ABSTRACT

Modern enterprise networks face increasingly sophisticated cyber threats that target both external and internal systems. Traditional security measures, while essential, are often insufficient to prepare teams for real-world attack scenarios. This project simulates a complete enterprise security environment by integrating both offensive (Red Team) and defensive (Blue Team) operations within a controlled lab setup.

The Red Team component focuses on reconnaissance, vulnerability exploitation, privilege escalation, and persistence using widely recognized penetration testing tools such as Nmap, SQL map, Metasploit, and Burp Suite. The Blue Team component implements advanced defense mechanisms, including firewall configurations, intrusion detection with Snort, centralized log analysis using the ELK stack, network monitoring with Nagios, VPN access control, DNS filtering, and email security measures (SPF, DKIM, DMARC).

By combining attack and defense strategies, this simulation enhances the understanding of real-world cyber operations, strengthens detection and response capabilities, and improves overall network resilience. The project also emphasizes industry-relevant tools and configurations, preparing participants for job-ready roles in cybersecurity operations centers (SOCs) and penetration testing teams.

Topics **Page No.**

LIST OF ABBREVIATIONS

Sr. No.	Abbreviation	Full Form
1.	IDS	Intrusion Detection System
2.	IPS	Intrusion Prevention System
3.	SOC	Security Operations Center
4.	VPN	Virtual Private Network
5.	TLS	Transport Layer Security
6.	SPF	Sender Policy Framework
7.	DKIM	DomainKeys Identified Mail
8.	DMARC	Domain-based Message Authentication, Reporting & Conformance
9.	ELK	Elasticsearch, Logstash, Kibana
10.	pfSense	Open-source Firewall/Router Platform
11.	DNS	Domain Name System
12.	OSINT	Open-Source Intelligence
13.	SQLi	SQL Injection
14.	XSS	Cross-Site Scripting

LIST OF TABLES

1. INTRODUCTION

As organizations increasingly rely on interconnected systems and online services, the threat landscape facing enterprise networks has grown more complex and sophisticated. Cybercriminals employ advanced techniques to exploit vulnerabilities, compromise data, and disrupt operations. To counter these threats, it is essential for security teams to not only deploy robust defensive mechanisms but also to understand the strategies and tools used by attackers.

The concept of Red and Blue Team operations has emerged as a powerful approach to improving cybersecurity readiness. The Red Team simulates real-world adversaries by conducting reconnaissance, exploiting vulnerabilities, and attempting to gain unauthorized access to systems. The Blue Team focuses on defense — detecting, analyzing, and responding to these attacks while implementing preventive measures to strengthen network security.

This project presents a complete enterprise security simulation lab that integrates both Red Team (offensive) and Blue Team (defensive) operations within a controlled environment. The Red Team leverages penetration testing tools such as Nmap, Burp Suite, SQL map, and Metasploit to simulate attacks, while the Blue Team uses pfSense, Snort, the ELK stack, Nagios, VPNs, DNS filtering, and email security protocols to monitor, detect, and respond to threats.

By combining these two perspectives, the project not only demonstrates practical attack and defense techniques but also fosters a deeper understanding of the full cyber kill chain. The simulation environment closely mirrors real-world enterprise setups, preparing cybersecurity professionals for incident response, penetration testing, and security operations center (SOC) roles.

1.1 Problem Statement

Modern enterprises face a constant barrage of cyber threats that range from opportunistic attacks to highly targeted campaigns. While many organizations deploy security tools such as firewalls, intrusion detection systems, and endpoint protection, these measures often remain untested against real-world attack scenarios. This gap between theoretical security and practical resilience can lead to undetected vulnerabilities, delayed incident response, and significant data breaches.

The primary challenge is to create a realistic, controlled environment where both offensive and defensive cybersecurity skills can be tested and enhanced. Red Team exercises help identify weaknesses in infrastructure, configurations, and processes, while Blue Team operations strengthen monitoring, detection, and response capabilities. Without such simulations, security teams may lack the experience needed to respond effectively to evolving threats.

This project addresses the need for a comprehensive security simulation lab that integrates Red and Blue Team operations in a single framework. By simulating real-world attack chains and deploying defensive countermeasures, the environment enables participants to practice threat detection, incident response, and system hardening — thereby bridging the gap between classroom learning and operational readiness.

2. LITERATURE SURVEY

The concept of Red and Blue Teaming originates from military training exercises, where opposing forces (Red) simulate attacks to test the defenses of the friendly forces (Blue). In the context of cybersecurity, this approach was formalized in the late 1990s and early 2000s as organizations began to recognize the need for proactive security testing.

Red Teaming focuses on simulating real-world adversaries using tactics, techniques, and procedures (TTPs) similar to those employed by cybercriminals and advanced persistent threats (APTs). It includes reconnaissance, vulnerability scanning, exploitation, privilege escalation, persistence, and data exfiltration. Commonly used tools include Nmap, Metasploit, Burp Suite, SQLmap, and privilege escalation scripts like LinPEAS.

Blue Teaming, in contrast, is dedicated to defending against these attacks. It involves real-time monitoring, intrusion detection, log analysis, incident response, and system hardening. Popular defensive tools include Snort for intrusion detection, pfSense for firewall and DMZ configuration, ELK Stack for centralized log management, and Nagios for network and service monitoring. Security hardening techniques often include TLS encryption, VPN gateways, DNS filtering, and email authentication protocols such as SPF, DKIM, and DMARC.

Several studies and training frameworks, such as the MITRE ATT&CK framework and NIST Special Publication 800-115, emphasize the importance of combining offensive and defensive simulations to create a comprehensive security strategy. Red vs. Blue Team simulations provide a practical platform for security analysts to understand the full cyber kill chain, from initial compromise to post-exploitation, while simultaneously developing effective detection and mitigation strategies.

A parameter-based comparison between Red and Blue Team roles is shown in Table 1.

Table 1: Comparison of Red Team and Blue Team Roles

Parameter	**Red Team**	**Blue Team**
Objective	Simulate attacks to find weaknesses	Detect, prevent, and respond to attacks
Approach	Offensive	Defensive
Tools Used	Nmap, Metasploit, Burp Suite, SQLmap, LinPEAS	Snort, pfSense, ELK Stack, Nagios, Pi-hole

Parameter	Red Team	Blue Team
Skills Required	Exploitation, privilege escalation, OSINT	Threat detection, log analysis, incident response
Outcome	Identify vulnerabilities and misconfigurations	Strengthen defenses and incident handling

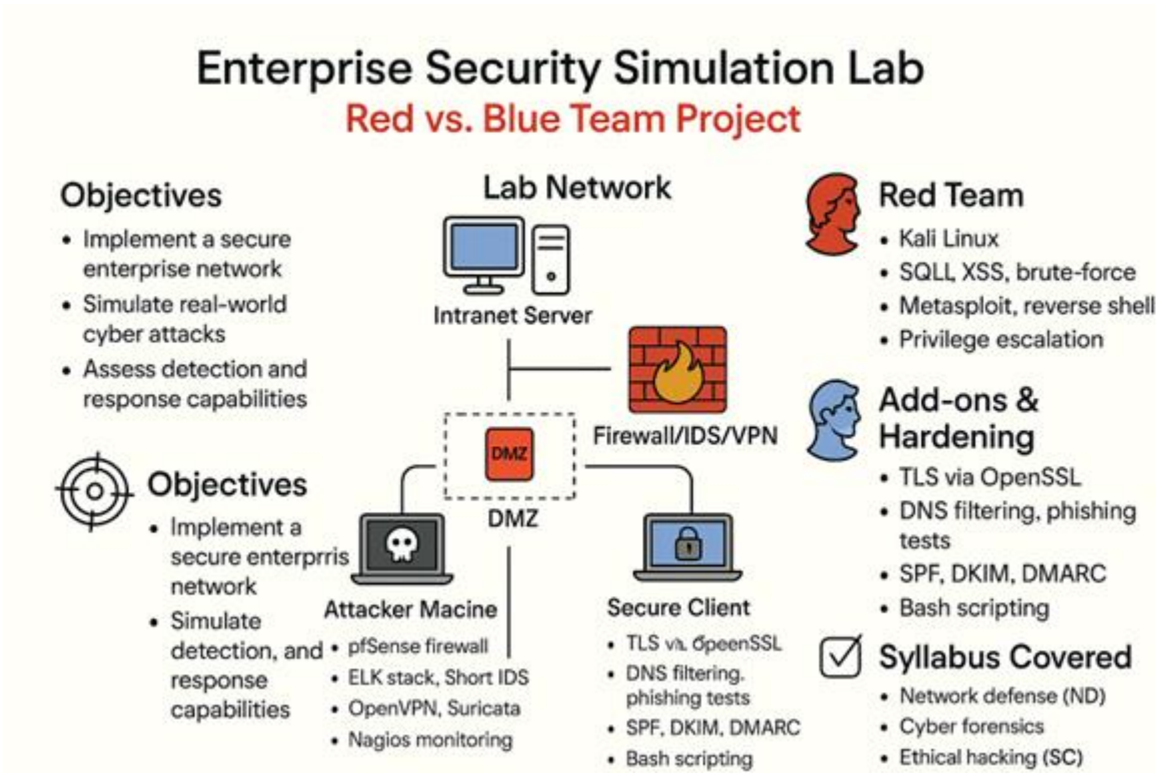
3. METHODOLOGY

This project implements a complete enterprise security simulation lab, combining Red Team (offensive) and Blue Team (defensive) operations in a controlled network environment. The methodology follows a structured approach to design, configure, and execute both attack and defense workflows.

The environment is divided into three primary zones:

1. **Attacker Zone (Red Team)** – Simulates malicious actors using penetration testing tools to exploit vulnerabilities in the target environment.
2. **Target Zone (Enterprise Services)** – Hosts web applications, databases, and network services configured with intentional vulnerabilities.
3. **Defensive Zone (Blue Team)** – Includes firewall, intrusion detection, log management, monitoring, and security hardening components to detect and mitigate attacks.

3.1 System Architecture



The system architecture consists of the following components:

- **Attacker Machine (Kali Linux)**
  - Performs reconnaissance, vulnerability scanning, exploitation, and persistence using tools such as Nmap, SQLmap, Burp Suite, Metasploit, and LinPEAS.
- **Target Server (Debian + DVWA)**
  - Runs the Damn Vulnerable Web Application (DVWA) with Apache and MySQL, secured with TLS to simulate real-world HTTPS environments.
- **Firewall and Intrusion Detection (pfSense + Snort)**
  - pfSense is configured to manage traffic between the attacker, target, and internet gateway, with a DMZ setup.
  - Snort is deployed for intrusion detection, with custom rules to identify SQL injection (SQLi), cross-site scripting (XSS), and brute force attempts.
- **Log Management (ELK Stack)**
  - Logs from Snort, the target server, and firewall are forwarded via Filebeat to Logstash, stored in Elasticsearch, and visualized in Kibana dashboards.
- **Monitoring (Nagios)**
  - Tracks system uptime, network traffic, and service availability; issues alerts upon detecting anomalies.
- **VPN Gateway (OpenVPN)**
  - Provides secure remote access for simulated remote employees and administrators.
- **DNS Filtering (Pi-hole)**
  - Blocks access to known malicious domains, preventing phishing and malware command-and-control communications.
- **Email Security**
  - Implements SPF, DKIM, and DMARC to protect against email spoofing and phishing.

This architecture ensures that every stage of the cyber kill chain is represented — from initial reconnaissance to final incident response — allowing participants to practice comprehensive attack and defense strategies in a realistic enterprise-like environment.

4. REQUIREMENT SPECIFICATION

This project has been implemented entirely in a virtualized environment using Oracle VirtualBox. The network is segmented into **WAN**, **LAN**, and **DMZ** zones, with dedicated machines assigned to Red Team and Blue Team roles. The specification below details the hardware and software requirements necessary to replicate the Enterprise Security Simulation Lab.

4.1 Software & Hardware Requirements

A. Software Requirements

1. **Virtualization Platform**
   - Oracle VirtualBox (latest stable version)
   - Host OS: Windows / Linux (any capable of running VirtualBox)
2. **Security Infrastructure**
   - **pfSense Firewall** (latest stable release)
     - Configured with **three network adapters**:
       - **WAN**: Connected to bridged adapter (DHCP from external network)
       - **LAN**: Internal network **192.168.30.0/24**, DHCP range **192.168.30.100 – 192.168.30.150**
       - **DMZ**: Internal network **192.168.20.0/24**, DHCP range **192.168.20.100 – 192.168.20.150**
     - Includes rule-based segmentation between WAN, LAN, and DMZ
     - Acts as gateway, firewall, and traffic filter between zones
3. **Red Team Environment**
   - **Kali Linux** machine connected to **WAN** network
   - Tools installed: Nmap, Burp Suite, SQLmap, Metasploit Framework, LinPEAS, Nikto, DNSenum, Netcat, GTFOBins scripts
4. **Blue Team Environment**
   - **Debian Linux** (connected to LAN)
     - Configured for defensive monitoring, intrusion detection, and incident response
     - Tools: Snort, ELK Stack (Filebeat, Logstash, Elasticsearch, Kibana), Nagios Core, OpenVPN, Pi-hole, Email Security (SPF, DKIM, DMARC)
   - **Windows Machine** (connected to LAN)

        - Used for accessing web-based management interfaces of pfSense, Kibana, Nagios, Pi-hole, and VPN dashboard

5. **Target Server (DMZ)**
    - Debian machine running **Damn Vulnerable Web Application (DVWA)** on HTTPS (Port 443)
    - Apache, MySQL configured for DVWA operation
    - Secured with XCA self-signed TLS certificate

B. Hardware Requirements

Component	Specification per VM	Total Setup
CPU	Dual-Core (2.4 GHz)	Host machine minimum Quad-Core (3.0 GHz recommended)
RAM	4 GB	Host machine ≥ 16 GB
Storage	50 GB	Host machine ≥ 500 GB SSD recommended
Network Adapters	3 (pfSense), 1 (other VMs)	VirtualBox Internal & Bridged networks

---

Networking Summary:

- **WAN**: External network access for attacker (Kali Linux)
- **LAN**: Internal corporate network for Blue Team (Defensive Debian + Windows UI)
- **DMZ**: Exposed service network hosting DVWA target server
- **pfSense**: Controls inter-zone communication, DHCP, and firewall rules

This configuration provides a realistic enterprise-like segmentation, enabling simulation of attacks from an external threat actor into a DMZ service, lateral movement attempts, and defensive monitoring from the LAN zone.

5. WORKING

The Enterprise Security Simulation Lab operates as a segmented network, enabling controlled Red Team attack simulations and Blue Team defensive monitoring. The interaction between the WAN, LAN, and DMZ zones replicates real-world enterprise network structures, allowing realistic offensive and defensive scenarios.

Operational Flow

1. **Attack Simulation (Red Team – WAN Zone)**
   - The Kali Linux attacker machine connects via the WAN interface to the pfSense firewall.
   - Reconnaissance activities are performed using tools such as **Nmap** and **DNSenum** to map active hosts and open ports in the DMZ.
   - Exploitation attempts target the DVWA application in the DMZ using **SQLmap**, **Burp Suite**, and **Metasploit**, simulating common web application vulnerabilities such as SQL Injection and Cross-Site Scripting (XSS).
   - Post-exploitation activities (e.g., **LinPEAS** privilege escalation, **Netcat** reverse shell) simulate persistence and lateral movement attempts toward the LAN.
2. **Target Server Exposure (DMZ Zone)**
   - The Debian server in the DMZ runs DVWA over HTTPS (port 443) with a self-signed TLS certificate.
   - The server is intentionally vulnerable to simulate real-world exploitable conditions.
   - pfSense firewall rules allow WAN-to-DMZ traffic on HTTPS, enabling controlled attack surface exposure.
3. **Defensive Monitoring (Blue Team – LAN Zone)**
   - The Debian Blue Team machine monitors network and server activity using **Snort** IDS rules configured to detect SQLi, XSS, and brute force attempts.
   - **ELK Stack** aggregates and visualizes logs from Snort, pfSense, and the target server, enabling real-time attack pattern analysis.
   - **Nagios Core** continuously checks host and service availability, triggering alerts upon detecting service downtime or performance degradation.
4. **Additional Security Controls**
   - **Pi-hole DNS filtering** blocks malicious domains to prevent phishing and malware callbacks.
   - **OpenVPN** provides secure remote administrative access for Blue Team analysts.

6. IMPLEMENTATION

This section outlines the step-by-step process used to configure the lab, execute attack simulations, monitor defensive responses, and implement hardening measures. The implementation is divided into three parts: Red Team (Offensive), Blue Team (Defensive), and Security Hardening.

- o **Email security measures** (SPF, DKIM, DMARC) protect against email spoofing in simulated phishing attempts.

5. **Incident Response Workflow**
   - o When suspicious activity is detected, alerts are sent via email to the Blue Team.
   - o Analysts investigate using Kibana dashboards, correlating Snort alerts, firewall logs, and server logs.
   - o Appropriate remediation steps (e.g., blocking an IP, disabling vulnerable services) are taken to contain and mitigate the threat.

This operational model ensures that every stage of the cyber kill chain—from reconnaissance to exfiltration—is tested against a live defensive posture, bridging the gap between theoretical cybersecurity knowledge and hands-on operational readiness.

IMPLEMENTATION

This section outlines the step-by-step process used to configure the lab, execute attack simulations, monitor defensive responses, and implement hardening measures. The implementation is divided into three parts: **Red Team (Offensive)**, **Blue Team (Defensive)**, and **Security Hardening**.

6.1 Red Team Implementation (Kali Linux – WAN)

Step 1: Reconnaissance

- Perform network scanning from the WAN to identify live hosts and open ports in the DMZ

```
aditya@Blue-Team:~$ nmap -sV 192.168.20.0/24
Starting Nmap 7.93 ( https://nmap.org ) at 2025-08-11 14:12 IST
Nmap scan report for 192.168.20.1
Host is up (0.0055s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 9.4 (protocol 2.0)
53/tcp  open  domain   Unbound
80/tcp  open  http     nginx
443/tcp open  ssl/http nginx

Nmap scan report for 192.168.20.100
Host is up (0.038s latency).
Not shown: 994 closed tcp ports (conn-refused)
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 9.2p1 Debian 2+deb12u6 (protocol 2.0)
53/tcp   open  domain   ISC BIND 9.18.33-1~deb12u2 (Debian Linux)
80/tcp   open  http     Apache httpd 2.4.62
111/tcp  open  rpcbind  2-4 (RPC #100000)
443/tcp  open  ssl/http Apache httpd 2.4.62
5666/tcp open  ssl/nrpe?
Service Info: Hosts: 192.168.20.100, www.dvwa.local; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 256 IP addresses (2 hosts up) scanned in 21.07 seconds
```

6.2 Blue Team Implementation

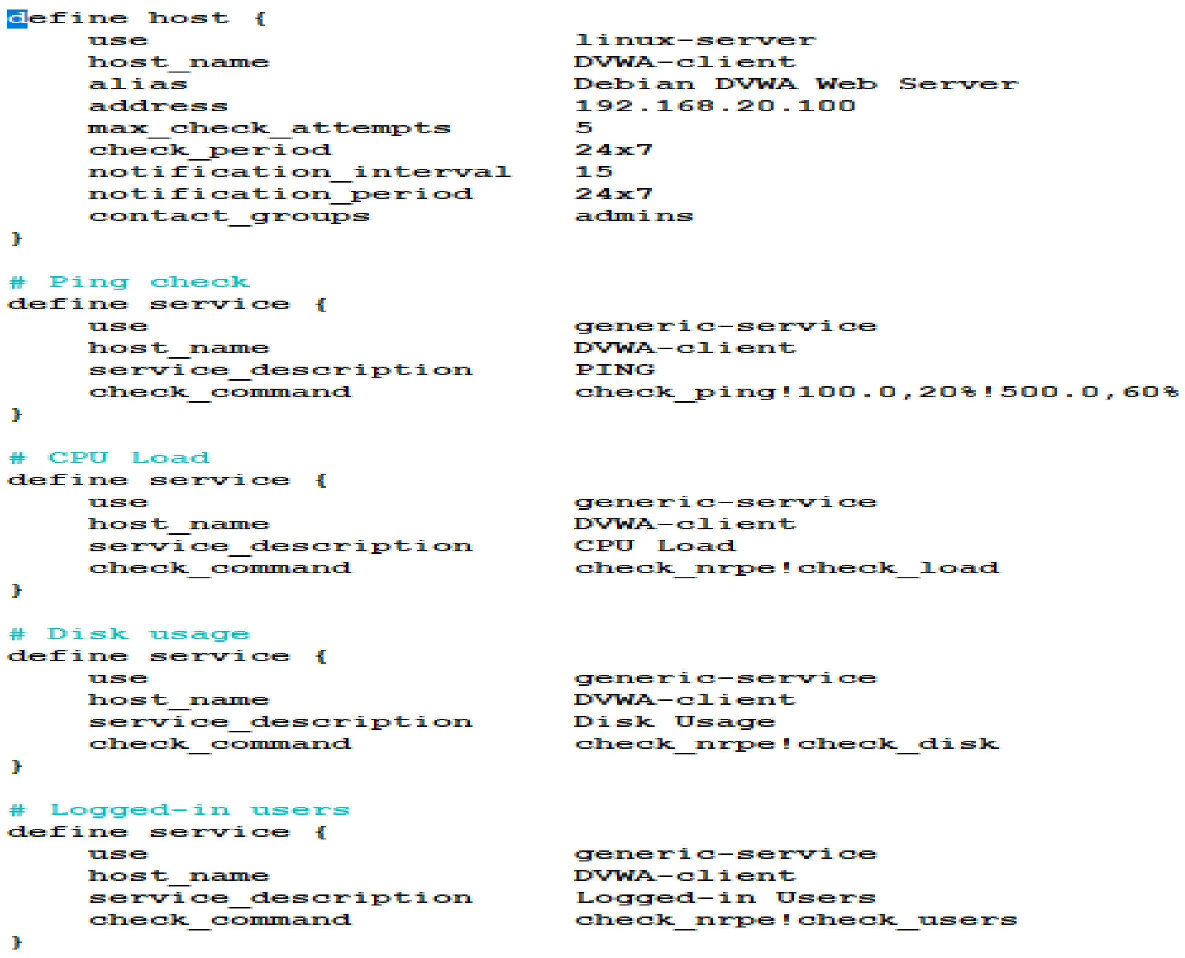
The Blue Team in this simulation is responsible for detecting, analyzing, and responding to Red Team activities. Defensive operations were deployed on a Debian Linux machine (LAN zone) and monitored via a Windows machine for UI access.

Step 1: Install Nagios Core

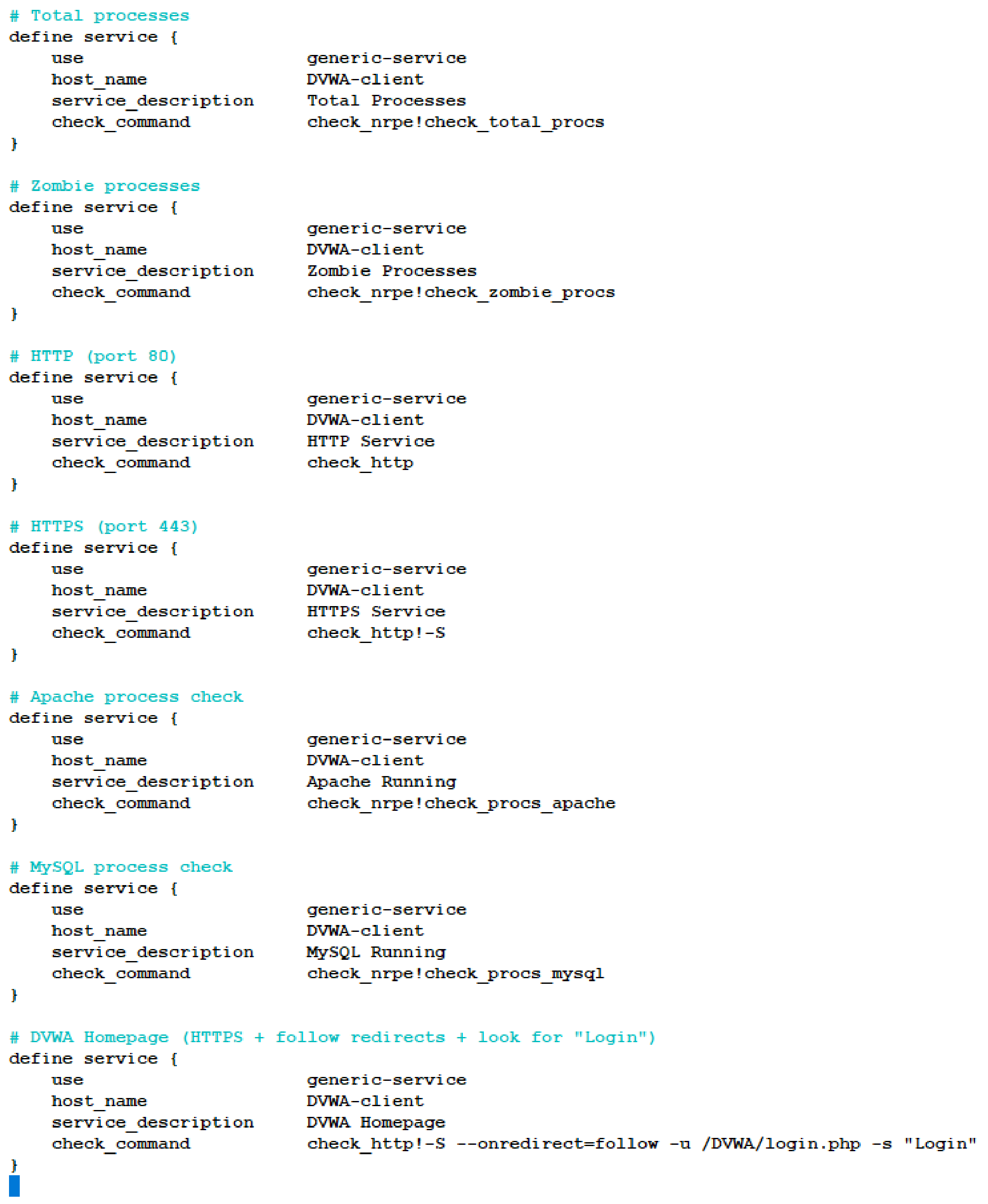
```
sudo apt update
sudo apt install nagios4 nagios-plugins nagios-nrpe-plugin
```

Step 2: Configure Hosts to be Monitored

- **Added entries for DVWA Server, pfSense, and Windows UI Machine in /etc/nagios4/conf.d/hosts.cfg:**



```
define host {
        use                     linux-server
        host_name               DVWA-client
        alias                   Debian DVWA Web Server
        address                 192.168.20.100
        max_check_attempts      5
        check_period            24x7
        notification_interval   15
        notification_period     24x7
        contact_groups          admins
}

# Ping check
define service {
        use                     generic-service
        host_name               DVWA-client
        service_description     PING
        check_command           check_ping!100.0,20%!500.0,60%
}

# CPU Load
define service {
        use                     generic-service
        host_name               DVWA-client
        service_description     CPU Load
        check_command           check_nrpe!check_load
}

# Disk usage
define service {
        use                     generic-service
        host_name               DVWA-client
        service_description     Disk Usage
        check_command           check_nrpe!check_disk
}

# Logged-in users
define service {
        use                     generic-service
        host_name               DVWA-client
        service_description     Logged-in Users
        check_command           check_nrpe!check_users
}
```

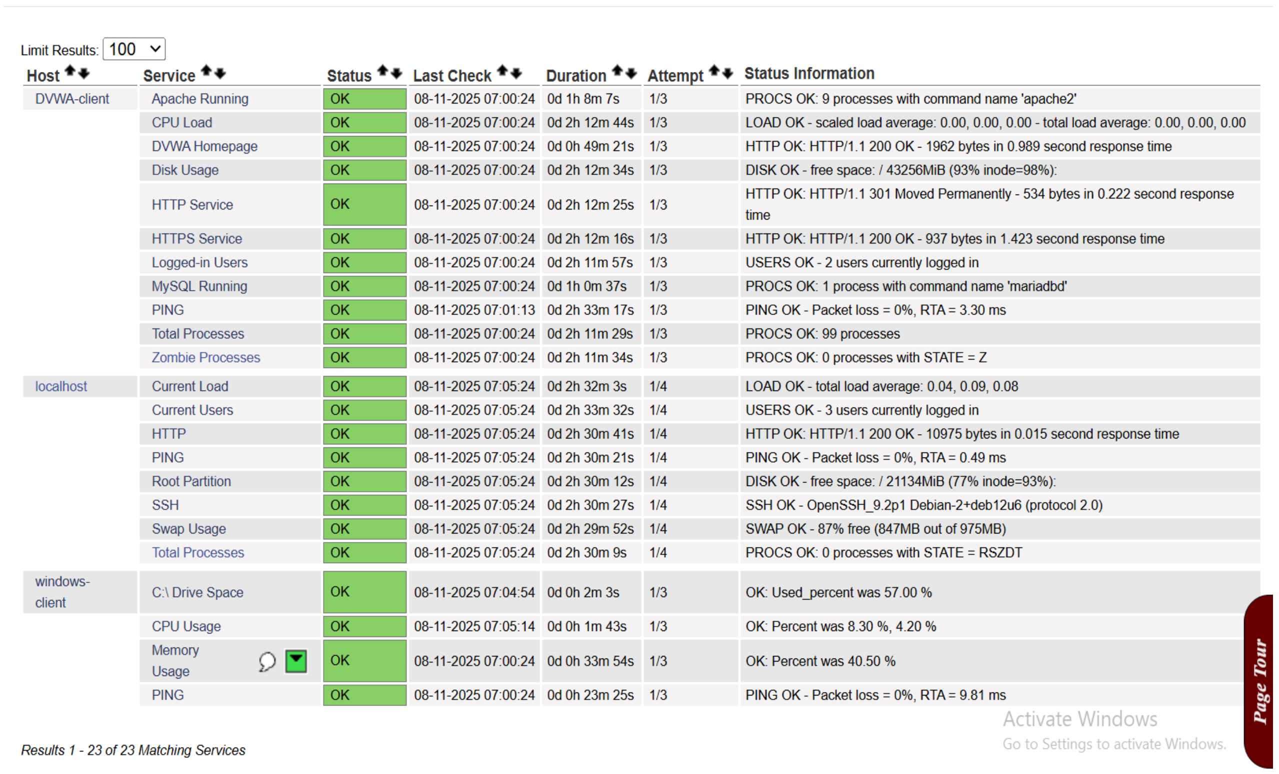


```
# Total processes
define service {
    use                     generic-service
    host_name               DVWA-client
    service_description     Total Processes
    check_command           check_nrpe!check_total_procs
}

# Zombie processes
define service {
    use                     generic-service
    host_name               DVWA-client
    service_description     Zombie Processes
    check_command           check_nrpe!check_zombie_procs
}

# HTTP (port 80)
define service {
    use                     generic-service
    host_name               DVWA-client
    service_description     HTTP Service
    check_command           check_http
}

# HTTPS (port 443)
define service {
    use                     generic-service
    host_name               DVWA-client
    service_description     HTTPS Service
    check_command           check_http!-S
}

# Apache process check
define service {
    use                     generic-service
    host_name               DVWA-client
    service_description     Apache Running
    check_command           check_nrpe!check_procs_apache
}

# MySQL process check
define service {
    use                     generic-service
    host_name               DVWA-client
    service_description     MySQL Running
    check_command           check_nrpe!check_procs_mysql
}

# DVWA Homepage (HTTPS + follow redirects + look for "Login")
define service {
    use                     generic-service
    host_name               DVWA-client
    service_description     DVWA Homepage
    check_command           check_http!-S --onredirect=follow -u /DVWA/login.php -s "Login"
}
```

Step 3: Configure Hosts to be Monitored Window Machine

```
define host {
    use                     windows-server
    host_name               windows-client
    alias                   Windows 10 Machine
    address                 192.168.30.101
    max_check_attempts      5
    check_period            24x7
    notification_interval   30
    notification_period     24x7
    contact_groups          admins
}

define service {
    use                     generic-service
    host_name               windows-client
    service_description     CPU Usage
    check_command           check_ncpa!aditya@123!cpu/percent!80!90
}

define service {
    use                     generic-service
    host_name               windows-client
    service_description     Memory Usage
    check_command           check_ncpa!aditya@123!memory/virtual/percent!80!90
}

define service {
    use                     generic-service
    host_name               windows-client
    service_description     C:\ Drive Space
    check_command           check_ncpa!aditya@123!disk/logical/C:|/used_percent!80!90
}

define service {
    use                     generic-service
    host_name               windows-client
    service_description     PING
    check_command           check_ping!100.0,20%!500.0,60%
}
```

Step 4: Access the Nagios Web UI



Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
DVWA-client	Apache Running	OK	08-11-2025 07:00:24	0d 1h 8m 7s	1/3	PROCS OK: 9 processes with command name 'apache2'
	CPU Load	OK	08-11-2025 07:00:24	0d 2h 12m 44s	1/3	LOAD OK - scaled load average: 0.00, 0.00, 0.00 - total load average: 0.00, 0.00, 0.00
	DVWA Homepage	OK	08-11-2025 07:00:24	0d 0h 49m 21s	1/3	HTTP OK: HTTP/1.1 200 OK - 1962 bytes in 0.989 second response time
	Disk Usage	OK	08-11-2025 07:00:24	0d 2h 12m 34s	1/3	DISK OK - free space: / 43256MiB (93% inode=98%):
	HTTP Service	OK	08-11-2025 07:00:24	0d 2h 12m 25s	1/3	HTTP OK: HTTP/1.1 301 Moved Permanently - 534 bytes in 0.222 second response time
	HTTPS Service	OK	08-11-2025 07:00:24	0d 2h 12m 16s	1/3	HTTP OK: HTTP/1.1 200 OK - 937 bytes in 1.423 second response time
	Logged-in Users	OK	08-11-2025 07:00:24	0d 2h 11m 57s	1/3	USERS OK - 2 users currently logged in
	MySQL Running	OK	08-11-2025 07:00:24	0d 1h 0m 37s	1/3	PROCS OK: 1 process with command name 'mariadbd'
	PING	OK	08-11-2025 07:01:13	0d 2h 33m 17s	1/3	PING OK - Packet loss = 0%, RTA = 3.30 ms
	Total Processes	OK	08-11-2025 07:00:24	0d 2h 11m 29s	1/3	PROCS OK: 99 processes
	Zombie Processes	OK	08-11-2025 07:00:24	0d 2h 11m 34s	1/3	PROCS OK: 0 processes with STATE = Z
localhost	Current Load	OK	08-11-2025 07:05:24	0d 2h 32m 3s	1/4	LOAD OK - total load average: 0.04, 0.09, 0.08
	Current Users	OK	08-11-2025 07:05:24	0d 2h 33m 32s	1/4	USERS OK - 3 users currently logged in
	HTTP	OK	08-11-2025 07:05:24	0d 2h 30m 41s	1/4	HTTP OK: HTTP/1.1 200 OK - 10975 bytes in 0.015 second response time
	PING	OK	08-11-2025 07:05:24	0d 2h 30m 21s	1/4	PING OK - Packet loss = 0%, RTA = 0.49 ms
	Root Partition	OK	08-11-2025 07:05:24	0d 2h 30m 12s	1/4	DISK OK - free space: / 21134MiB (77% inode=93%):
	SSH	OK	08-11-2025 07:05:24	0d 2h 30m 27s	1/4	SSH OK - OpenSSH_9.2p1 Debian-2+deb12u6 (protocol 2.0)
	Swap Usage	OK	08-11-2025 07:05:24	0d 2h 29m 52s	1/4	SWAP OK - 87% free (847MB out of 975MB)
	Total Processes	OK	08-11-2025 07:05:24	0d 2h 30m 9s	1/4	PROCS OK: 0 processes with STATE = RSZDT
windows-client	C:\ Drive Space	OK	08-11-2025 07:04:54	0d 0h 2m 3s	1/3	OK: Used_percent was 57.00 %
	CPU Usage	OK	08-11-2025 07:05:14	0d 0h 1m 43s	1/3	OK: Percent was 8.30 %, 4.20 %
	Memory Usage	OK	08-11-2025 07:00:24	0d 0h 33m 54s	1/3	OK: Percent was 40.50 %
	PING	OK	08-11-2025 07:00:24	0d 0h 23m 25s	1/3	PING OK - Packet loss = 0%, RTA = 9.81 ms

Results 1 - 23 of 23 Matching Services

SNORT

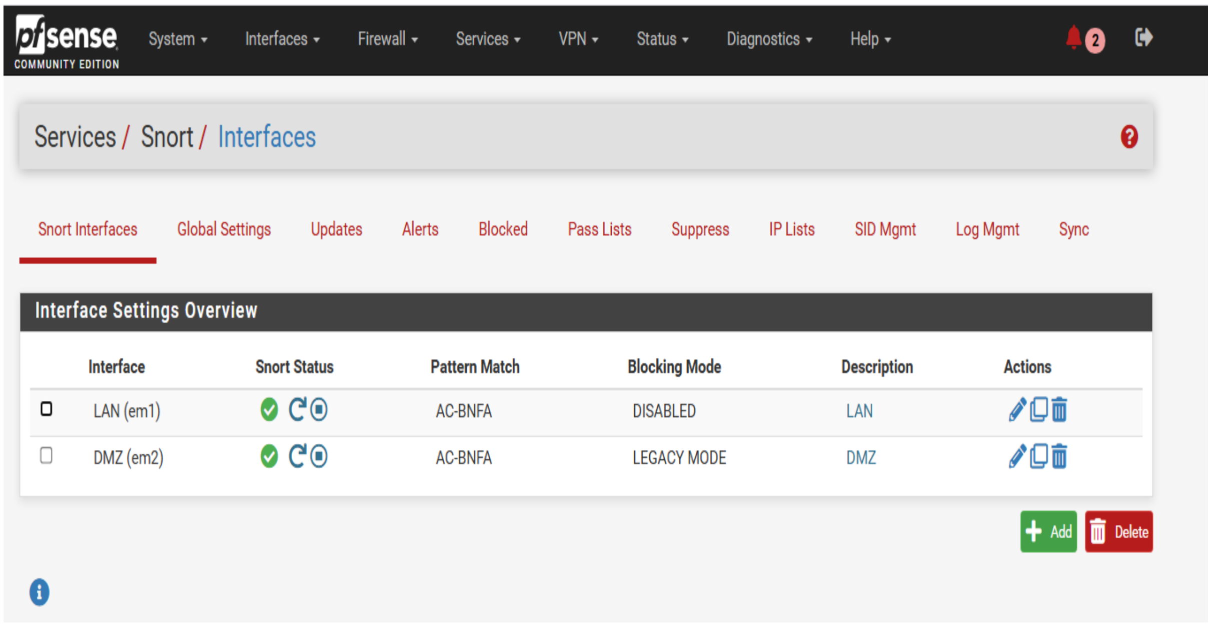
Step 1: Install Snort on pfSense

- **Accessed the pfSense Web UI from the Blue Team's Windows machine.**
- **Navigated to:**

System → Package Manager → Available Packages

Step 2: Add Monitored Interfaces

- **Added LAN (192.168.30.0/24) and DMZ (192.168.20.0/24) interfaces to Snort for monitoring.**
- **Enabled logging and alert generation for both.**

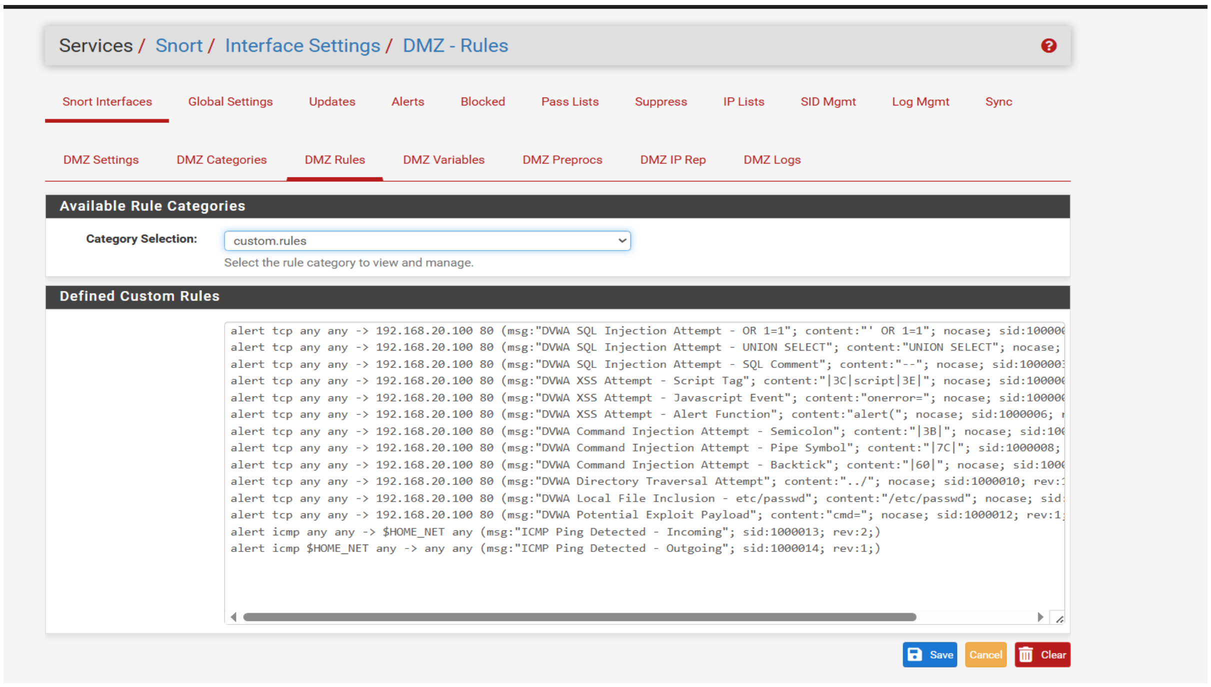


Interface Settings Overview

	Interface	Snort Status	Pattern Match	Blocking Mode	Description	Actions
☐	LAN (em1)	✓	AC-BNFA	DISABLED	LAN	
☐	DMZ (em2)	✓	AC-BNFA	LEGACY MODE	DMZ	

Step 3: Create Detection Rules

- **Added custom detection rules for SQL Injection and XSS attacks:**



Available Rule Categories

Category Selection: custom.rules

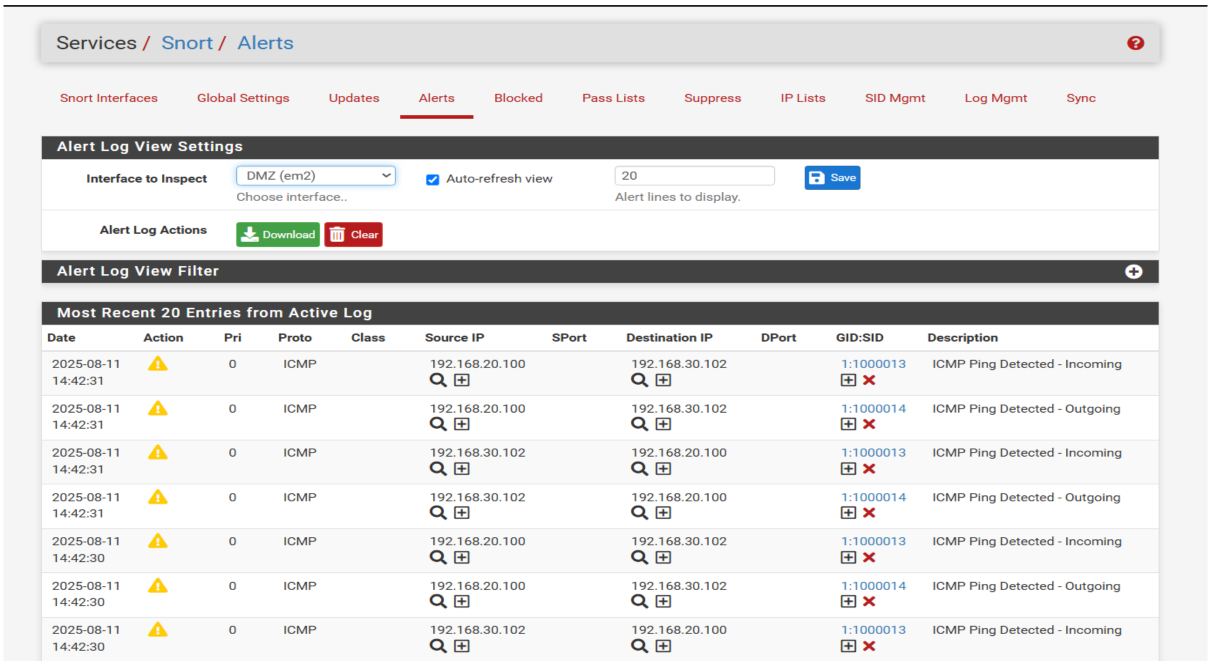
Select the rule category to view and manage.

Defined Custom Rules

```
alert tcp any any -> 192.168.20.100 80 (msg:"DVWA SQL Injection Attempt - OR 1=1"; content:"' OR 1=1"; nocase; sid:100000
alert tcp any any -> 192.168.20.100 80 (msg:"DVWA SQL Injection Attempt - UNION SELECT"; content:"UNION SELECT"; nocase;
alert tcp any any -> 192.168.20.100 80 (msg:"DVWA SQL Injection Attempt - SQL Comment"; content:"--"; nocase; sid:100000
alert tcp any any -> 192.168.20.100 80 (msg:"DVWA XSS Attempt - Script Tag"; content:"|3C|script|3E|"; nocase; sid:100000
alert tcp any any -> 192.168.20.100 80 (msg:"DVWA XSS Attempt - Javascript Event"; content:"onerror="; nocase; sid:100000
alert tcp any any -> 192.168.20.100 80 (msg:"DVWA XSS Attempt - Alert Function"; content:"alert("; nocase; sid:1000006;
alert tcp any any -> 192.168.20.100 80 (msg:"DVWA Command Injection Attempt - Semicolon"; content:"|3B|"; nocase; sid:100
alert tcp any any -> 192.168.20.100 80 (msg:"DVWA Command Injection Attempt - Pipe Symbol"; content:"|7C|"; sid:1000008;
alert tcp any any -> 192.168.20.100 80 (msg:"DVWA Command Injection Attempt - Backtick"; content:"|60|"; nocase; sid:100
alert tcp any any -> 192.168.20.100 80 (msg:"DVWA Directory Traversal Attempt"; content:"../"; nocase; sid:1000010; rev:
alert tcp any any -> 192.168.20.100 80 (msg:"DVWA Local File Inclusion - etc/passwd"; content:"/etc/passwd"; nocase; sid
alert tcp any any -> 192.168.20.100 80 (msg:"DVWA Potential Exploit Payload"; content:"cmd="; nocase; sid:1000012; rev:1
alert icmp any any -> $HOME_NET any (msg:"ICMP Ping Detected - Incoming"; sid:1000013; rev:2;)
alert icmp $HOME_NET any -> any any (msg:"ICMP Ping Detected - Outgoing"; sid:1000014; rev:1;)
```

Step 4: Test and Verify

- **From the Kali Linux machine, executed SQL Injection and XSS payloads against the DVWA server in DMZ.**
- **Snort successfully detected the malicious activity and generated alerts in the pfSense dashboard.**



Alert Log View Settings

Interface to Inspect: DMZ (em2) — Choose interface..

☑ Auto-refresh view

20 — Alert lines to display.

Alert Log Actions

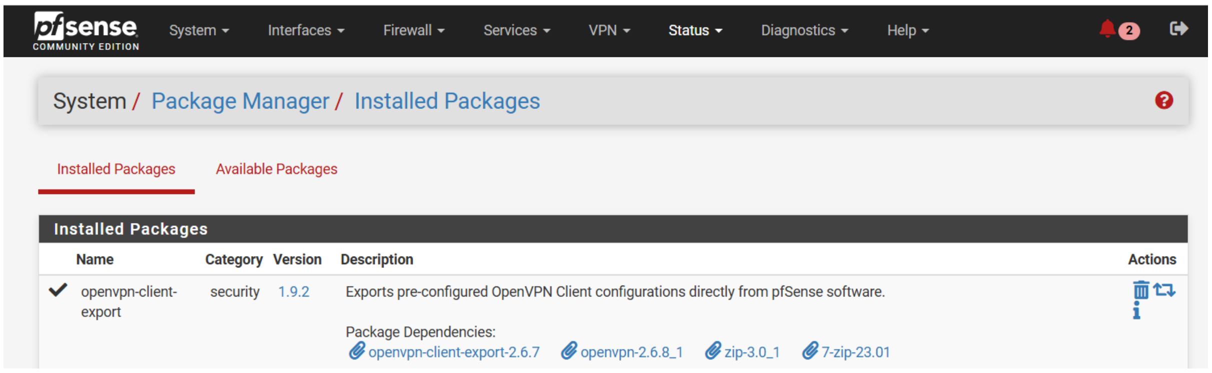
Most Recent 20 Entries from Active Log

Date	Action	Pri	Proto	Class	Source IP	SPort	Destination IP	DPort	GID:SID	Description
2025-08-11 14:42:31	⚠	0	ICMP		192.168.20.100		192.168.30.102		1:1000013	ICMP Ping Detected - Incoming
2025-08-11 14:42:31	⚠	0	ICMP		192.168.20.100		192.168.30.102		1:1000014	ICMP Ping Detected - Outgoing
2025-08-11 14:42:31	⚠	0	ICMP		192.168.30.102		192.168.20.100		1:1000013	ICMP Ping Detected - Incoming
2025-08-11 14:42:31	⚠	0	ICMP		192.168.30.102		192.168.20.100		1:1000014	ICMP Ping Detected - Outgoing
2025-08-11 14:42:30	⚠	0	ICMP		192.168.20.100		192.168.30.102		1:1000013	ICMP Ping Detected - Incoming
2025-08-11 14:42:30	⚠	0	ICMP		192.168.20.100		192.168.30.102		1:1000014	ICMP Ping Detected - Outgoing
2025-08-11 14:42:30	⚠	0	ICMP		192.168.30.102		192.168.20.100		1:1000013	ICMP Ping Detected - Incoming

OPENVPN

Step 1: Install and Enable OpenVPN

- **Accessed pfSense Web UI from the Blue Team's Windows machine.**
- **Navigated to:**



System / Package Manager / Installed Packages

Installed Packages | Available Packages

Installed Packages

	Name	Category	Version	Description	Actions
✓	openvpn-client-export	security	1.9.2	Exports pre-configured OpenVPN Client configurations directly from pfSense software. Package Dependencies: openvpn-client-export-2.6.7 openvpn-2.6.8_1 zip-3.0_1 7-zip-23.01	

Step 2: Configure OpenVPN Server

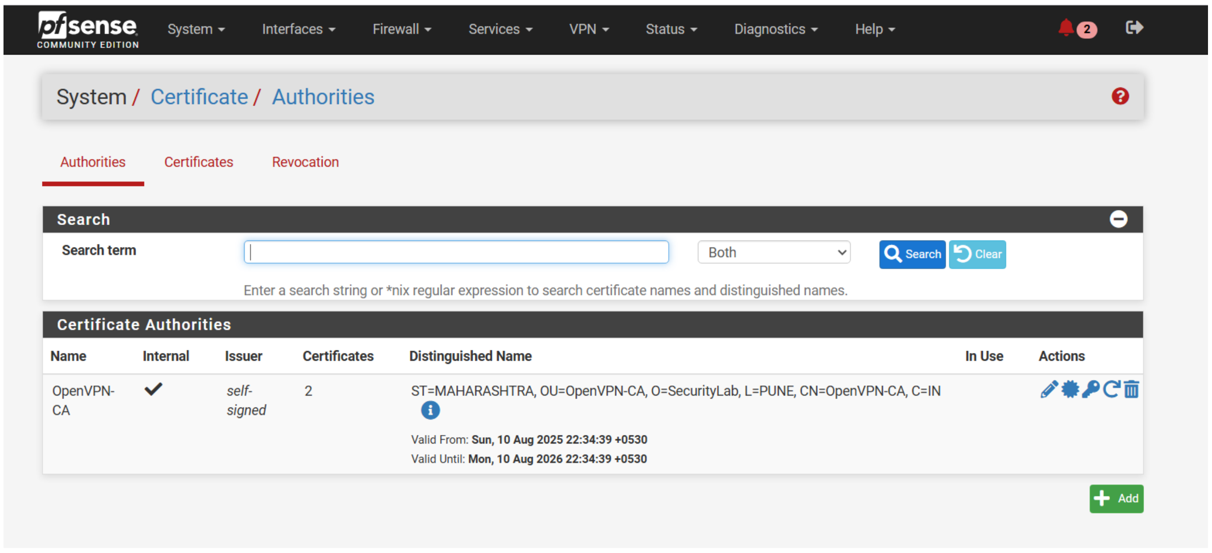
- **Navigated to**

VPN → OpenVPN → Wizards

- **Selected Local User Access and created a new Certificate Authority (CA).**
- **Generated a server certificate and applied it to the OpenVPN server instance.**
- **Set the listening interface to LAN, protocol to UDP, and port to 1194 (default).**

Step 3: Create OpenVPN User and Certificate

- **Created a new VPN user in pfSense and assigned a client certificate.**
- **Exported the .ovpn configuration file using the OpenVPN Client Export Utility.**



System / Certificate / Authorities

Authorities | Certificates | Revocation

Search

Search term: [ ] Both — Search — Clear

Enter a search string or *nix regular expression to search certificate names and distinguished names.

Certificate Authorities

Name	Internal	Issuer	Certificates	Distinguished Name	In Use	Actions
OpenVPN-CA	✓	*self-signed*	2	ST=MAHARASHTRA, OU=OpenVPN-CA, O=SecurityLab, L=PUNE, CN=OpenVPN-CA, C=IN Valid From: **Sun, 10 Aug 2025 22:34:39 +0530** Valid Until: **Mon, 10 Aug 2026 22:34:39 +0530**		

Add

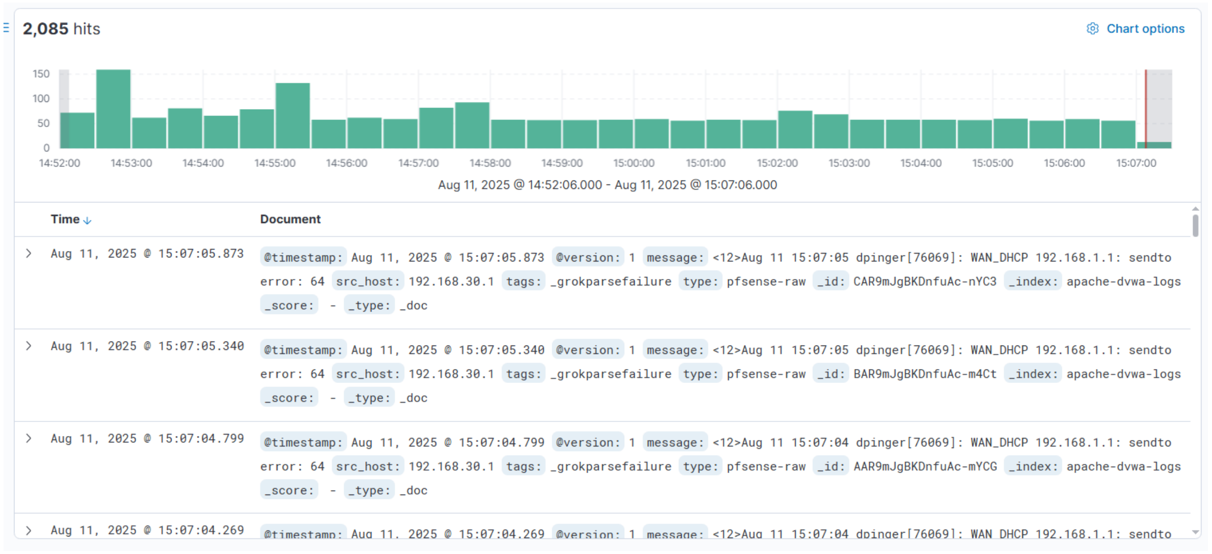
Log Monitoring (Blue Team – ELK Stack & CLI)

The Blue Team monitored security events using the ELK Stack (Elasticsearch, Logstash, Kibana) and direct packet capture via tcpdump.

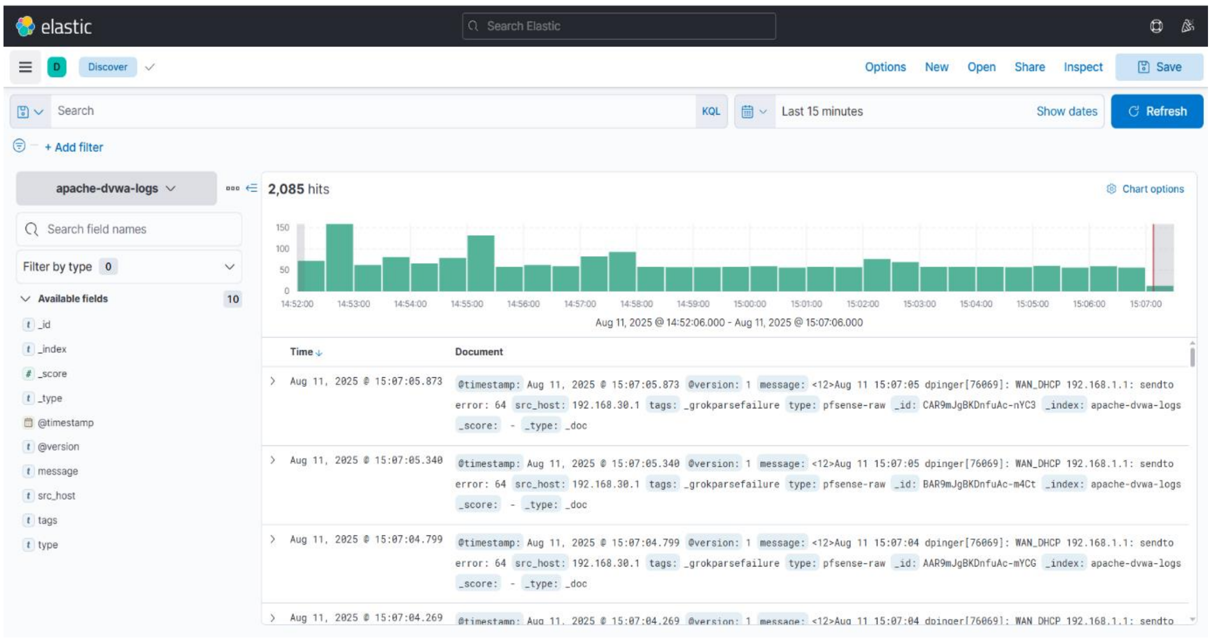
```
aditya@Blue-Team:~$ sudo tcpdump -i any udp port 5514 -nn -A
tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
12:26:16.028766 enp0s3 In  IP 192.168.30.1.514 > 192.168.30.102.5514: SYSLOG local0.alert, length: 118
E....0..@..r.......f.....~%0<129>Aug 11 12:26:16 snort[54749]: [1:1000014:1] ICMP Ping Detected - Outgoing {ICMP} 192.168.30.102 -> 192
.168.20.100
12:26:16.028767 enp0s3 In  IP 192.168.30.1.514 > 192.168.30.102.5514: SYSLOG local0.alert, length: 118
E.......@.$........f.....~>N<129>Aug 11 12:26:16 snort[54749]: [1:1000013:2] ICMP Ping Detected - Incoming {ICMP} 192.168.30.102 -> 192
.168.20.100
12:26:16.030639 enp0s3 In  IP 192.168.30.1.514 > 192.168.30.102.5514: SYSLOG local0.alert, length: 118
E.......@..........f.....~%0<129>Aug 11 12:26:16 snort[54749]: [1:1000014:1] ICMP Ping Detected - Outgoing {ICMP} 192.168.20.100 -> 192
.168.30.102
12:26:16.030640 enp0s3 In  IP 192.168.30.1.514 > 192.168.30.102.5514: SYSLOG local0.alert, length: 118
E.......@..........f.....~>N<129>Aug 11 12:26:16 snort[54749]: [1:1000013:2] ICMP Ping Detected - Incoming {ICMP} 192.168.20.100 -> 192
.168.30.102
12:26:16.030640 enp0s3 In  IP 192.168.30.1.514 > 192.168.30.102.5514: SYSLOG local0.alert, length: 118
E...I`..@.sC.......f.....~%0<129>Aug 11 12:26:16 snort[54749]: [1:1000014:1] ICMP Ping Detected - Outgoing {ICMP} 192.168.30.102 -> 192
.168.20.100
12:26:16.031553 enp0s3 In  IP 192.168.30.1.514 > 192.168.30.102.5514: SYSLOG local0.alert, length: 118
E...L...@.o........f.....~>N<129>Aug 11 12:26:16 snort[54749]: [1:1000013:2] ICMP Ping Detected - Incoming {ICMP} 192.168.30.102 -> 192
.168.20.100
12:26:16.033292 enp0s3 In  IP 192.168.30.1.514 > 192.168.30.102.5514: SYSLOG local0.alert, length: 118
E...B...@.y........f.....~%0<129>Aug 11 12:26:16 snort[54749]: [1:1000014:1] ICMP Ping Detected - Outgoing {ICMP} 192.168.20.100 -> 192
.168.30.102
12:26:16.033292 enp0s3 In  IP 192.168.30.1.514 > 192.168.30.102.5514: SYSLOG local0.alert, length: 118
E...(...@..........f.....~>N<129>Aug 11 12:26:16 snort[54749]: [1:1000013:2] ICMP Ping Detected - Incoming {ICMP} 192.168.20.100 -> 192
.168.30.102
12:26:16.134645 enp0s3 In  IP 192.168.30.1.514 > 192.168.30.102.5514: SYSLOG auth.alert, length: 131
E....w..@..........f......+.<33>Aug 11 12:26:16 snort[42468]: [1:2000002:1] LAN ALERT: Outgoing Ping from Internal Host {ICMP} 192.168.
```

Kibana Dashboard Monitoring

Kibana provided a centralized view of Snort alerts, pfSense firewall logs, and DVWA web server events. This allowed analysts to visualize attack patterns, timelines, and correlate Red Team activities with network logs.



```
Aug 11, 2025 @ 15:07:05.873  @timestamp: Aug 11, 2025 @ 15:07:05.873 @version: 1 message: <12>Aug 11 15:07:05 dpinger[76069]: WAN_DHCP 192.168.1.1: sendto
                             error: 64 src_host: 192.168.30.1 tags: _grokparsefailure type: pfsense-raw _id: CAR9mJgBKDnfuAc-nYC3 _index: apache-dvwa-logs
                             _score: - _type: _doc
Aug 11, 2025 @ 15:07:05.340  @timestamp: Aug 11, 2025 @ 15:07:05.340 @version: 1 message: <12>Aug 11 15:07:05 dpinger[76069]: WAN_DHCP 192.168.1.1: sendto
                             error: 64 src_host: 192.168.30.1 tags: _grokparsefailure type: pfsense-raw _id: BAR9mJgBKDnfuAc-m4Ct _index: apache-dvwa-logs
                             _score: - _type: _doc
Aug 11, 2025 @ 15:07:04.799  @timestamp: Aug 11, 2025 @ 15:07:04.799 @version: 1 message: <12>Aug 11 15:07:04 dpinger[76069]: WAN_DHCP 192.168.1.1: sendto
                             error: 64 src_host: 192.168.30.1 tags: _grokparsefailure type: pfsense-raw _id: AAR9mJgBKDnfuAc-mYCG _index: apache-dvwa-logs
                             _score: - _type: _doc
Aug 11, 2025 @ 15:07:04.269  @timestamp: Aug 11, 2025 @ 15:07:04.269 @version: 1 message: <12>Aug 11 15:07:04 dpinger[76069]: WAN_DHCP 192.168.1.1: sendto
```



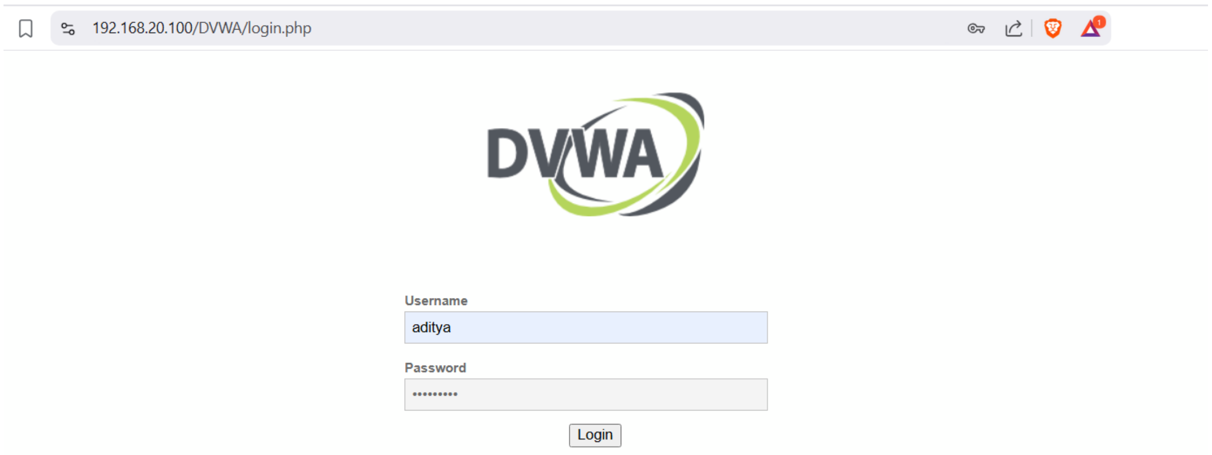
```
Aug 11, 2025 @ 15:07:05.873  @timestamp: Aug 11, 2025 @ 15:07:05.873 @version: 1 message: <12>Aug 11 15:07:05 dpinger[76069]: WAN_DHCP 192.168.1.1: sendto
                             error: 64 src_host: 192.168.30.1 tags: _grokparsefailure type: pfsense-raw _id: CAR9mJgBKDnfuAc-nYC3 _index: apache-dvwa-logs
                             _score: - _type: _doc
Aug 11, 2025 @ 15:07:05.340  @timestamp: Aug 11, 2025 @ 15:07:05.340 @version: 1 message: <12>Aug 11 15:07:05 dpinger[76069]: WAN_DHCP 192.168.1.1: sendto
                             error: 64 src_host: 192.168.30.1 tags: _grokparsefailure type: pfsense-raw _id: BAR9mJgBKDnfuAc-m4Ct _index: apache-dvwa-logs
                             _score: - _type: _doc
Aug 11, 2025 @ 15:07:04.799  @timestamp: Aug 11, 2025 @ 15:07:04.799 @version: 1 message: <12>Aug 11 15:07:04 dpinger[76069]: WAN_DHCP 192.168.1.1: sendto
                             error: 64 src_host: 192.168.30.1 tags: _grokparsefailure type: pfsense-raw _id: AAR9mJgBKDnfuAc-mYCG _index: apache-dvwa-logs
                             _score: - _type: _doc
Aug 11, 2025 @ 15:07:04.269  @timestamp: Aug 11, 2025 @ 15:07:04.269 @version: 1 message: <12>Aug 11 15:07:04 dpinger[76069]: WAN_DHCP 192.168.1.1: sendto
```

DVWA Deployment & TLS Certificate Configuration

Step 1: Install Apache, MySQL, and PHP

- **sudo apt update**
- **sudo apt install apache2 mariadb-server php php-mysqli php-gd libapache2-mod-php**

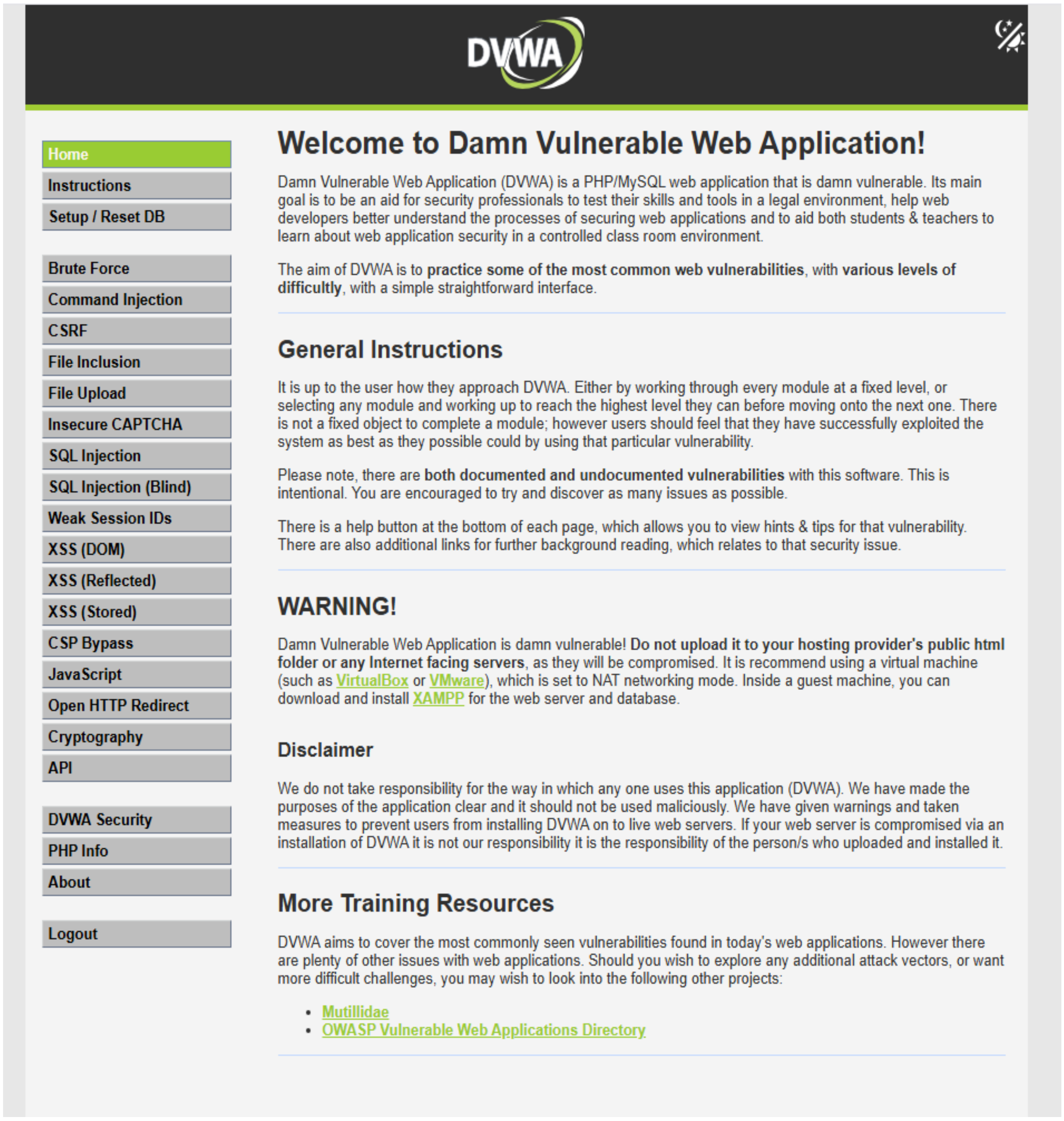
  - **Apache serves the DVWA web pages.**
  - **MariaDB stores the DVWA backend database.**
  - **PHP enables dynamic processing of DVWA scripts.**


192.168.20.100/DVWA/login.php

Username: aditya

Password: •••••••••

Step 2: ACCESS SUCCESSFULL LOGING PAGE



Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerabilities** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can download and install XAMPP for the web server and database.

Disclaimer

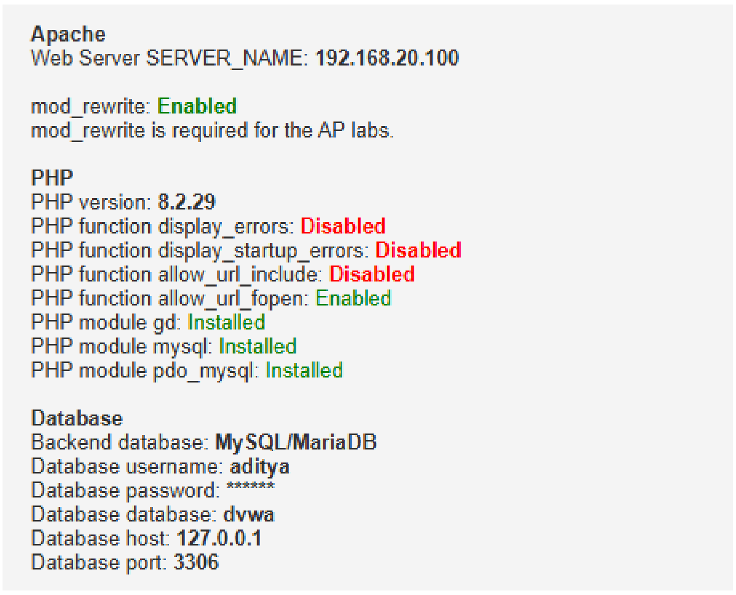
We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors, or want more difficult challenges, you may wish to look into the following other projects:

- Mutillidae
- OWASP Vulnerable Web Applications Directory

SETUP PAGE :-



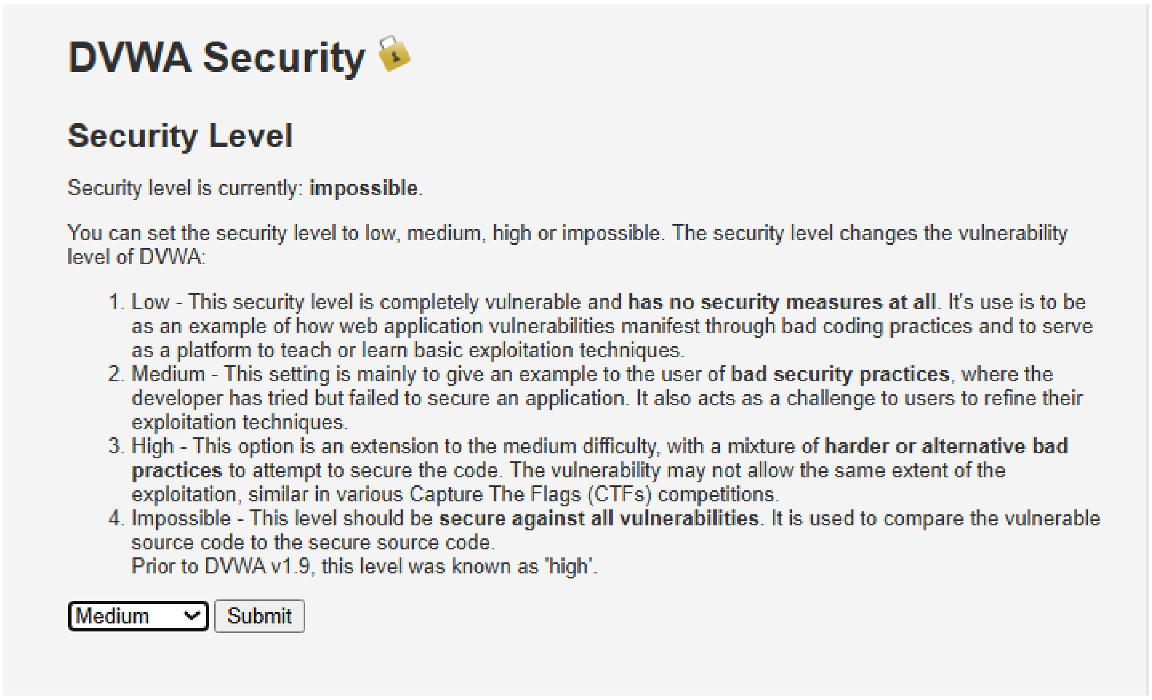
Apache
Web Server SERVER_NAME: **192.168.20.100**

mod_rewrite: **Enabled**
mod_rewrite is required for the AP labs.

PHP
PHP version: **8.2.29**
PHP function display_errors: **Disabled**
PHP function display_startup_errors: **Disabled**
PHP function allow_url_include: **Disabled**
PHP function allow_url_fopen: Enabled
PHP module gd: Installed
PHP module mysql: Installed
PHP module pdo_mysql: Installed

Database
Backend database: **MySQL/MariaDB**
Database username: **aditya**
Database password: ******
Database database: **dvwa**
Database host: **127.0.0.1**
Database port: **3306**

DVWA SECURITY



DVWA Security

Security Level

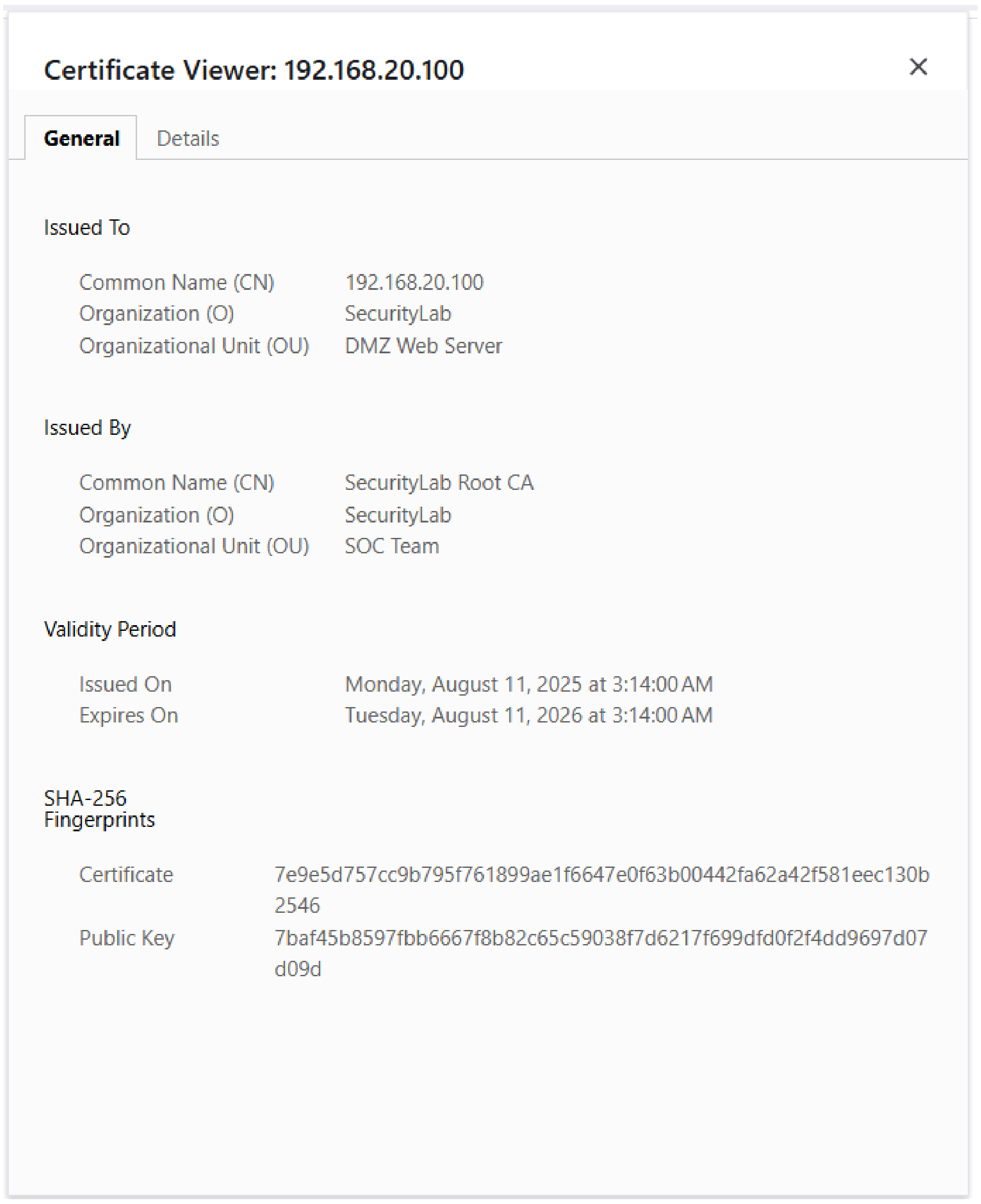
Security level is currently: **impossible**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
   Prior to DVWA v1.9, this level was known as 'high'.

Verify HTTPS Access

- **Accessed DVWA via: CERTIFICIATE**



Certificate Viewer: 192.168.20.100

Issued To

Common Name (CN)	192.168.20.100
Organization (O)	SecurityLab
Organizational Unit (OU)	DMZ Web Server

Issued By

Common Name (CN)	SecurityLab Root CA
Organization (O)	SecurityLab
Organizational Unit (OU)	SOC Team

Validity Period

Issued On	Monday, August 11, 2025 at 3:14:00 AM
Expires On	Tuesday, August 11, 2026 at 3:14:00 AM

SHA-256 Fingerprints

Certificate	7e9e5d757cc9b795f761899ae1f6647e0f63b00442fa62a42f581eec130b2546
Public Key	7baf45b8597fbb6667f8b82c65c59038f7d6217f699dfd0f2f4dd9697d07d09d

7. APPLICATIONS

The enterprise security simulation lab developed in this project can be applied in:

1. Cybersecurity Training – Provides a safe and realistic environment for students and professionals to practice both offensive (Red Team) and defensive (Blue Team) skills.
2. Incident Response Drills – Simulates attack scenarios to test and improve organizational response procedures.
3. Tool Evaluation – Enables security teams to test IDS/IPS, firewalls, SIEMs, and VPNs in a controlled setting before deployment in production.
4. Penetration Testing Practice – Offers a variety of exploitable services (e.g., DVWA) to improve vulnerability assessment and exploitation skills.
5. Network Segmentation Testing – Validates inter-zone security policies between WAN, LAN, and DMZ segments.

8. ADVANTAGES & DISADVANTAGES

Advantages:

- Realistic simulation of enterprise network architecture.
- Safe environment for testing high-risk cyberattacks.
- Supports both attack and defense skill development.
- Modular design allows easy reconfiguration for new scenarios.
- Low-cost setup using open-source tools and virtualization.

Disadvantages:

- Requires a relatively powerful host machine to run multiple VMs.
- Limited scalability compared to cloud-based cyber ranges.
- Self-signed certificates may not fully replicate enterprise PKI environments.
- Learning curve for configuring advanced security tools like ELK and Snort.

9. CONCLUSION

This project successfully designed and deployed a comprehensive virtual enterprise security lab that mirrors real-world network architectures and security operations. By integrating **pfSense**, **Snort**, **Nagios**, **OpenVPN**, **ELK Stack**, and a deliberately vulnerable **DVWA** server, the environment enabled a seamless interplay between **Red Team attack simulations** and **Blue Team defense strategies**.

The lab provided a controlled yet realistic platform for testing intrusion detection, incident response, and network monitoring in a multi-segmented architecture comprising WAN, LAN, and DMZ zones. Through the simulation of sophisticated attack vectors and the deployment of layered defenses, the project not only validated security tool configurations but also enhanced practical cybersecurity skills.

Overall, this environment bridges the gap between theoretical concepts and real-world application, making it a valuable resource for training, research, and security assessments. Looking ahead, the setup can be further enhanced by introducing **automated attack frameworks**, **cloud-based scalability**, and **integration of advanced SIEM and threat intelligence solutions** to reflect the evolving cybersecurity landscape.

10. REFERENCES

1. **Damn Vulnerable Web Application (DVWA) – Official Documentation**
   https://github.com/digininja/DVWA
2. **pfSense – Official User Guide and Documentation**
   https://docs.netgate.com/pfsense/en/latest/
3. **Snort IDS – Official Documentation**
   https://www.snort.org/
4. **Nagios Core – User Manual and Configuration Guide**
   https://www.nagios.org/projects/nagios-core/
5. **OpenVPN – Community Edition Documentation**
   https://openvpn.net/community-resources/
6. **ELK Stack – Elastic Official Documentation**
   https://www.elastic.co/what-is/elk-stack
7. **Kali Linux – Official Documentation**
   https://www.kali.org/docs/
8. **XCA (X Certificate and Key Management) – User Manual**
   https://hohnstaedt.de/xca/