My Active Directory 2.0 Cybersecurity Project: From Detection to Automated Response

Project Goal

The primary objective of this project was to establish a fully functional **Active Directory (AD) environment** in the cloud, implement **Splunk for security monitoring and alerting** to detect unauthorized logins, and integrate **Shuffle as a Security Orchestration, Automation, and Response (SOAR) platform** to automate incident response workflows. This comprehensive approach allows me to demonstrate my capabilities across multiple cybersecurity domains.

Project Architecture

Throughout this project, I leveraged **Vultr Cloud Platform** to host my virtual machines (VMs), taking advantage of a $300 credit. The core components of my lab environment included:

- **JAZEB-ADDC01**: My **Domain Controller (Windows Server 2022)**, hosting Active Directory services for the `mydfir.local` domain.
- **JAZEB-SPLUNK**: My **Splunk Server (Ubuntu 22.04)**, responsible for ingesting and analyzing security telemetry.
- **Cloud instance**: My **Windows Test Machine (Windows Server 2022)**, joined to the `mydfir.local` domain to simulate an endpoint.
- An **Attacker Machine**: Representing external threats attempting to authenticate to my test machine.
- **Slack**: Integrated for **alert notifications**.
- **Shuffle**: My **SOAR platform**, used to build automated response playbooks.

My lab environment was logically mapped out using `draw.io` to visualize the flow of data and interactions, from telemetry collection to automated responses.

Key Steps and Accomplishments

Here's a detailed breakdown of the steps I took to build out this project:

Part 1: Diagramming the Environment

Before any build, I focused on **visualizing my lab environment** by creating a detailed diagram using `draw.io`. This crucial first step helped me map out the logical flow of my infrastructure and security components.

- I designed three primary virtual machines in Vultr: **JAZEB-ADDC01** (my Domain Controller running Windows Server and hosting the `mydfir.local` domain), **JAZEB-SPLUNK** (my Splunk server on Ubuntu, visually represented in dark green), and **Cloud instance** (my Windows test machine, in light blue, also running Windows Server and joined to `mydfir.local`).
- I included an **"Attacker Machine"** (red) to symbolize external attempts to authenticate to my Cloud instance, which was intentionally exposed to the public internet for this simulation.
- I conceptualized the **telemetry flow** (purple dashed lines) from my Windows machines (JAZEB-ADDC01 and Cloud instance) to JAZEB-SPLUNK.
- The diagram also laid out the integration with **Slack** (light purple) for alert notifications and **Shuffle** (light orange) as my SOAR platform, illustrating how Splunk alerts would trigger a playbook in Shuffle, sending notifications and orchestrating a response.
- A core part of the diagram was outlining the **successful unauthorized login playbook**: sending an email to the SOC analyst, prompting them to disable the user, and then automatically disabling the user in Active Directory if confirmed, followed by a Slack notification.

Part 2: Cloud Infrastructure Setup

In this phase, I deployed and configured my three virtual machines in the Vultr cloud and ensured their network connectivity.

- **VM Deployment**:
    - **JAZEB-ADDC01**: Deployed a Windows Standard 2022 server with 2 vCPUs, 4GB RAM, and 80GB disk space.
    - **Cloud instance**: Deployed another Windows Standard 2022 server with 1 vCPU, 2GB RAM, and 55GB disk space.
    - **JAZEB-SPLUNK**: Deployed an Ubuntu 22.04 server, choosing a more powerful configuration with 4 vCPUs, 8GB RAM, and 160GB disk space, as Splunk can be resource-intensive.
- **Firewall Configuration**: I created a Vultr firewall group (`my-dfir-ad-project-2.0`) to secure my VMs. Initially, I restricted SSH (Port 22) and RDP (Port 3389) access to **only my public IP address**, ensuring a secure initial setup.
- **Virtual Private Cloud (VPC) Network**: I enabled VPC for all three VMs, assigning them private IP addresses within the 10.22.96.x range (JAZEB-ADDC01: `10.22.96.4`, Cloud

instance: `10.22.96.3`, JAZEB-SPLUNK: `10.22.96.5`). This allowed for **secure internal communication** between my machines.

- **Network Troubleshooting**: I successfully resolved a common issue where Windows VMs initially used a 169.x.x.x IP address on their VPC interface. This was fixed by manually configuring the static private IP, subnet mask (`255.255.255.240`), and leaving the DNS/gateway blank on the Windows network adapter settings. This ensured **all my machines could communicate with each other internally via ping**.

Part 3: Active Directory Configuration

This part involved transforming one of my Windows servers into a fully functional Domain Controller and integrating my test machine into the newly created domain.

- **Domain Controller Setup**:
    - I RDP'd into **JAZEB-ADDC01** and installed the **Active Directory Domain Services** role.
    - Subsequently, I promoted **JAZEB-ADDC01** to a **new forest** with the domain name `mydfir.local`.
- **User Creation**: Within Active Directory Users and Computers, I created a new user account: **Jenny Smith** with the logon name `JSmith` and password `Winter2025!` (for testing purposes, with password set to never expire).
- **Domain Join**: I configured my **Cloud instance** to join the `mydfir.local` domain. A critical step here was troubleshooting a DNS resolution issue; the Cloud instance initially could not contact the domain. I resolved this by explicitly configuring the **Cloud instance's DNS server to point to JAZEB-ADDC01's private IP address (10.22.96.4)**.
- **Remote Desktop Access**: To enable remote authentication for my test user, I configured the Cloud instance to **allow remote connections** and added `JSmith` to the permitted users for RDP. This allowed me to successfully RDP into the Cloud instance using `mydfir\JSmith`.

Part 4: Splunk Configuration and Alerting

This was a pivotal phase where I set up my Splunk environment to collect security telemetry and create an alert for unauthorized RDP logins.

- **Splunk Server Installation**:
    - I SSH'd into my **JAZEB-SPLUNK** Ubuntu server.
    - Performed an `apt-get update` and `apt-get upgrade -y` to ensure the system was up-to-date.
    - Downloaded the **Splunk Enterprise `.deb` package** from Splunk's website.
    - Installed Splunk using `dpkg -i` and started the Splunk service from `/opt/Splunk/bin`, accepting the license and creating my `my-dfir` administrator account.
- **Web Interface Access**:

    - Initially, I couldn't access Splunk's web interface on **Port 8000**. I identified this as a firewall issue.
    - I configured both the **Vultr cloud firewall** to allow TCP connections on Port 8000 from **my IP address**.
    - Crucially, I also updated the **Ubuntu host firewall (UFW)** on JAZEB-SPLUNK with `ufw allow 8000`. This two-step firewall configuration immediately resolved the access issue.
- **Splunk Initial Setup**:
    - Logged into Splunk and changed the **time zone to GMT** for consistent data ingestion.
    - Installed the **"Splunk Add-on for Microsoft Windows"** application from the Splunk Apps section. This add-on is vital for **properly parsing Windows event logs**, ensuring field names like `user` are extracted correctly.
    - Created a **custom index** named `my-dfir-ad` under "Settings" -> "Indexes" to store Active Directory-related telemetry.
    - Configured a **new data receiving port on Port 9997** (Splunk's default forwarder port) under "Settings" -> "Forwarding and Receiving".
- **Windows Endpoint Telemetry Collection (Universal Forwarder)**:
    - Downloaded the **Splunk Universal Forwarder** (64-bit Windows) from the Splunk website.

    - Installed the Universal Forwarder on both my **Cloud instance** and **JAZEB-ADDC01**.

    - During installation, I specified **JAZEB-SPLUNK's private IP address (10.22.96.5) and Port 9997** as the receiving indexer.

Crucially, I manually configured the `inputs.conf` file (located in `C:\Program Files\SplunkUniversalForwarder\etc\system\local`) on both Windows machines. I added an entry to collect **Windows Security Event Logs (Event IDs)** and directed them to my `my-dfir-ad` index:

```
 [WinEventLog://Security]
index=my-dfir-ad
disabled=false
```

    -
    - Changed the **SplunkForwarder service** "Log On" setting to **"Local System account"** and restarted the service on both Windows machines.

    - **Troubleshooting Telemetry**: Initially, Splunk wasn't receiving data. I resolved this by adjusting **JAZEB-SPLUNK's Ubuntu host firewall (UFW)** to `ufw allow`

`9997` for incoming connections.

  - **Verification**: I confirmed successful telemetry reception in Splunk using the search `index=my-dfir-ad`, observing events from both `JAZEB-ADDC01` and `Cloud instance`.

- **Generating Test Data for Unauthorized Logins**:
  - To simulate an external attack, I temporarily loosened the **Vultr firewall's RDP rule** for my Cloud instance, allowing inbound connections on Port 3389 from **"anywhere" (`0.0.0.0/0`)**.
  - I then performed an **RDP login from a different external IP address** (via VPN) to my Cloud instance using Jenny Smith's account. This action generated a successful login event in Windows, specifically **Event ID 4624** with Logon Types 7 or 10, which are indicative of RDP connections.
- **Creating an Alert for Unauthorized Successful RDP Logins**:
  - I focused on **Event ID 4624 (successful logon)** and **Logon Types 7 or 10 (RDP)**.

I developed a **Splunk Search Processing Language (SPL) query** to identify these events:
index=my-dfir-ad event_code=4624 (Logon_Type=7 OR Logon_Type=10) Source_Network_Address=* Source_Network_Address!="-" Source_Network_Address!="40.*" | stats count by _time, computer, Source_Network_Address, user, Logon_Type

  - This query specifically filters for successful RDP logins, ensures a source IP exists, and excludes "authorized" IPs (in this demo, those starting with `40.*`) to identify potentially unauthorized access.
  - I saved this query as a scheduled alert titled **`my-dfir-unauthorized-successful-login-RDP`**. I configured it to run **every minute** (for testing), look at events from the "past 60 minutes", and trigger if the count of events was "greater than 0". The alert was set to add to "triggered alerts" with a "medium" severity.
  - I successfully verified that the alert triggered as expected after my simulated unauthorized login.

Part 5: Slack and Shuffle Automation

In the final part, I integrated Slack and Shuffle to automate the response to detected unauthorized logins, building a mini-playbook.

- **Shuffle Setup**: I created an account on Shuffle and initiated a new workflow named `my-dfir-ad-project-2.0`.
- **Splunk-Shuffle Webhook Integration**:

  - I configured my `my-dfir-unauthorized-successful-login-RDP` alert in Splunk to use a **webhook action**, pointing to the unique webhook URI provided by my Shuffle workflow.
  - Ensured the Shuffle workflow was "started" to actively listen for incoming alerts.
  - I verified that Splunk alerts were successfully captured by Shuffle, displaying details like `search_name`, `IP address`, `user`, and `Logon_Type`.
- **Slack Integration**:
  - I created a **Slack workspace (`my-dfir-projects`)** and a dedicated channel named `#alerts`.
  - I authenticated Shuffle with my Slack workspace using a one-click login.
  - I configured the Shuffle workflow to send **alert notifications to the `#alerts` channel**, including key details such as the alert name, timestamp, affected user, and source IP.
  - I confirmed that alerts from Splunk, processed by Shuffle, were successfully posted to my Slack channel.
- **Email Notification for User Action**:
  - I added a "User Input" node in Shuffle to represent the SOC analyst's decision point.
  - This node was configured to send an **email notification** to a designated SOC analyst email address, asking: **"Would you like to disable the user?"**.
- **Active Directory Integration for User Disablement**:
  - I integrated the **"Active Directory" application** within Shuffle.
  - **Crucially, I configured the authentication to my JAZEB-ADDC01** via LDAP (Port 389), specifying the server's private IP, domain (`mydfir.local`), and an administrative logon user (`Steven` – a test domain admin account I created to resolve initial authentication issues).
  - The workflow was designed to perform the **"disable user" action** on the `JSmith` account if the SOC analyst chose to disable it.
- **Status Check and Final Slack Notification**:
  - To ensure the playbook's reliability, I implemented a mechanism to **verify the user's disabled status** before sending a final notification.
  - After the "disable user" action, the playbook uses the "get user attributes" action from the Active Directory app to retrieve the `userAccountControl` property.
  - A **conditional branch** was added: if `userAccountControl` **"contains account disabled"**, then a final Slack message is sent to the `#alerts` channel stating: **"Account: [user] has been disabled"**. This ensures the SOC analyst receives confirmation only after the action is truly completed.
  - I rigorously tested the workflow, ensuring that the account was disabled in Active Directory and the final Slack notification was sent only when the analyst confirmed the action. I also tested the "no" path, confirming no action was taken.

Conclusion

This **Active Directory 2.0 Cybersecurity Project** has provided me with invaluable hands-on experience in building, securing, monitoring, and automating a critical enterprise environment. From designing the network architecture and configuring core services like Active Directory and Splunk, to developing advanced Splunk queries and orchestrating automated responses with Shuffle, I have gained a deep understanding of the practical aspects of security operations.

This project specifically highlights my capabilities in:

- **Cloud Infrastructure Management** (Vultr)
- **Windows Server Administration** (Active Directory, DNS)
- **Linux Server Management** (Ubuntu)
- **Security Information and Event Management (SIEM)** with Splunk (installation, configuration, data ingestion, **alert creation for unauthorized RDP logins**)
- **Security Orchestration, Automation, and Response (SOAR)** with Shuffle (workflow development, integration with Splunk, Slack, and Active Directory)
- **Network Firewall Configuration** (Vultr and host-based UFW)
- **Troubleshooting and Problem Solving** in complex environments

By showcasing this project, I demonstrate my readiness to contribute effectively as a **SOC Analyst**, leveraging these skills to detect, analyze, and respond to cyber threats. I am confident that this comprehensive portfolio piece reflects my curiosity, dedication to continuous learning, and ability to "do things differently".