Relatório de Andamento: Detecção de SQL Injection com Suricata e ELK

Disciplina: Tópicos Avançados em Segurança Computacional

Projeto: Detecção de SQL Injection em Ambiente Web Simulado

Versão do Documento: 1.1 (Arquitetura de 3 VMs Revisada)

1. Introdução

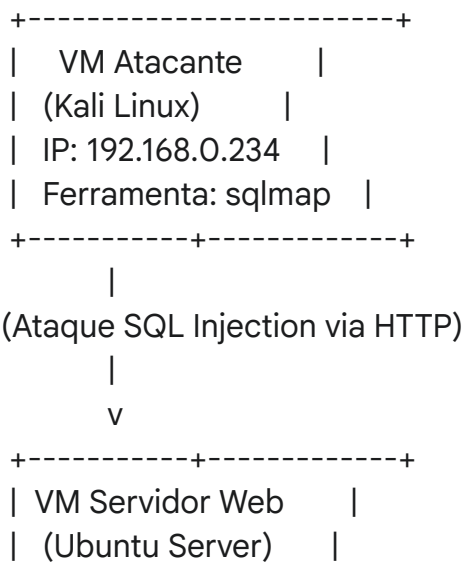
Este documento detalha o progresso na construção de um laboratório para simulação e detecção de ataques de *SQL Injection*. O objetivo do projeto é criar um ambiente controlado para executar um ataque, monitorar a rede e os sistemas em tempo real, e centralizar os logs gerados para análise e correlação em um sistema SIEM (*Security Information and Event Management*).

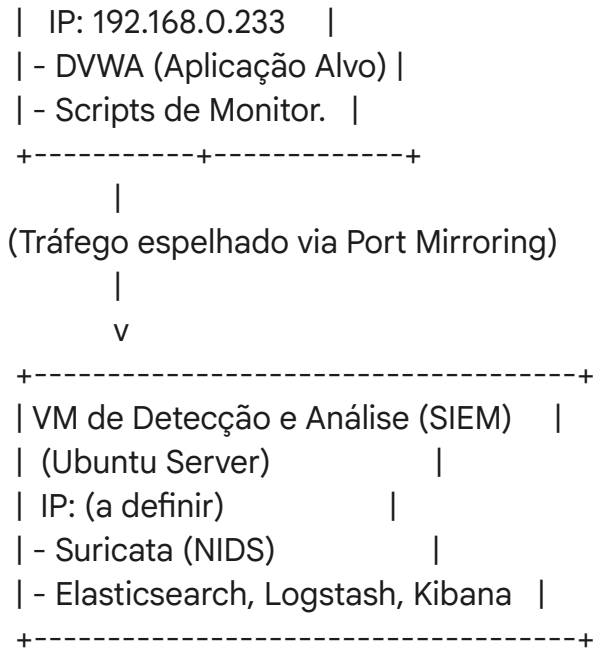
Até o momento, a infraestrutura virtual foi estabelecida com uma arquitetura de três VMs, e a plataforma SIEM está em fase de construção.

2. Topologia do Ambiente

A infraestrutura do laboratório foi revisada para uma arquitetura de três máquinas virtuais (VMs) distintas, garantindo o isolamento entre os componentes de aplicação, detecção e ataque. Todas operam na rede local simulada (192.168.0.0/24).

A detecção de rede será realizada através do espelhamento de tráfego (*Port Mirroring*) da VM do Servidor Web para a VM de Detecção.





3. Configuração do Ambiente

3.1. VM Servidor Web

Esta VM é dedicada exclusivamente a hospedar a aplicação alvo e os scripts de monitoramento local.

- **Aplicação Web Vulnerável:** Foi implantado o **Damn Vulnerable Web Application (DVWA)**, uma aplicação PHP/MySQL projetada para ser intencionalmente vulnerável.
- **Monitoramento Local por Scripts:** Foram desenvolvidos dois scripts Bash para monitoramento de recursos do sistema. Estes scripts são executados periodicamente via crontab.
  - **Agendamento (crontab -e):**
    */10 * * * * /path/to/monitor_processos.sh
    */10 * * * * /path/to/monitor_conexoes.sh

3.2. VM Atacante

Uma instância padrão do **Kali Linux** é utilizada como a plataforma de ataque. A principal ferramenta a ser empregada é o sqlmap, que automatiza o processo de detecção e exploração de falhas de *SQL Injection*.

3.3. VM de Detecção e Análise (SIEM)

Esta VM centraliza todas as ferramentas de segurança e análise de logs.

- **Sistema de Detecção de Intrusão (IDS):** Foi instalado o **Suricata**. Ele operará no modo NIDS (Network-based IDS), monitorando uma interface de rede secundária configurada para receber o tráfego espelhado da VM do Servidor Web. As regras do conjunto *Emerging Threats (ET) Open* foram ativadas.
- **Stack ELK (SIEM):** Para atender ao requisito de "Coleta, análise e correlação de logs", foi iniciada a montagem do **Stack ELK** nesta mesma VM. A instalação progrediu até a seguinte etapa:
  1. **Java Development Kit (JDK):** Instalado como pré-requisito.
  2. **Elasticsearch:** Instalado e configurado para aceitar conexões da rede local.
  3. **Kibana:** Instalado e configurado para ser acessível pela rede.
  4. **Logstash:** A instalação foi concluída. O próximo passo é a configuração de um *pipeline* para processar os logs.

4. Scripts de Monitoramento Desenvolvidos

Os seguintes scripts foram criados para a coleta de informações básicas do sistema na VM do Servidor Web.

4.1. monitor_processos.sh

Este script coleta informações sobre os 10 processos que mais consomem memória no sistema e armazena o resultado em um arquivo de log.

```
#!/bin/bash
# Script para monitorar os processos com maior consumo de memória.

# Define o arquivo de log
LOG="/var/log/monitor_processos.log"

# Adiciona um cabeçalho com data e hora ao log
echo "---- $(date) ----" >> "$LOG"

# Executa o comando 'ps' para listar processos, ordena por uso de memória (%mem)
# e seleciona os 10 primeiros resultados (incluindo o cabeçalho).
ps aux --sort=-%mem | head -n 10 >> "$LOG"

# Adiciona uma linha em branco para melhor legibilidade
echo "" >> "$LOG"
```

4.2. monitor_conexoes.sh

Este script lista todas as conexões de rede ativas (TCP e UDP) no servidor e salva a saída em um arquivo de log.

```
#!/bin/bash
# Script para monitorar todas as conexões de rede ativas.

# Define o arquivo de log
LOG="/var/log/monitor_conexoes.log"

# Adiciona um cabeçalho com data e hora ao log
echo "---- $(date) ----" >> "$LOG"

# Utiliza o comando 'ss' para listar todos os sockets de rede (TCP e UDP),
# sem resolver nomes de serviço (n) ou de host (a).
ss -tuna >> "$LOG"

# Adiciona uma linha em branco para melhor legibilidade
echo "" >> "$LOG"
```

5. Próximos Passos

1. **Configurar o Espelhamento de Tráfego:** Garantir que a configuração de *Port Mirroring* no hipervisor (VirtualBox) está funcionando e que a VM de Detecção recebe o tráfego da VM Web.
2. **Finalizar a Configuração do ELK:**
   - Criar o arquivo de configuração do *pipeline* do Logstash para processar os logs do Suricata.
   - Instalar e configurar o **Filebeat** na VM de Detecção para ler o arquivo local /var/log/suricata/eve.json e enviá-lo para o Logstash na mesma máquina.
3. **Execução do Cenário de Ataque:**
   - Lançar o ataque de *SQL Injection* utilizando o sqlmap a partir da VM Atacante.
4. **Análise e Visualização:**
   - Verificar a chegada dos logs no Kibana.
   - Criar um *Index Pattern* e um *dashboard* para apresentar os resultados do ataque.