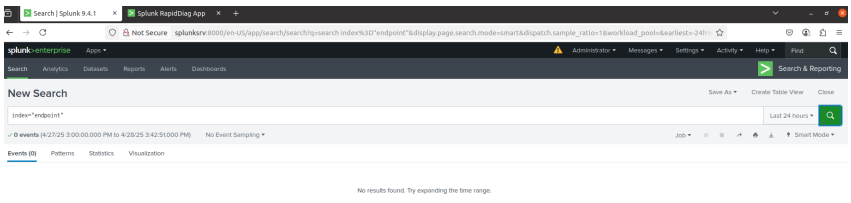

Figure 3.3: Splunk Logs.

3.2 Monitoring Critical Events and Alerts in Splunk

Before jumping into attack simulations, it's good to set up some basic **dashboards or alerts** in Splunk to monitor the AD environment:

- **Domain Controller logs**: We use Splunk to filter Security Event Logs on the DC. For example, monitor **Event ID 4720** (user account created), **4728** (member added to a security group), **4625** (failed logon), **4625** (successful logon), **4740** (account lockout), **4719** (audit policy change), **1102** (audit log cleared) etc. These can be indicators of attacks or policy changes.

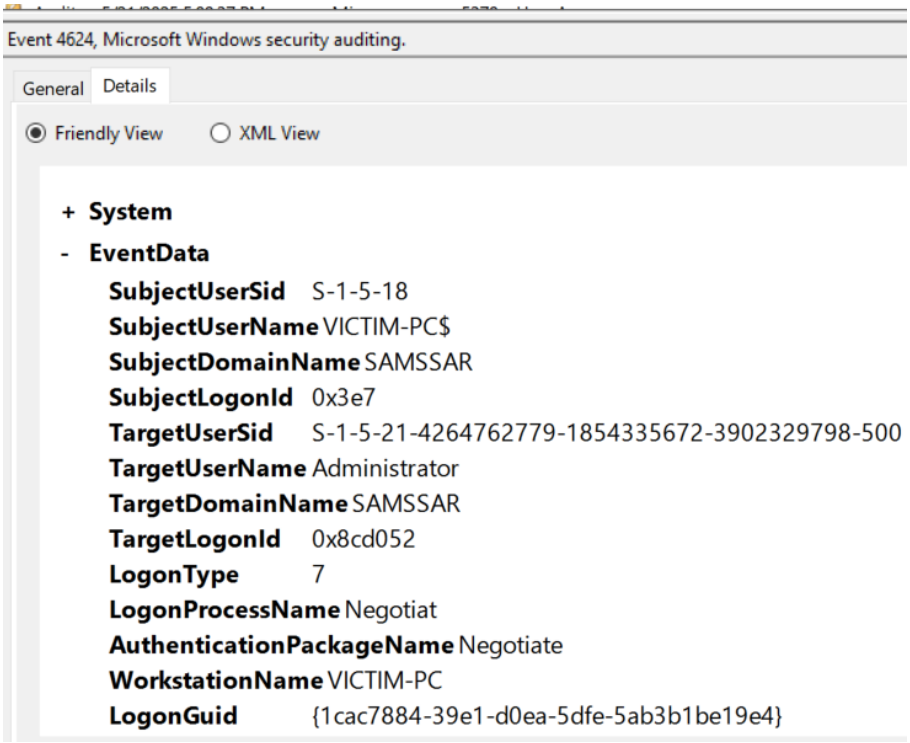

Figure 3.4: Successful Logon.

- **Windows 10 client logs**: Monitor for suspicious activity like multiple failed logons (4625), or local logon by an admin account, etc. If Sysmon is enabled, look for Sysmon Event ID 1 (process creation) for odd processes (like `mimikatz.exe` execution).
- **Dashboard**: We can also create a simple Splunk dashboard showing logon attempts, account changes, etc. For example, a panel for "Recent failed logon attempts" (`index=endpoint EventCode=4625 | timechart count by host`) or "New user or group changes" (filter EventCode 4720,4728, etc.).

Splunk's role in the lab is to act as the **central SIEM**, aggregating logs for analysis. An often-cited Microsoft best practice: *"Many compromises could be discovered early if logs were monitored and alerted"* – our goal is to simulate that monitoring.

We will come back to Splunk when we simulate attacks (Section 6) to see how to detect them. Now, with monitoring in place, let's simulate some attacks on the AD environment using the Kali machine and specialized tools.

Chapter 4

Attack Simulation with Kali Linux

In this section, we step into the shoes of an attacker to test our AD lab's defenses. Using Kali Linux, we will perform reconnaissance and attacks against the Windows environment using tools like **BloodHound**, **Mimikatz**, and **CrackMapExec**. This will help us identify security gaps and generate logs that we can detect with Splunk.

> **Disclaimer:** All activities here should be done in isolated lab environment **only**. These techniques are dangerous on real networks. Ensure you have permission and containment (which we do, since it's our lab).

4.1 Reconnaissance with BloodHound (Active Directory Mapping)

BloodHound is an AD reconnaissance tool that reveals hidden relationships and attack paths in AD by using graph theory. It's popular among attackers and penetration testers for mapping out who can do what in a domain.

Step 1: Run BloodHound data collector

BloodHound has two parts: the data collector (called **SharpHound** if using Windows, or Python/PowerShell alternatives) and the GUI interface (running on Neo4j database). The easiest method: use the **SharpHound** executable on a domain-connected system to gather data. In our lab, we log onto the Windows 10 client as a domain user. We Copy `SharpHound.exe` (from the BloodHound GitHub release) onto the Win10 machine. Then we run it to collect data. For example, open CMD on Win10 and run:

```
SharpHound.exe -c all -d samssar.local -ZipFileName loot.zip
```

This will collect all available AD info (users, groups, group memberships, privileged rights, sessions, ACLs, etc.) and output a `loot.zip` file with the data. SharpHound uses the domain credentials of the user running it (so if we run as a normal user, it gathers what that user can query via LDAP – which is actually a lot since many queries don't require admin rights).

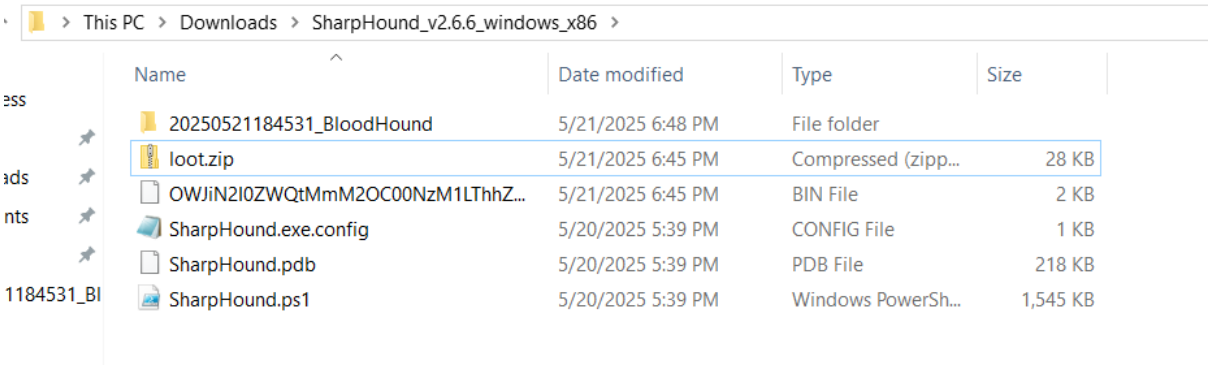

Figure 4.1: Generate the ZIP File.

Step 2: Analyze data in BloodHound

Now we transfer the generated ZIP to Kali (we could use SMB, a Python HTTP server, or simply use WinSCP). On Kali, we start the **Neo4j** database: `neo4j console` and we browse to `http://localhost:7474` to ensure it's running (default user/pass is neo4j/neo4j, it will prompt to change it). We launch the BloodHound GUI (`bloodhound` command in Kali). Log in with the Neo4j credentials. Then we use the **Upload Data** button in BloodHound to import the `loot.zip` from SharpHound.

BloodHound will visualize the domain to identify potential attack paths. For example, BloodHound might show that a certain user has local admin rights on a machine where a Domain Admin is logged in, implying a path to DA. Or it might reveal weak ACLs. This is **valuable for defenders** to understand, as it shows us misconfigurations. In our simple lab, with only a few users, we might not have complex paths, but if we made a few groups and delegated rights, BloodHound could highlight them. Remember, BloodHound can reveal relationships that are not obvious, aiding an attacker in privilege escalation.

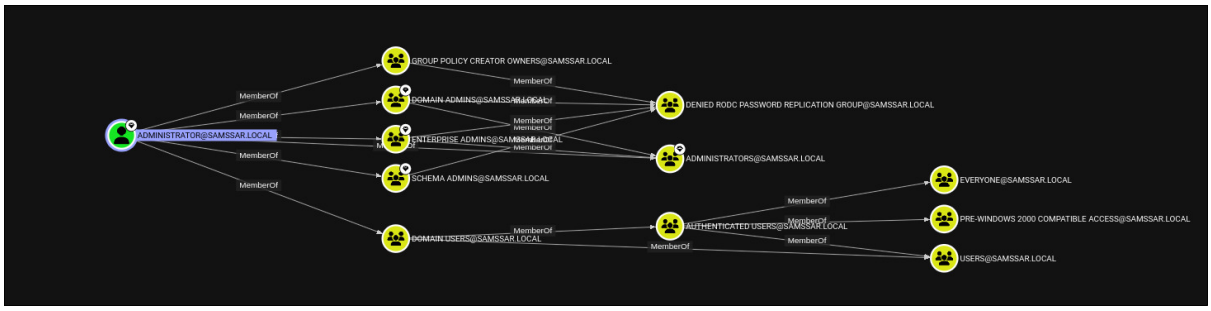

Figure 4.2: Privilege Escalation Path.

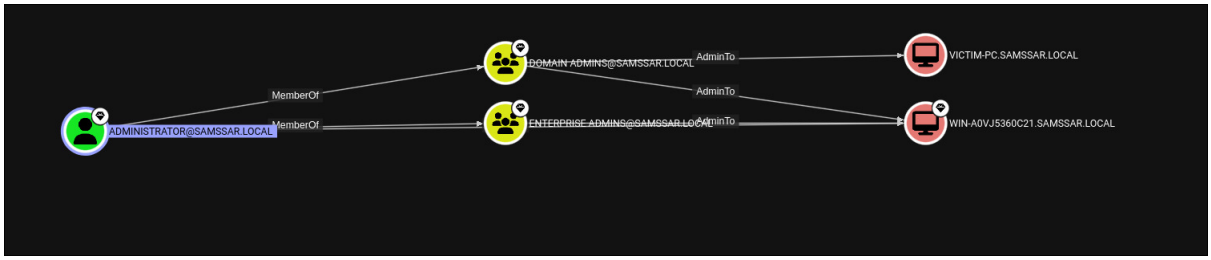

Figure 4.3: Administrator's Direct Admin Rights to Domain Workstations.

Now that recon is done, let's simulate an **active attack** – dumping credentials with Mimikatz.

4.2 Credential Dumping with Mimikatz

Mimikatz is a famous post-exploitation tool that can dump passwords and hashes from Windows memory. Attackers use it to perform actions like extracting clear-text credentials, NTLM hashes, Kerberos tickets, and even creating **Golden Tickets** (forging Kerberos tickets). We will use Mimikatz to dump credentials from our Windows machines.

Step 1: Gain a foothold

In an actual attack, an adversary would first compromise a machine (e.g. via phishing or an exploit) to run Mimikatz. In our lab, we can simulate this by using the Domain Admin account (since we control everything). But for a more realistic scenario, let's assume the attacker compromised the Windows 10 client as a standard user and then managed to escalate to local admin on that machine (for instance, via some vulnerability or by guessing the local admin password if it was weak – though LAPS would mitigate that). We will simply ensure we have **Administrator rights on the target machine** to run Mimikatz (Mimikatz requires admin or SYSTEM privileges to access LSASS memory).

Step 2: Run Mimikatz

On the Windows 10 machine, we copy over the Mimikatz binary (`mimikatz.exe`). We run it as Administrator (right-click Run as admin). In the mimikatz console, then we execute the following commands:

- `privilege::debug` – this grants mimikatz the debug rights it needs (should return "Privilege '20' OK").
- `sekurlsa::logonpasswords` – this dumps credentials of currently logged on users from LSASS memory. We might see the clear-text password of any user who is logged in or recently authenticated, as well as NTLM hashes and Kerberos tickets.

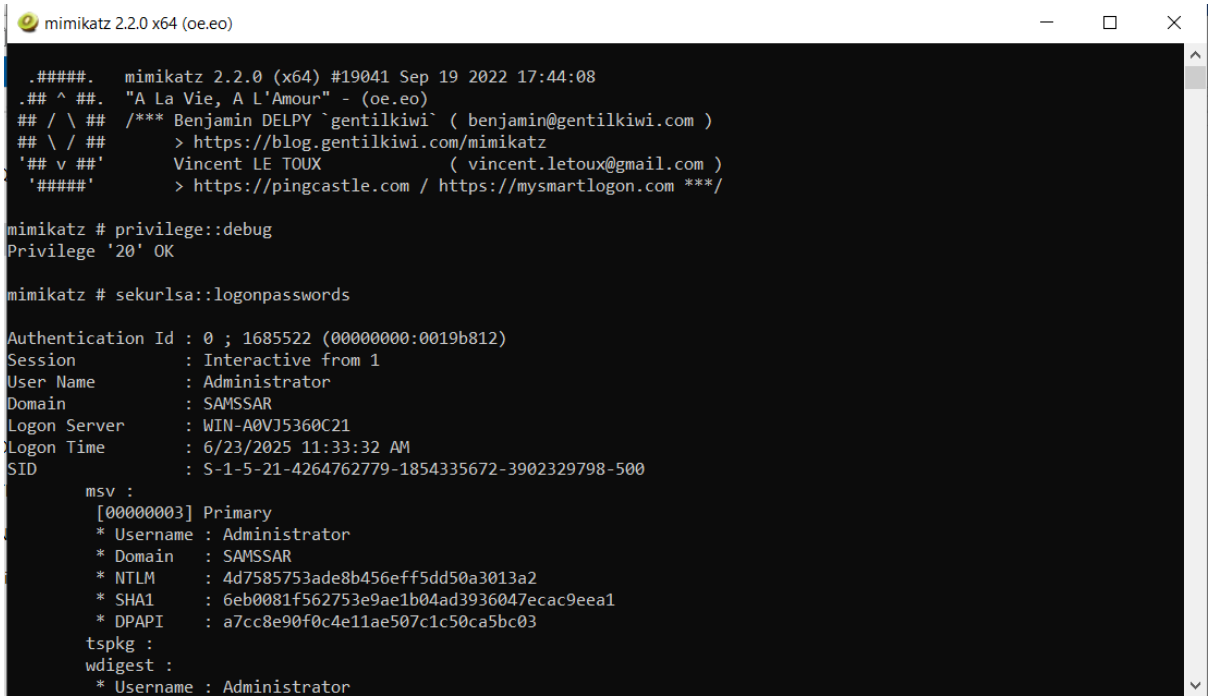

Figure 4.4: Credential Dumping by Mimikatz.

If a domain admin (administrator) was logged in, we might directly get the DA's password in cleartext (since by default Windows 10/Server 2016+ still keep creds in memory for a while). If not, we will at least get the NTLM hash of any accounts(in our we get them both). Mimikatz can also do `lsadump::lsa /inject` or `lsadump::dcsync` if run on a DC to get ALL domain hashes, but let's not fully compromise everything yet.

What we have now is what an attacker covets: credentials. For example, we may have obtained the NTLM hash of the domain admin account. We can attempt a **Pass-the-Hash (PtH)** or use the hash to impersonate the user.

Step 3: Lateral Movement with Pass-the-Hash

Instead of cracking any hashes (which we can attempt if we captured a weak password hash), an attacker often uses the hash directly. Using the Kali machine and **CrackMapExec**, we can test the hash. For instance, on Kali, run:

```
crackmapexec smb 192.168.1.5 -u labadmin -H <ntlm_hash>
```

This tells CME to authenticate to the DC (192.168.1.5) via SMB with user labadmin and the hash. If the hash is correct, it will show success and that we have admin access. Essentially, we've performed a **pass-the-hash** attack – using the stolen hash to authenticate without knowing the plaintext password.

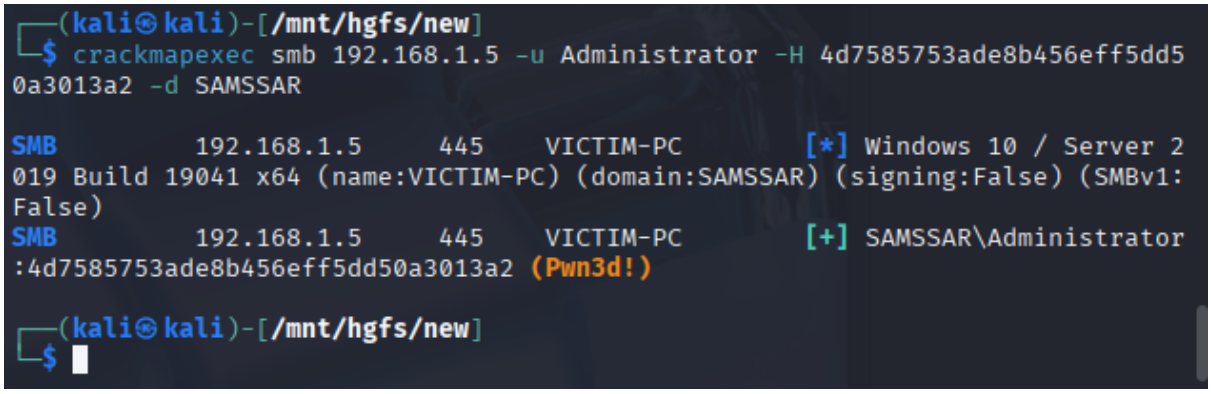

Figure 4.5: Pass-the-Hash.

Alternatively, we could use Mimikatz on the Windows machine to perform PtH directly via `sekurlsa::pth` (Pass-the-hash) which will create a new process with the stolen token, but using CME from Kali keeps the attack steps on the Kali side.

Step 4: Dumping AD Secrets

With domain admin credentials (either via hash or password), the attacker can now fully compromise AD. For example, using **Mimikatz's dcsync** feature to impersonate the DC and pull the **NTDS.dit** credentials database. On a DC (or from Kali using Impacket's `secretsdump.py`), one could do:

```
mimikatz.exe "lsadump::dcsync /domain:lab.local /user:Administrator"
```

This would output the NTLM hash (and possibly Kerberos keys) of the Administrator (or any user specified, even krbtgt account). For instance, dumping the `krbtgt` account hash is the first step to forging a **Golden Ticket** – a Kerberos TGT that can authenticate as any user in the domain. Mimikatz can then do `kerberos::golden` to create a golden ticket with that hash. In a lab, we can try this for learning, but we are cautious as it can get complex. A Golden Ticket basically gives persistence (the attacker can always re-auth as DA until krbtgt password is rotated).

For our purposes, assume the attacker has successfully dumped some credentials (say the Domain Admin's cleartext password or hash). The mission now is accomplished from the attacker perspective – they have domain control.

We have generated several **attack artifacts** that should be visible in logs:

- Mimikatz executing and touching LSASS (if Sysmon is logging, we'd have an Event ID 10 for LSASS access with suspicious call trace).
- Possibly Windows Security Event ID 4688 (process creation) for mimikatz.exe if process auditing is on.
- If using DCSync, Event ID 4662 on the DC with operation "Directory Service Access" on specific objects might be logged.
- A bunch of logon events: pass-the-hash via SMB results in Event ID 4624 on the DC (type 3, NTLM authentication) that might have an odd LogonProcess (such as "NtLmSsp") and same user from a new host.

- If a Golden Ticket was used, Kerberos service ticket requests with RC4 encryption might appear (since a forged ticket often uses RC4 by default, which is unusual in modern environments).

We will use Splunk to detect some of these in Section 6.

Chapter 5

Security Hardening Best Practices

To secure the AD lab environment, we implement several best practices. Some we already touched on (Group Policy settings, LAPS, etc.). Here we summarize and add other hardening measures:

- **Use LAPS for Local Admin Accounts**: We installed LAPS in Section 2.4. Verify it's working: each computer's local admin password should be unique and rotate per policy. This stops an attacker from using one machine's local creds to move laterally to another.
- **Apply Least Privilege**: Using least privileged accounts for daily tasks. Domain Admin accounts should not be used to log into regular workstations (to avoid putting DA creds on a potentially compromised box).
- **Secure Administrative Workstations**: Ideally, have a separate hardened admin VM (not accessible to attackers) for performing domain admin tasks. In our small lab, this could simply mean only log into DC directly for admin tasks and not doing so on the client.
- **Update and Patch Systems**: Keeping all machines updated (we did initial Windows Updates, but continue to apply updates). Many attacks (like EternalBlue for SMB) are mitigated by patches. Ensuring antivirus or Windows Defender is running on Windows machines – this can catch known malware (Defender would flag mimikatz if not bypassed). we can even enable Attack Surface Reduction (ASR) rules or Credential Guard in Windows 10 to make credential dumping harder.
- **Firewall Configuration**: We enabled Windows Firewall; ensure it's configured to restrict lateral movement. For instance, block SMB and WMI from client-to-client. Domain controllers need SMB for admin tasks, but regular clients sharing files might be unnecessary in a lab. Only allow what is needed. The firewall can also block outbound connections for unauthorized apps (e.g. block Kali IP if detected, though that's tricky dynamically). The key is internal segmentation – even within a flat lab, we can simulate by mentally assigning roles (DC vs client) and ensuring clients cannot easily talk to each other except to the DC.
- **Secure Protocols**: Disabling old protocols that attackers abuse. For example:

  - **NTLMv1**: Ensuring NTLMv1 is disabled (it is by default on modern Windows). Only NTLMv2 should be allowed. (Group Policy: Network Security: Restrict NTLM: deny NTLMv1).
  - **SMB1**: Disabling SMBv1 on all systems (it's off by default in latest Windows 10/2019+, but check). SMBv1 is outdated and a security risk (e.g. WannaCry used it).
  - **LDAP**: If we want to be thorough, enforce LDAP signing so that LDAP simple binds aren't allowed (prevents MiTM on LDAP). This might interfere with tools like BloodHound if not using LDAPS, so in a learning lab we might skip.

- **Group Policy Hardening**: Beyond what we set, consider implementing:
  - **User Rights Assignments**: e.g. via GPO, denying "local logon" to DC for all but admins, denying "remote logon" to servers for non-admins, etc.
  - **Protected Groups**: AD has a concept of "Protected Users" group which disables caching creds and older auth protocols for members (could put admin accounts in there for extra security).
  - **Audit Policy**: We enabled a broad audit. Review logs regularly to fine-tune (e.g. filter noise events).
  - **Sysmon**: We deploy Sysmon with a good config that filters out noise and logs relevant events (like code injection, process creation with command-lines, etc.). Olaf Hartong's SysmonModular config is a great resource.

- **Securing Domain Controllers**: Domain Controllers are critical – treat them as Tier 0 assets. Only Domain Admins should log on to DCs, and no one should surf the internet or check email from a DC. In the lab, we don't use the DC for things like web browsing or downloading random tools. This reduces exposure.

- **Backup and Recovery**: Maintaining backups of AD (System State backup or VM snapshot). If an attack like ransomware hits, we need to restore AD. Our lab uses snapshots as a form of backup.

- **Security Updates and Monitoring Tools**: Installing any available security monitoring tools: e.g. Windows Defender Advanced Threat Protection or even open source Helk or OSSEC. In our case, Splunk is our main monitoring tool.

Security hardening is an ongoing process. The above covers key areas (accounts, credentials, networking, auditing). Finally, we focus on how to **detect** and respond to threats using Splunk and the logs we've collected.

Chapter 6

Threat Detection and Response with Splunk

Now that we've set up logging (Splunk) and executed some attacks, we can use Splunk to detect those malicious activities. We'll also outline how to respond once such threats are detected. Effective detection involves knowing what patterns in logs indicate an attack, and creating alerts for them.

6.1 Detecting Active Directory Attacks in Logs

Let's map our simulated attacks to log events and show how to search/alert on them in Splunk:

- **Mimikatz Execution (LSASS Dump)**: Mimikatz often triggers identifiable events. If Sysmon is running, one of the best indicators is Sysmon **Event ID 10** (process access) where the target process is LSASS and the source process is something unusual. For example, the Splunk query might be:

```
index=endpoint EventCode=10 TargetImage=*\\lsass.exe
GrantedAccess="0x1010"
```

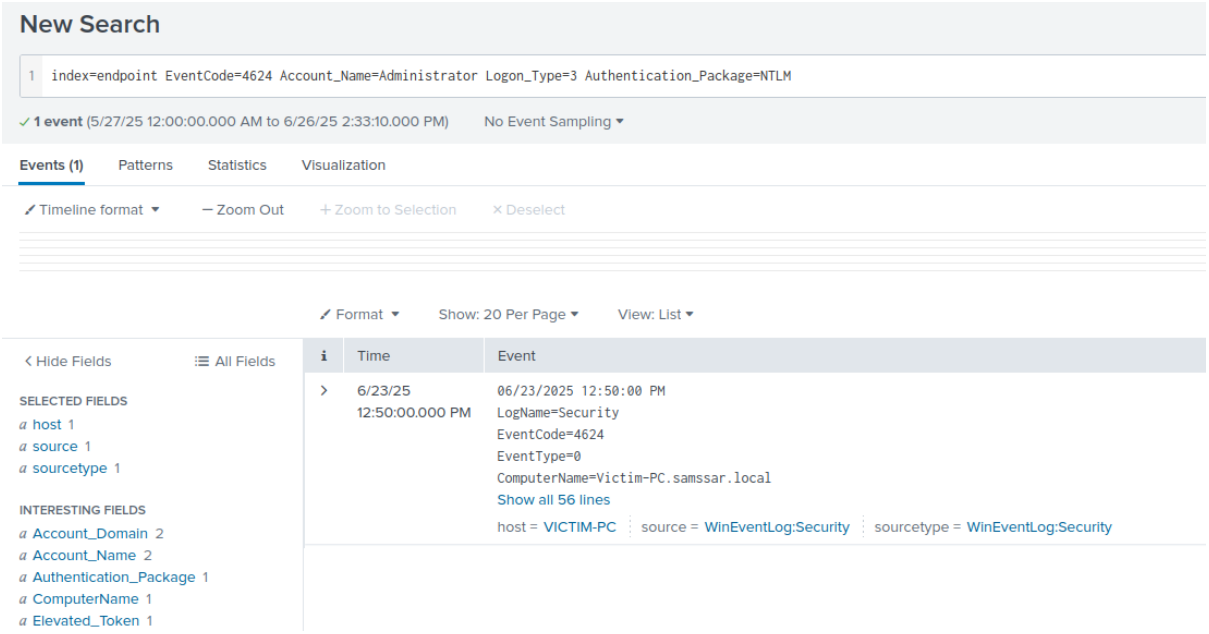

Figure 6.1: Mimikatz Execution Event

This looks for a process requesting access rights 0x1010 (READ/WRITE) on LSASS – which is typical of Mimikatz or similar tools. Splunk's Threat Research Team provides a detection analytic for this LSASS access pattern. If found, this is a high-fidelity sign of credential dumping. **Alert** on any such event, since normal processes shouldn't open LSASS with those rights.

Additionally, if Script Block Logging (4104 events) is enabled and Mimikatz was run via PowerShell (Invoke-Mimikatz), we could search Security Event Log **ID 4104** for Mimikatz keywords. Splunk has a detection for Mimikatz PowerShell usage by looking at script block contents. For example, search for `EventCode=4104 AND Message=*mimikatz*`. That would catch the PowerShell tooling.

Windows Security Event ID 4688 (process creation) might show "mimikatz.exe" launched; we could alert on that by maintaining a watchlist of known hacker tools. But that's easy to evade by renaming the binary. The behavioral detection via Sysmon is more robust.

- **Pass-the-Hash / Lateral Movement**: When we used CME with a hash or performed remote execution, the Windows logs on the target (DC01) recorded a logon event. **Event ID 4624** (successful login) with Logon Type 3 (network) and using NTLM authentication, with a strange source workstation name, could indicate PtH. One heuristic: If we see a logon event where **Security ID** is a domain admin, Logon Type 3, and **Authentication Package: NTLM** instead of Kerberos (in a domain environment, domain admin authentications should normally use Kerberos), that's suspicious. Splunk's ESCU (security content) has a search to detect Pass-the-Hash by looking for NTLM logons for accounts that usually use Kerberos.

  We can craft a Splunk query like: find domain admin account logons using NTLM. For example:

```
index=endpoint EventCode=4624 Account_Name=Administrator
Authentication_Package=NTLM Logon_Type=3
```

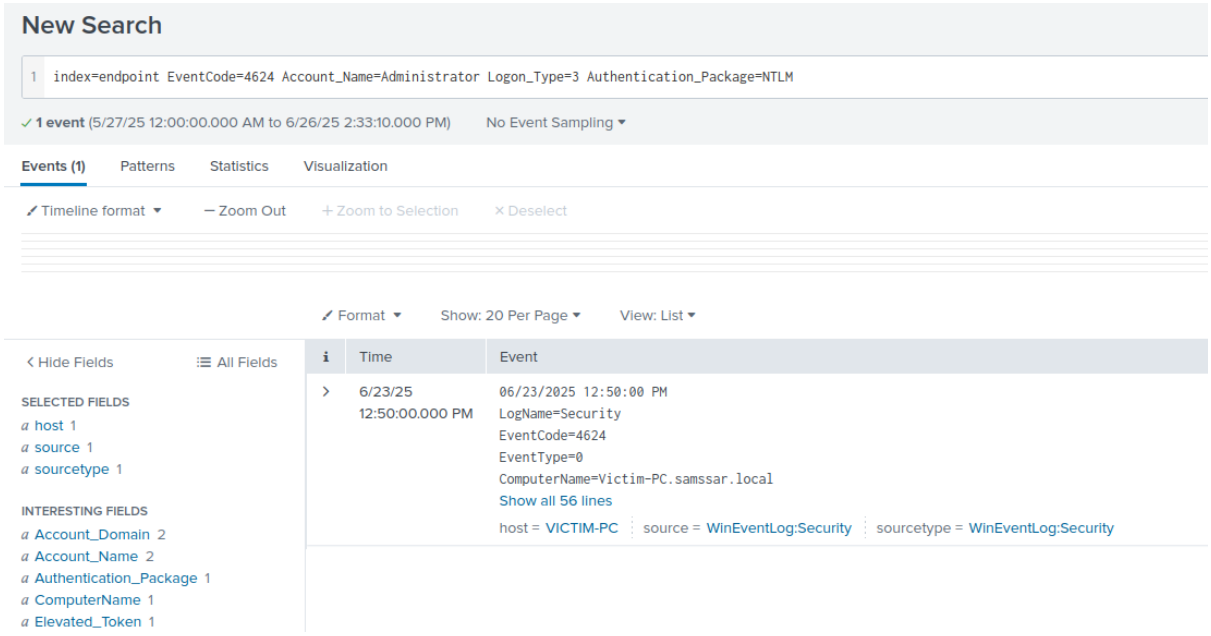

Figure 6.2: NTLM Network Logons for Administrator

and maybe exclude cases where Workstation Name is the DC itself (because DC-to-DC might use Kerberos for certain tasks). If such an event appears from a new host (like the Kali IP or Victim-PC), raise an alert. Similarly, multiple failed logons (4625) followed by a success for the same account could indicate password guessing followed by success.

- **DCSync Attack**: DCSync (via Mimikatz or Impacket's secretsdump) impersonates a domain controller to pull credentials. It doesn't create typical interactive logons, but it does trigger directory replication events. Looking for **Event ID 4662** on the DC with **Operation:** "Replication", or specifically the presence of the GUID `89e95b76-444d-4c62-991a-0facbeda640c` which corresponds to "GetNCChanges" (replication) operation on AD (this is an advanced detection). Also, Event 4625 might show an account (like "ATTACKER$") failing to authenticate if they didn't use proper sync privileges. An easier approach: Monitor membership of the "Replication Role" – but that's static. There is a known Microsoft ATA alert for DCsync. In Splunk, we might not catch DCSync easily without something like Azure ATP or by enabling auditing of Directory Services Access (we did) and then filtering 4662 events where properties accessed include "NTLM-Hash". This is complex; for a lab, just be aware DCSync can be detected by careful log analysis (Microsoft's Advanced Threat Analytics identifies it via unusual replication requests).

- **Golden Ticket (Kerberos anomalies)**: If an attacker uses a Golden Ticket, one indicator is a Kerberos TGT with an unusually long lifetime or an invalid certificate signature. But one easy catch: if we see **Event ID 4769** (Kerberos Service Ticket

Request) where the encryption type is **RC4-HMAC** for a Kerberos ticket in an environment where all clients are modern (Windows 10+ prefer AES), it could indicate a forged ticket or an attack like Kerberoasting. Splunk's research content has a detection for Golden Ticket by looking for RC4 usage in TGT requests. we could search:

```
index=endpoint EventCode=4769 TicketEncryptionType=0x17
```

(0x17 is RC4) and Service Name = krbtgt, for example. If found for a privileged account, that's a red flag. Another sign is Event 4624 where Logon Type = 9 (which means "NewCredentials" logon, often used with pass-the-ticket attacks).

- **BloodHound Enumeration**: Detecting BloodHound is tricky. However, BloodHound's SharpHound generates a high number of LDAP queries in a short time. We enabled **Audit Directory Services** in advanced auditing, we see many Event ID 4662 (Operation LDAP search) from a single user. If a normal user account suddenly performs extensive LDAP queries (especially for admin group memberships, etc.), we could alert on that. Another approach is to monitor the **volume** of certain events per user. Splunk can do anomaly detection or baselining to alert on deviations.

- **Password Spraying/Brute Force**: This would manifest as a large number of **4625 (failed logon)** events, possibly across many accounts. If we see many 4625 with Reason "bad password" and different usernames from one IP (the Kali IP or a single host) in a short time, that's a spray. In Splunk, we can run a query to detect >N failures from one source in Y minutes. For example:

  ```
  index=endpoint EventCode=4625
  | stats count by Computer, WorkstationName, IpAddress, AccountName
  | where count > 10
  ```

  This could identify an IP or workstation that had >10 failed logins for various accounts, indicative of spraying. We can refine by looking for the same password used (but Windows log doesn't record password tried, only failures).

- **Privileged Group Changes**: We touched on this – any addition to Domain Admins (Event 4728 for the "Administrators" global group or 4728 for Domain Admins since it's a global security group) should alert. Splunk query:

  ```
  index=endpoint EventCode=4728 Group_Name="Administrators"
  ```

  alert and include who added whom.

- **Audit Log Clearance**: If an attacker clears Security log to cover tracks, Event **1102** is logged ("Audit log was cleared"). Always alert on Event ID 1102 on any host – it's rarely done legitimately. That event will show which user cleared the log. In Splunk: `EventCode=1102` as a high priority alert.

Many of these detections can be set as Splunk **Alerts** (with thresholds or real-time). Splunk can send an email or trigger a script when conditions meet. In a lab, we could just monitor manually or simulate an analyst reviewing the Splunk dashboard.

Splunk Security Essentials app provides out-of-the-box queries for many of these scenarios. For example, it has a use-case for detecting Mimikatz, pass-the-hash, golden ticket, etc., which one can adapt.

6.2 Incident Response in the Lab

Detection is only half the battle; responding is next. In our lab scenario, what would an incident response look like?

- **Isolate the compromised machine**:Splunk alerts that, say, Mimikatz was detected on **Victim-PC**, the first step is to contain. In a real network, we'd remove that machine from the network to stop the attacker from moving further. In our lab, this could mean shutting down the Victim-PC VM or disabling its NIC. "Isolate the compromised machine from the network to prevent the attacker from accessing other systems" is standard advice.
- **Alert and Gather Evidence**: Notify the admin team of the breach. Pause any other risky activity. Start collecting memory or disk if we want to practice forensics. Since this is a controlled lab, we might skip deep forensics and instead check logs to understand scope.
- **Eradicate the threat**: Removing any malicious files or tools (delete mimikatz.exe from the PC, for example). If the attacker created any backdoor accounts or added credentials (like Golden Ticket persist until krbtgt reset), address those.
- **Password Resets**: Assume credentials on that machine are compromised. Change the password of any user that logged onto Victim-PC (especially privileged accounts).
- **Monitoring Post-Incident**: Increase scrutiny after an incident. Enable even more verbose logging for a while. In Splunk, we could set up additional alerts. Ensuring that the attacker hasn't left scheduled tasks or trojans (check Task Scheduler, services, startup registry keys on compromised hosts).
- **Practice Restoring**: The environment was severely compromised, we restore it from snapshot. In our lab, this is easy – revert to a clean snapshot if things get out of hand. In production, we'd rely on clean backups.

Finally, this lab is perfect for practicing these detection and response steps repeatedly. We tried different attacks and see if our alerts catch them, adjusting as needed.

6.3 Using Splunk for Continuous Monitoring

To conclude, integrate Splunk into daily operations of our lab:

- **Dashboards** for AD health and security: e.g., login activity, changes, system errors.
- **Real-time Alerts** for critical events (we listed many). For example, an alert when a domain admin logs on interactively to a client machine during off hours, or an alert for any 1102 event.
- **Reports**: Schedule a daily security report of notable events.
- **Splunk Queries for Threat Hunting**: Periodically, runing searches for signs of known attack techniques even if no alert fired. E.g., searching for any use of the `Ntdsutil`, vssadmin, or unusual process names that might indicate other attack methods.

By combining proactive hardening (Section 5) with vigilant monitoring and alerting (Section 6), our AD lab becomes an effective training ground for cybersecurity. We can try to "beat" our defenses with Kali and then improve them.

Conclusion

In summary, setting up an AD lab with a domain controller, client, Splunk SIEM, and attacker machine allows us to simulate an enterprise environment and practice both offensive and defensive security. We installed and configured AD DS, set up a Splunk server and forwarders for centralized logging, and conducted attacks using tools like BloodHound, Mimikatz, and CrackMapExec. We then applied security hardening such as unique local admin passwords (LAPS), strong policies, and auditing, and used Splunk to detect malicious patterns (like Mimikatz's behavior in logs or pass-the-hash attempts). Finally, we outlined how to respond to incidents (e.g. isolate hosts and remediate). By following this guide, we can build a robust AD test lab and gain hands-on experience in both administering a secure AD environment and detecting/responding to threats within it.

Bibliography

[1] Splunk Enterprise, https://www.splunk.com/en_us/download/splunk-enterprise.html

[2] Kali Linux, https://www.kali.org/get-kali/

[3] Microsoft Sysinternals Sysmon, https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

[4] BloodHound, https://github.com/BloodHoundAD/BloodHound

[5] Mimikatz, https://github.com/gentilkiwi/mimikatz

[6] Impacket, https://github.com/fortra/impacket

[7] Microsoft LAPS, https://learn.microsoft.com/en-us/windows-server/identity/laps/laps-overview

[8] Olaf Hartong, SysmonModular, https://github.com/olafhartong/sysmon-modular

[9] Microsoft Advanced Threat Analytics, https://docs.microsoft.com/en-us/advanced-threat-analytics/what-is-ata