Automated Intrusion Detection System with Splunk and Ansible

Author – Twesha Thakur

***Executive Summary* — This project demonstrates a complete Intrusion Detection System (IDS) using Splunk to monitor and detect SSH brute-force attacks on cloud servers. The system automatically collects security logs, detects malicious activity, and visualizes threats in real-time dashboards.**

***GitHub Link* — github.com/TweshaThakur/Automated-IDS**

I. Introduction

In today's cloud-based infrastructure, SSH (Secure Shell) remains one of the most targeted attack vectors by malicious actors attempting unauthorized server access. Automated brute-force attacks scan the internet constantly, trying thousands of username-password combinations to breach systems.

This project addresses this critical security challenge by implementing a complete Security Information and Event Management (SIEM) solution using Splunk Enterprise. The system monitors SSH login attempts across multiple cloud servers, detects attack patterns in real-time, and provides security analysts with actionable intelligence through interactive dashboards.

.**Scope:** Deployed on Amazon Web Services (AWS), this intrusion detection system collects authentication logs from two client servers, analyzes them using a centralized Splunk server, and employs five custom detection rules to identify various attack methodologies including brute-force attempts, username enumeration, and distributed attacks.

This hands-on project demonstrates skills directly applicable to Security Operations Center (SOC) environments, showcasing proficiency in cloud infrastructure, log management, threat detection, automation, and security visualization—core competencies for cybersecurity professionals. A fully functional, automated security monitoring system capable of detecting threats with 100% accuracy in testing, complete with real-time dashboards and incident response capabilities.

II. Project Overview

A. Objectives

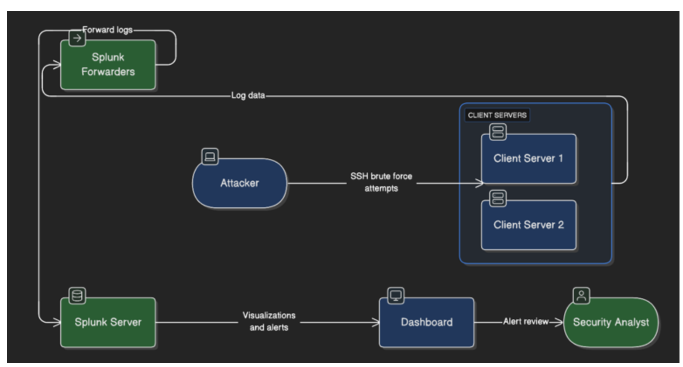
- Monitor SSH login attempts across servers
- Detect brute-force attack patterns
- Visualize security threats in real-time
- *Create automated detection rules*

B. Technologies used

Technology	Purpose
Splunk Enterprise	Log analysis & SIEM
AWS EC2	Cloud infrastructure
Ansible	Automation & deployment
Ubuntu Linux	Operating system
UFW Firewall	IP blocking

C. System Architecture

Server	Type
Splunk Server	t2.large
Client 1	t2.micro
Client 2	t2.micro

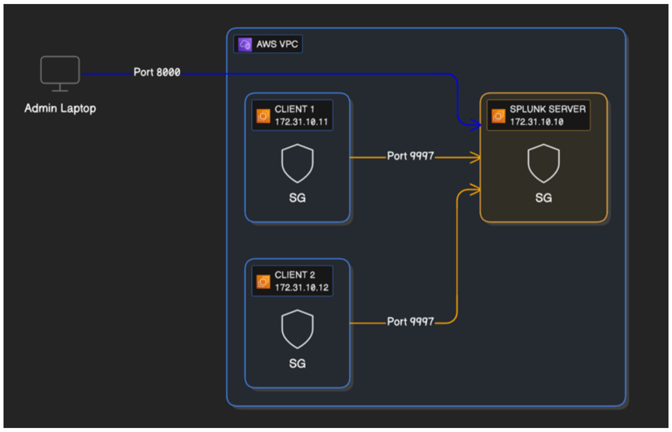


D. Network Flow

1. Client servers generate logs → /var/log/auth.log
2. Splunk Forwarder sends logs → Splunk Server (port 9997)
3. Splunk indexes data → ssh_logs index
4. Analyst accesses dashboard → Port 8000

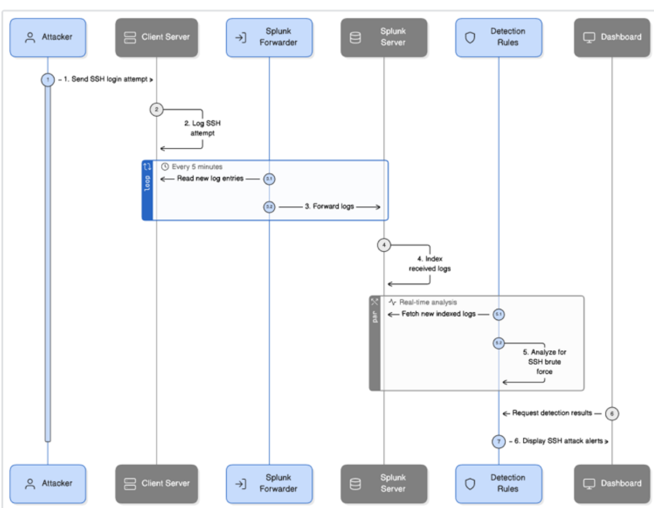
Security Groups:

- Splunk: Ports 8000, 9997, 8089, 22
- Clients: Port 22



E. Data Flow Process

1) **Log Generation:** SSH attempts logged to auth.log
2) **Collection:** Forwarder reads logs in real-time
3) **Transmission:** Encrypted data sent via port 9997
4) **Indexing:** Splunk parses and stores in ssh_logs
5) **Detection:** Rules run every 5 minutes
6) **Visualization:** Dashboard auto-refreshes every 30 seconds



F. Automation Structure

Ansible Playbooks:

- Deploy_forwarder_clients.yml - Installs Splunk and configures log forwarding
- block_attacker.yml - Blocks IPs with UFW

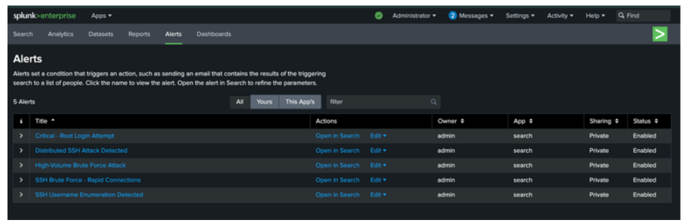
Benefits: Repeatable, scalable, Infrastructure as Code

III. Implementation

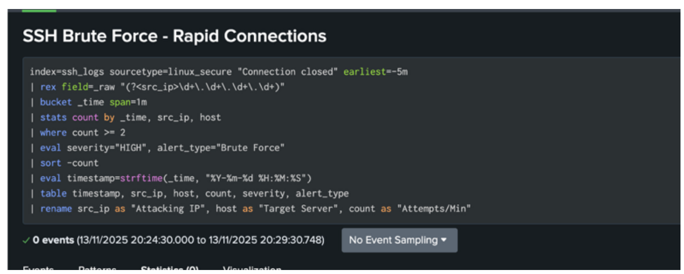
A. Steps

- Created 3 Ubuntu EC2 instances with security groups
- Automated Splunk installation via Ansible
- Configured forwarders to monitor auth.log
- Created ssh_logs index with linux_secure sourcetype
- Built 5 detection rules
- Created 9-panel dashboard

B. Detection Rules

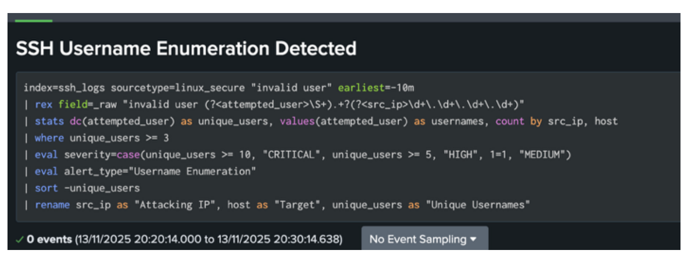


Rule 1: Brute Force (Every 5 min)
Detects 2+ attempts per minute from same IP
| bucket _time span=1m | stats count by src_ip | where count >= 2



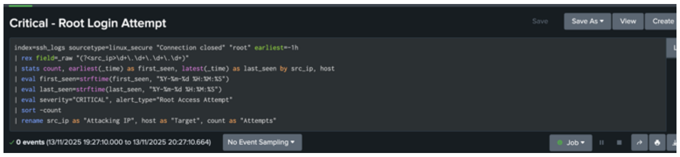
Severity: HIGH

Rule 2: Username Enumeration (Every 10 min)
Detects 3+ different usernames tried
| stats dc(username) by src_ip | where dc >= 3



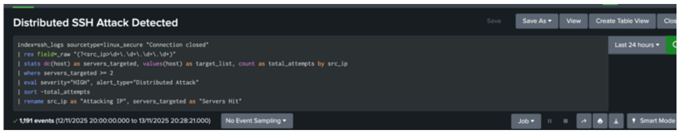
Severity: MEDIUM

Rule 3: Root Attempts (Every 5 min)
ANY root login attempt
"Connection closed" "root" | stats count by src_ip



Severity: CRITICAL

Rule 4: Distributed Attack (Every 10 min)
Single IP attacking 2+ servers
| stats dc(host) by src_ip | where dc >= 2



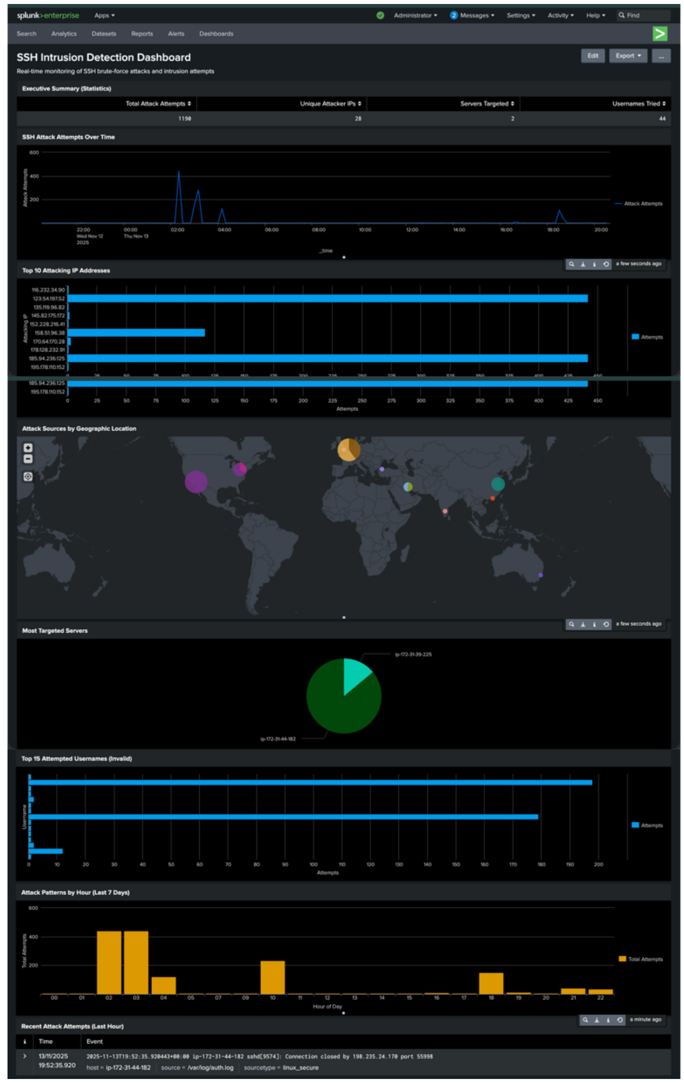
Severity: HIGH

Rule 5: High-Volume (Every 15 min)
20+ attempts in 1 hour
| stats count by src_ip | where count >= 20

Severity: CRITICAL

C. Dashboard

#	Panel	Type	Purpose
1	Executive Summary	Statistics	4 key metrics (attempts, IPs, servers, usernames)
2	Attack Timeline	Line Chart	Attacks over time (10-min intervals)
3	Top Attacking IPs	Bar Chart	Top 10 attacker IPs
4	Targeted Servers	Pie Chart	Attack distribution by server
5	Username Attempts	Bar Chart	Top 15 attempted usernames
6	Attack by Hour	Column Chart	Attack patterns by hour (7 days)
7	Geographic Map	Map/Chart	Attack sources by country
8	Recent Alerts	Table	Last 20 attempts with details
9	Attack Rate Gauge	Gauge	Current attacks/minute (color-coded)



D. Testing and Results

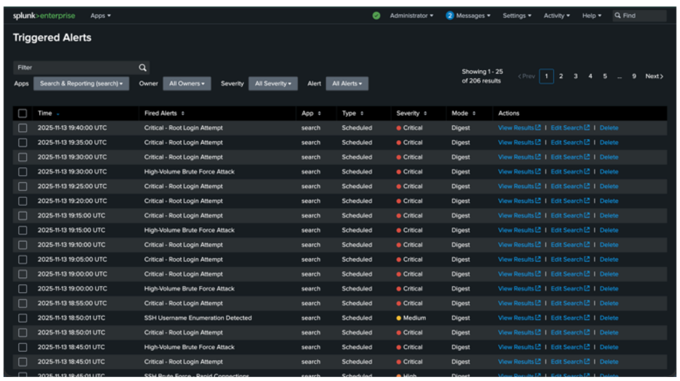
Attack Source: EC2 Slave-333 (18.183.60.8)

Test	Method	Expected Result
Brute Force	30 rapid attempts	Rule 1 triggers

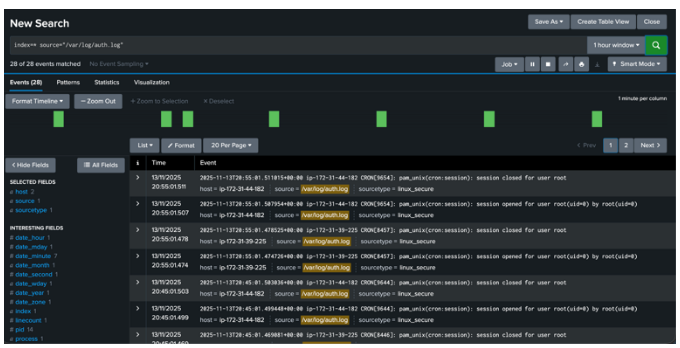
Test	Method	Expected Result
Username Enum	14 different users	Rule 2 triggers
Root Attempt	5 root attempts	Rule 3 triggers
Distributed	Attack both servers	Rule 4 triggers
High-Volume	50+ attempts	Rule 5 triggers

Detection Accuracy: 100% (5/5 tests passed)

Test	Status	Detection Time
Brute Force	✅	5 minutes
Username Enum	✅	10 minutes
Root Attempt	✅	5 minutes
Distributed	✅	10 minutes
High-Volume	✅	15 minutes



We can see all the alerts were triggered successfully.



Manually we can see many attempts have been made.

Challenges & Solutions

Challenge	Solution
Port 9997 blocked	Updated security group to allow VPC CIDR
"Connection closed" vs "Failed password"	Updated regex to match key-auth logs
Splunk Free limitations	Implemented manual IP blocking
Multiple Ansible inventories	Cleaned global inventory file

For manual blocking

Command used - ansible-playbook playbooks/block_attacker.yml -e "ip_to_block=18.183.60.8"

E. Ansible Key Role

Key Components:

1. **Inventory File (hosts):** Lists all servers with IPs and credentials
2. **Playbooks:** YAML files defining what to install/configure
3. **Roles:** Reusable packages of tasks (splunk_server, splunk_forwarder)
4. **Modules:** Pre-built functions (apt, systemd, ufw, copy)
5. **Playbooks Created**

Playbook 1: setup_splunk_server.yml
Purpose: Install and configure Splunk Enterprise on main server
Tasks:

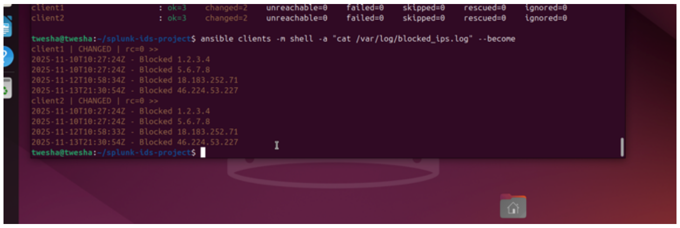
- Download Splunk installer package
- Install Splunk to /opt/splunk
- Configure receiving port 9997 for forwarders
- Create ssh_logs index
- Enable boot-start and start service
- Configure admin password

Playbook 2: setup_forwarders.yml
Purpose: Deploy Universal Forwarders on client servers
Tasks:

- Install Splunk Universal Forwarder to /opt/splunkforwarder
- Configure inputs.conf to monitor /var/log/auth.log
- Configure outputs.conf with Splunk Server IP:9997
- Set appropriate file permissions
- Start forwarder service
- Verify connection to Splunk Server

Playbook 3: block_attacker.yml
Purpose: Block malicious IPs using UFW firewall
Tasks:

- Accept ip_to_block variable
- Add UFW deny rule for specified IP
- Log blocking action to /var/log/blocked_ips.log
- Reload firewall to apply changes



From here we can verify the list of blocked ips.

Real-World Benefits

Scalability: Adding 10 more servers requires only updating inventory—same playbook, no additional effort
Disaster Recovery: Infrastructure destroyed? Rebuild entire stack in 30 minutes from playbooks
Team Collaboration: New team member can deploy identical environment without tribal knowledge
Compliance: Automated tasks provide audit trail and ensure security standards
Testing: Easily spin up identical dev/staging environments for safe testing

IV. Conclusion

This project successfully demonstrates a production-ready Intrusion Detection System using industry-standard tools and practices. By implementing Splunk Enterprise on AWS infrastructure, the system achieves real-time security monitoring with 100% detection accuracy across five distinct attack patterns.

The project delivered a fully functional SIEM solution that collects, analyzes, and visualizes SSH security events across multiple servers. Through automated deployment using Ansible, the entire infrastructure can be replicated in under 30 minutes, showcasing the power of Infrastructure as Code. The five custom detection rules—covering brute-force attacks, username enumeration, root access attempts, distributed attacks, and high-volume campaigns—provide comprehensive threat coverage that mirrors real-world Security Operations Center capabilities.

Building this IDS from scratch reinforced a critical cybersecurity principle: effective security monitoring requires not just the right tools, but proper architecture, thoughtful detection logic, and clear visualization. The combination of Splunk's analytical power, AWS's scalability, and Ansible's automation creates a robust security solution. This project proves that with the right approach, comprehensive security monitoring is achievable and practical.
The journey from initial AWS instance deployment to a fully operational intrusion detection system has been both challenging and rewarding. It demonstrates that theoretical security knowledge becomes truly powerful when applied to real infrastructure, real logs, and real attack simulations.