1. Project Overview

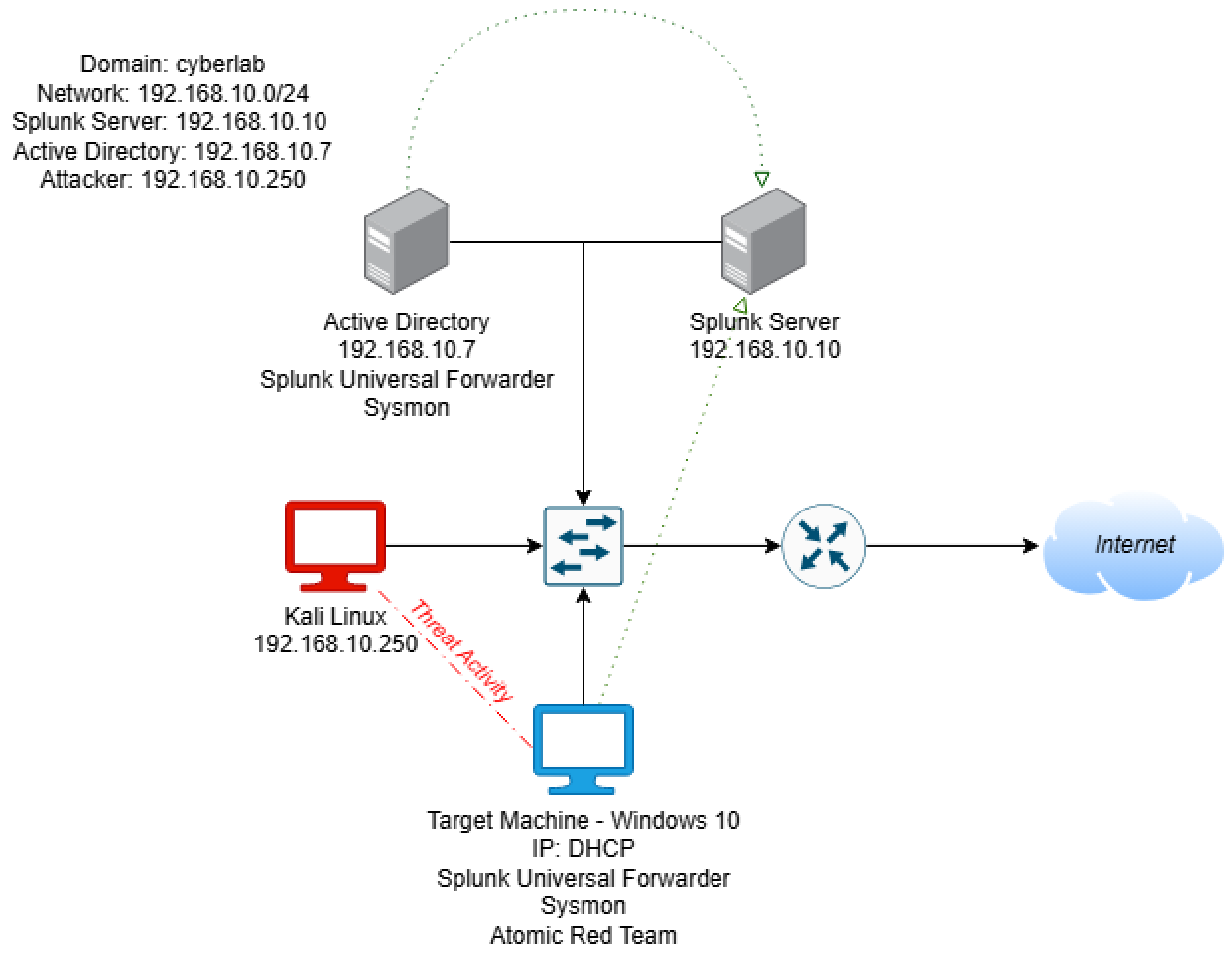
The goal of this project was to build a fully functioning SOC home lab that simulates enterprise environments and generates telemetry for security monitoring, alerting, and incident investigations.

Although I had prior exposure to Active Directory and Windows Server concepts through coursework, this project was my first time building and managing a full domain in a home lab environment. I built a Windows domain, configured a workstation, enabled custom logging, simulated attacker behavior, and analyzed the telemetry in Splunk to refine my SOC skills.

2. Lab Architecture

- **Virtual Machines:**
  - **Windows Server 2022 (Domain Controller)**
    - Domain: cyberlab.local
    - Services: Active Directory, DNS, GPO
  - **Windows 10 Workstation**
    - Joined to cyberlab.local
    - Users: cyberlab\lspark | cyberlab\jmurphy
    - Telemetry tools: Sysmon, Splunk Universal Forwarder, Atomic Red Team
  - **Ubuntu (Splunk Server)**
    - Splunk Enterprise running on port 8000
    - Custom Index: endpoint
  - **Kali Linux (Attacker Machine)**
    - Used for RDP brute-force attempts
    - Tools: Hydra, Crowbar, Ncrack, xfreerdp,

Below is the network diagram I created to illustrate the lab architecture, including the system roles, configurations, and IP assignments used throughout the project:


Domain: cyberlab
Network: 192.168.10.0/24
Splunk Server: 192.168.10.10
Active Directory: 192.168.10.7
Attacker: 192.168.10.250
Active Directory
192.168.10.7
Splunk Universal Forwarder
Sysmon
Splunk Server
192.168.10.10
Kali Linux
192.168.10.250
Threat Activity
Internet
Target Machine - Windows 10
IP: DHCP
Splunk Universal Forwarder
Sysmon
Atomic Red Team

3. Objectives

- Build an enterprise-like Windows domain and endpoint setup.
- Configure advanced security logging using Sysmon.
- Forward logs to Splunk for SOC analysis.
- Generate real-world adversary telemetry using:
    - RDP brute-force activity
    - PowerShell suspicious behavior
    - Atomic Red Team invokes
- Troubleshoot SIEM ingestion issues and network/policy configurations.
- Perform detections and analysis in Splunk.

4. Key Technical Work & Learning Outcomes

4.1 Active Directory & Windows Domain Administration

Building on my previous experience with AD, I:

- Installed and configured **Windows Server** as a Domain Controller.
- Created the domain structure (cyberlab.local).
- Joined the Windows 10 machine to the domain.
- Managed users, passwords, and domain login policies.
- Learned how GPO settings affect authentication (e.g., NLA, security layers, RDP policies).

4.2 Endpoint Logging Setup

One of the core goals of this project was to build realistic endpoint telemetry similar to what real SOC analysts receive in an enterprise environment. To achieve this, I deployed and configured multiple logging components on the Windows 10 workstation.

Sysmon Deployment

I installed Sysmon v13 using Olaf Hartong's sysmon-modular configuration, which prioritizes high-value security telemetry while filtering out unnecessary noise. This setup enabled detailed logging for key Sysmon events such as:

- **1** – Process creation
- **3** – Network connections
- **11** – File creation

These logs are essential for threat hunting and detection engineering.

Splunk Universal Forwarder Setup

The Windows 10 machine acted as the **endpoint**, meaning it was the host generating events that would be forwarded to Splunk for analysis.

I created a custom index in Splunk named **endpoint**, then configured the forwarder to send:

- Security logs
- System logs
- Application logs
- Sysmon Operational logs

This is the configuration file used, which creates a full logging stack similar to what SOCs use for endpoint detection.

4.3 SIEM Troubleshooting Experience

During the initial setup, Splunk was not receiving *any* telemetry from the Windows 10 endpoint. This became one of the most educational parts of the project because it forced me to troubleshoot the entire ingestion path.

Issue Identified

Splunk showed **zero logs** in the “endpoint” index. After reviewing the Universal Forwarder configuration, I realized that during installation:

- I **did not select the Windows Event Log collection options**
- This meant none of the WinEventLog inputs were actually active on the endpoint
- Sysmon was configured correctly, but its events were not being forwarded

Because of this, the forwarder was running, but **nothing was being collected at all**.

Fix Implemented

To correct the issue, I manually added the missing Windows Event Log monitors using:

```
.\splunk.exe add monitor "C:\Windows\System32\Winevt\Logs\Security.evtx"

.\splunk.exe add monitor "C:\Windows\System32\Winevt\Logs\System.evtx"

.\splunk.exe add monitor "C:\Windows\System32\Winevt\Logs\Application.evtx"
```

Then I restarted the Universal Forwarder service:

```
net stop splunkforwarder

net start splunkforwarder
```

After the restart, telemetry immediately began flowing into Splunk.

Skills Learned

This process taught me:

- How Splunk inputs actually work
- How to diagnose missing telemetry at the collection layer
- How Splunk UF uses inputs.conf to collect logs
- How to manually configure log monitoring on Windows endpoints
- How to validate that the endpoint → forwarder → index pipeline is working

This experience was valuable because it mirrors real SOC engineering tasks where endpoints silently fail to send telemetry, and analysts must diagnose whether the issue is with the forwarder, the inputs, or the logs themselves.

5. Offensive Simulation / Telemetry Generation

5.1 RDP Brute-Force Attempts (Detailed Explanation)

To generate authentication-related telemetry for Splunk, I began by attempting RDP brute-force attacks against my Windows 10 lab machine. My initial plan was simple: use different brute-force tools, include the correct password in the wordlist, and observe both successful and failed login activity.

Tools Used

- **Crowbar**
- **Hydra**
- **Ncrack**
- **xfreerdp** (manual login validation)

What Happened During Testing

1. Starting with Crowbar

The first tool I tried was **Crowbar**, since it is specifically designed for RDP brute-forcing. However, Crowbar failed immediately. Even with the **correct password included**, Crowbar could not authenticate.

This made no sense at first—so I began troubleshooting.

2. Troubleshooting Attempt 1: NLA Configuration

The first thing I investigated was **Network Level Authentication (NLA)**.
NLA forces users to authenticate *before* an RDP session is created, which can break brute-force tools that expect to brute-force *after* the connection is established.

I checked and modified:

- NLA settings
- RDP security layer (forcing "RDP Security Layer" instead of "SSL")
- Encryption levels

Since none of these changes allowed Crowbar to authenticate, I expanded the troubleshooting to include Windows policies and OS-level restrictions that influence RDP login behavior. This included reviewing:

- Account lockout policy
- Failed login attempt timer
- NTLM authentication settings
- Whether the system expected domain-style usernames (e.g., cyberlab\user)

Even after adjusting these settings, Crowbar still failed to authenticate.
This made it clear that the issue was not caused by any single configuration, but rather by how the brute-force tools interacted with modern RDP security protocols.

5.2 Switching to Hydra and Ncrack

After exhausting everything I could do with Crowbar, I moved on to **Hydra** and **Ncrack**.

Result

- Both Hydra and Ncrack also failed to authenticate.
- Neither tool logged a successful login, **even with the correct password in the wordlist**.

At this point, it became clear that the issue was not the password, but how brute-force tools interact with **CredSSP, NTLM, and TLS handshakes** required by modern RDP.

However—this failure revealed something very important:

Even though Hydra and Ncrack failed to log in, they WERE generating Windows events.

This included:

- Event ID 4624 – Successful logon5379
- Event ID 5379 – Credential Manager credentials read4634
- Event ID 4634 – Logoff event

This was exactly the telemetry I needed for the Splunk analysis portion of my project.

Skills Gained

Through this testing and troubleshooting, I learned:

Authentication Protocol Knowledge

- How **NLA**, **NTLM**, **CredSSP**, and **TLS** work together during RDP authentication
- Why many brute-force tools fail against modern Windows systems

Troubleshooting Skills

- Diagnosing failures caused by username formatting
- Understanding how RDP negotiates encryption and security layers
- Recognizing policy-based restrictions like account lockout and timer delays

SOC-Relevant Skills

- How failed authentication attempts appear in Splunk
- How to correlate brute-force behavior with Windows event logs
- How to validate credentials safely using xfreerdp

Final Result

Even though **none** of the brute-force attempts actually succeeded, the process generated the telemetry I needed and gave me a deeper understanding of how Windows authentication works in real life.

The real value came from:

- The troubleshooting
- The log generation
- The insights into RDP security
- And the practice analyzing failed authentication attempts in Splunk

This made the exercise a success from a defensive and educational perspective.

5.2 Atomic Red Team Execution (From My Perspective)

After generating RDP authentication telemetry, I wanted to produce more diverse, attack-like events for Splunk—especially related to PowerShell, credential access, and process creation. To do this, I installed **Atomic Red Team** on my Kali Linux attacker machine and executed several atomic tests directly on the Windows endpoint.

I specifically targeted tests that were easy to run but designed to mimic realistic attacker techniques.

One of the first things I did was run PowerShell-based tests that used **Invoke-style commands**, such as downloading scripts or spawning hidden processes. Some of these commands required me to bypass PowerShell execution policies, which let me see how Windows logs suspicious PowerShell behavior when policies are overridden.

Running these tests immediately triggered multiple Windows event logs, including Sysmon data. On Splunk, I was able to observe:

- **Sysmon Event ID 1** every time an atomic test spawned a new process
- **Sysmon Event ID 3** when a test created an outbound network connection
- **Event ID 4104** showing script-block logging for the suspicious PowerShell commands I executed

Overall, Atomic Red Team gave me a controlled way to simulate adversarial activity and confirm that my Sysmon and Splunk configurations were capturing the right events. It also helped me practice identifying what malicious PowerShell usage looks like from a defender's point of view.