

Gabriela Paola Sereniski - a64937
Reginaldo Gregório de Souza Neto - a61482

Primeiro Trabalho Prático

Instituto Politécnico de Bragança – IPB
Escola Superior de Tecnologia e Gestão – ESTiG
Mestrado em Informática
Segurança em Sistemas Informáticos
Professor Tiago Miguel Ferreira Guimaraes Pedrosa

Bragança
Dezembro / 2024

Sumário

1	Introdução	3
2	Cenário	3
3	proxy	4
4	Sistema de Produção	5
5	Honeypot	7
6	SIEM	7
6.1	Wazuh Indexer	8
6.2	Wazuh Server	8
6.3	Wazuh Dashboard	9
6.4	Wazuh Agent	9
7	Ataques e monitoramento	11
8	Conclusão	14
9	Referências	15
	Anexo A - Configuração de rede das máquinas	16
	Anexo B - Configuração do Nginx e ModSecurity	18
	Anexo C - Configurações dos contêineres de produção	21
	Anexo D - Log SIEM de tentativa de login incorreto	23
	Anexo E - Log SIEM descoberta de invasor via brute force	25
	Anexo F - Log SIEM SQL Injection	27

1 Introdução

A análise de segurança em ambientes simulados é uma prática comum para compreender vulnerabilidades, testar medidas de proteção e estudar comportamentos maliciosos em sistemas computacionais. Esses ambientes controlados permitem reproduzir cenários reais de ataques e defesas, facilitando a identificação de pontos fracos e a avaliação da eficácia das soluções de segurança implementadas, além de ser um ótimo meio de aprendizado.

Este relatório apresenta o desenvolvimento de um ambiente utilizando o Virtual-Box, composto por cinco máquinas virtuais com as seguintes funções: uma máquina atacante, um sistema de monitoramento e análise de segurança (SIEM), uma máquina que hospeda dois contêineres Docker (um servidor Apache e um banco de dados SQL) e um Honeypot, que possui as mesmas características da máquina anterior, porém com versões desatualizadas e com aplicações vulneráveis. Além disso, foi configurado uma máquina que analisa as ações do atacante no servidor e realiza proxy reverso, para encaminhar tráfego suspeito ao Honeypot e legítimo ao servidor verdadeiro.

O objetivo principal deste trabalho foi entender a configuração do proxy reverso, tecnologias de detecção de ações suspeitas e compreender como funciona um SIEM. Para testar o ambiente, explorar vulnerabilidades, monitorar interações e implementar técnicas de segurança cibernética. Estudando o comportamento de invasores e a eficácia dos mecanismos de defesa, busca-se aprimorar o entendimento e a aplicação de estratégias de proteção em sistemas computacionais.

Todos os arquivos de configurações utilizados estão disponíveis no [GitHub](#).

2 Cenário

A configuração de rede do ambiente de simulação foi realizada no VirtualBox, com a criação de uma rede NAT e três redes internas, conforme ilustrado na figura 1.

A Rede NAT é utilizada para conectar a máquina atacante a uma das interfaces do proxy. A Rede PROD é uma rede interna utilizada apenas para a conexão do proxy à VM de servidores legítimos, o que o isola do acesso irrestrito via NAT. A Rede HONEY é destinada ao tráfego de informações entre o proxy e o Honeypot. Ela é utilizada apenas para o redirecionamento de requisições consideradas suspeitas ou maliciosas ao servidor legítimo. Caso um atacante comprometa o Honeypot ou sua rede, o impacto fica contido por não haver conexão direta com os demais serviços, minimizando o risco para o ambiente de produção e monitoramento.

Por fim, a Rede MGMT (Management) é responsável pela comunicação entre o proxy, servidor de produção e Honeypot com o sistema de monitoramento. Segmentar

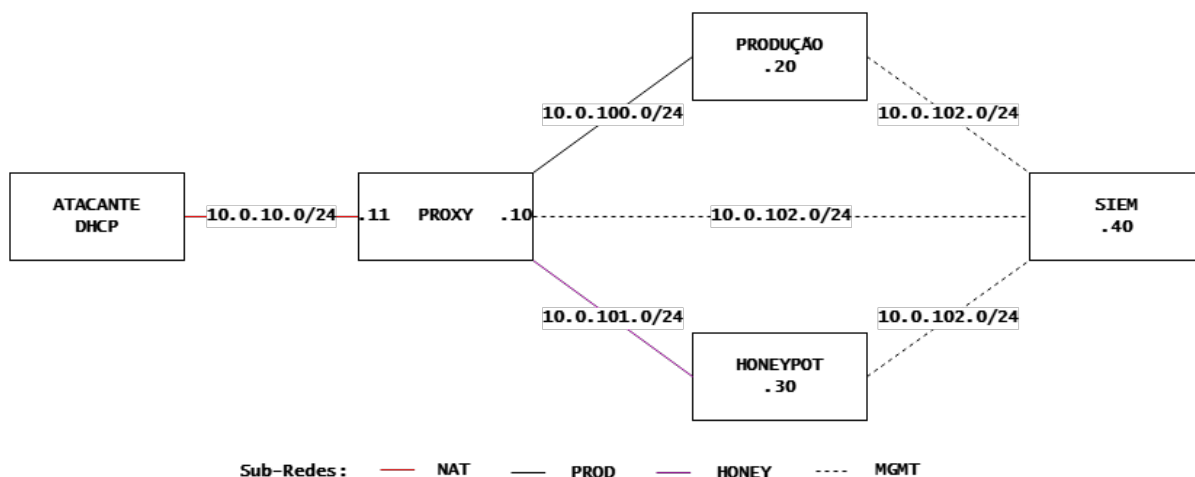


Figura 1 – Diagrama da rede

a rede dessa forma diminui as chances de comprometimento completo do cenário e facilita sua recuperação, uma vez que se uma delas falha, os demais serviços continuam funcionando.

Para configurar as interfaces de rede de cada VM foi feito uso do Netplan, uma utilidade disponível para sistemas Linux que permite a configuração da rede com arquivos YAML. Na pasta `/etc/netplan/` foi adicionado um arquivo chamado `00-config.yaml` contendo os IPs desejados para cada interface. O conteúdo deste arquivo pode ser encontrado no [Anexo A - Configuração de rede das máquinas](#).

3 proxy

O proxy é executado em uma máquina virtual que tem Debian 12 como sistema operacional. Nela, foi configurado o Nginx ([NGINX, 2024](#)), que é um servidor web de alto desempenho, também utilizado como balanceador de carga, proxy reverso e proxy de e-mail. No caso do nosso cenário, ele serve como porta de entrada para o servidor legítimo ou o Honeypot. O Nginx foi configurado em conjunto com o ModSecurity ([ModSecurity Project, 2024](#)). O ModSecurity é um firewall de aplicações web de código aberto projetado para proteger servidores e aplicações contra ataques, detectando e prevenindo vulnerabilidades com base em regras configuráveis.

Os passos seguidos para instalação e configuração base, tanto do Nginx quanto do ModSecurity, são explicados no tutorial *How to Set Up ModSecurity with Nginx on Debian/Ubuntu* ([XIAO, 2022](#)). O tutorial aborda a instalação do ModSecurity e do Nginx, a configuração do módulo ModSecurity e a ativação das OWASP Core Rule Set (CRS), um conjunto popular de regras para detecção e prevenção de ataques como SQL Injection e Cross-Site Scripting (XSS). Além disso, explica como habilitar o monitoramento ou

modo de bloqueio, testar a configuração e interpretar os logs para identificar potenciais ameaças.

O guia também oferece orientações para personalizar as regras de segurança conforme as necessidades do ambiente, destacando práticas recomendadas para proteção contra vulnerabilidades web. O CRS avalia cada requisição com um conjunto extenso de regras que calculam um índice de anomalia. Caso esse índice ultrapasse certo limiar, por padrão, o ModSecurity faz o bloqueio do acesso.

No diretório `/etc/nginx/sites-available/` foi adicionado o arquivo `proxy.conf` (disponível no [Anexo B - Configuração do Nginx e ModSecurity](#)), que permite que o Nginx encaminhe requisições externas para o devido servidor. O ModSecurity foi configurado em conjunto com o Nginx, na pasta `/etc/nginx/modsec/`, e foram adicionadas as regras padrão do CRS na versão 4.9.0, disponíveis em ([OWASP, 2024](#)). O arquivo principal (`modsecurity.conf`) define o comportamento do firewall, incluindo o nível de detecção e a ativação das regras do CRS, enquanto os arquivos de regras, contidos no subdiretório `/coreruleset-4.9.0/rules/` contêm padrões para detectar ataques comuns, como força bruta e SQL Injection.

Por padrão, o tráfego é encaminhado para o servidor real, entretanto, foram feitas adições ao CRS para que, com base nas regras de bloqueio, fossem criadas condições para realizar o redirecionamento automático de ações suspeitas para o servidor do Honeypot. Esse fluxo protege os servidores reais, isola ataques em um ambiente controlado e mantém logs das atividades para monitoramento e investigação. As regras adicionais podem ser visualizadas no [Anexo B - Configuração do Nginx e ModSecurity](#).

Com base nas regras de bloqueio, criamos condições para realizar o redirecionamento automático para o servidor Honeypot. Também no [Anexo B - Configuração do Nginx e ModSecurity](#) é possível ver as regras customizadas utilizadas.

Quando alguma anomalia é detectada, o ModSecurity registra informações sobre elas no arquivo `/var/log/modsec_audit.log`, arquivo que deverá ser monitorado pelo SIEM.

4 Sistema de Produção

O sistema de produção é executado em uma máquina com Ubuntu Server 24.04.1 LTS. A aplicação escolhida para servir de alvo dos ataques consiste em um site de consulta de Pokémons que possui duas telas interativas. A primeira é uma tela de login que possui os campos para inserção das credenciais de acesso (Figura 2), e a segunda é uma tela de listagem dos Pokémons pertencentes ao usuário logado (Figura 3).

Quando um usuário faz login na plataforma, a mesma retorna a ele apenas os

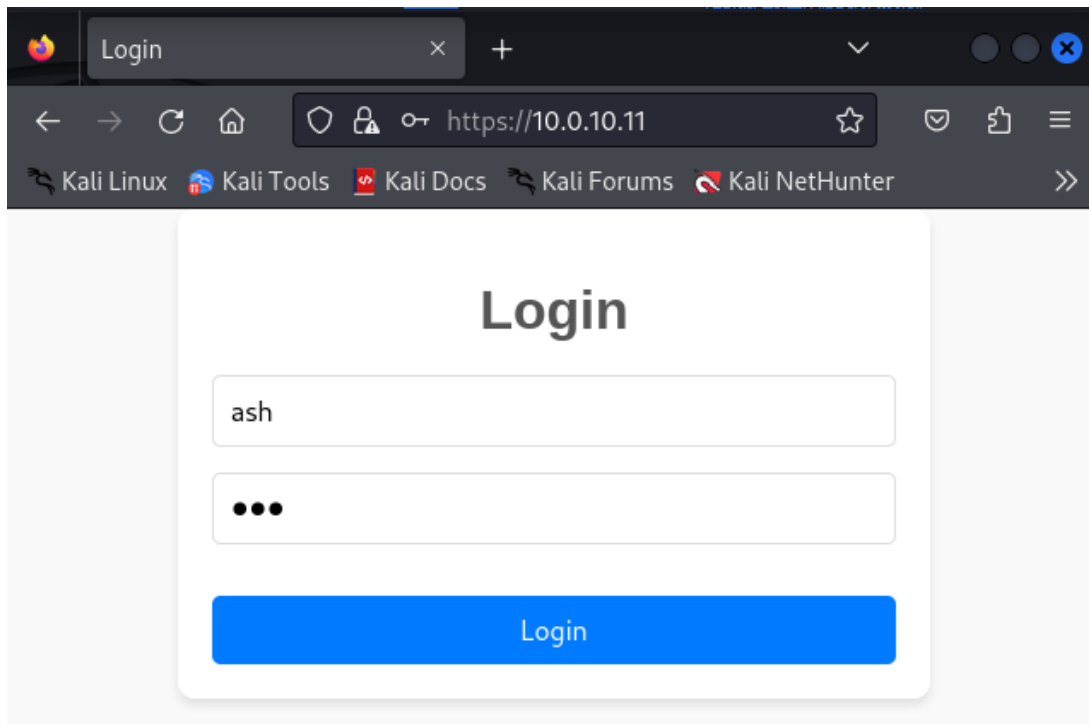


Figura 2 – Página de Login - PROD.

Pokémons disponíveis na base de dados que pertencem a esse usuário. Logo, um usuário não é capaz de ver informações sobre Pokémons que não o pertencem. A exceção a regra é o usuário do tipo administrador, que é capaz de visualizar todos os Pokémons da base de dados.

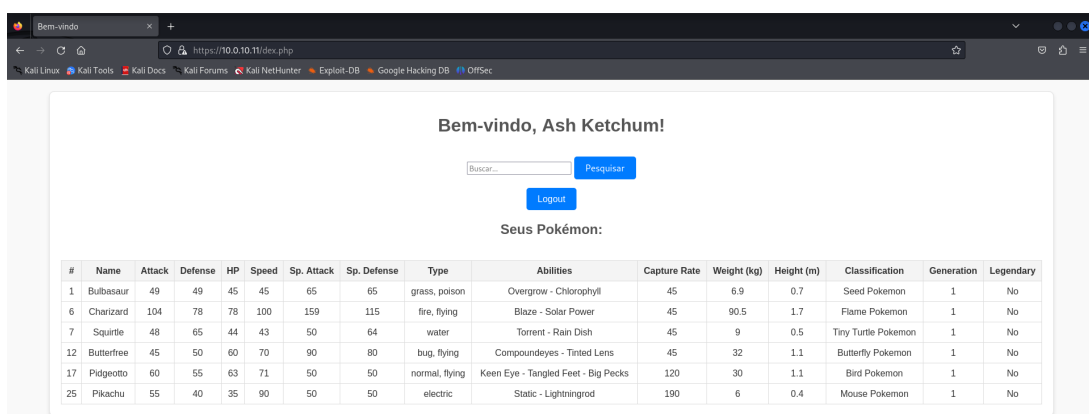


Figura 3 – Listagem de Pokémons - PROD

Para prover esse serviço, foi realizada a configuração de contêineres Docker com um servidor Apache PHP na versão 8.1 e um servidor MySQL 8.0. No servidor Apache foi utilizado o módulo `mod_rewrite` para redirecionar automaticamente as solicitações feitas à porta 80 (HTTP) para a porta 443 (HTTPS). Esse redirecionamento é importante para garantir que mesmo os usuários que acessam o serviço de maneira não segura sejam automaticamente transferidos para uma conexão segura.

Além disso, foi utilizado um certificado SSL/TLS autoassinado para viabilizar conexões HTTPS. O uso desse tipo de certificado se deve à ausência de um domínio público registrado, o que torna inviável a obtenção de um certificado emitido por uma Autoridade Certificadora (CA) tradicional. Os arquivos de configuração da aplicação podem ser visualizados no [Anexo C - Configurações dos contêineres de produção](#).

5 Honeypot

O Honeypot, que executa em uma máquina virtual com Debian 12, foi implementado como uma cópia do servidor de produção mas em um sistema controlado, sem a presença de dados reais e projetado para atrair atacantes e registrar suas atividades. Para isso, foram instaladas imagens nas versões 5.6 do Apache PHP e 5.6 do MySQL nos mesmos moldes de configuração da máquina anterior.

É válido ressaltar que o servidor legítimo e o Honeypot possuem os mesmos serviços e interfaces dinâmicas, como pode ser visto na Figura 4, para que o atacante acredite que está interagindo com o sistema real. A aplicação do Honeypot também contém diversas vulnerabilidades à SQL Injection em seu código PHP.

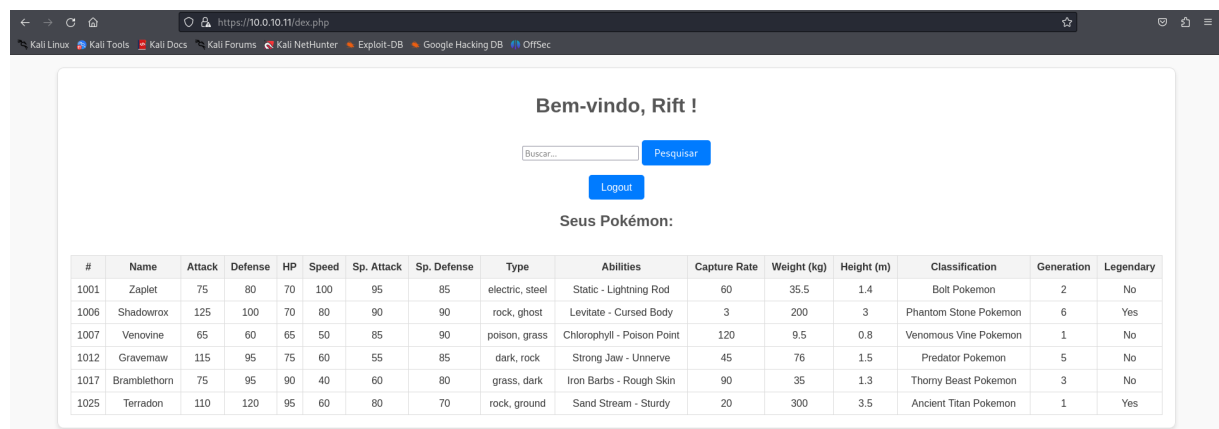


Figura 4 – Página de Login - HONEYPOT

Entretanto, as informações contidas no banco de dados do Honeypot, embora sigam os mesmos padrões do servidor real, são falsas, garantindo que, em caso de invasão, nenhuma informação sensível seja comprometida.

6 SIEM

Para monitorar os nós do cenário optou-se por utilizar uma máquina virtual Ubuntu 24.04.1 com o Wazuh ([Wazuh, Inc., 2024d](#)). O Wazuh é uma plataforma de segurança de código aberto que oferece monitoramento de integridade, detecção de intru-

sões, análise de logs e conformidade, integrando funcionalidades para proteger sistemas e dados em tempo real.

A configuração do Wazuh foi feita de acordo com os passos descritos em sua documentação oficial, disponível em ([Wazuh Team, 2024](#)). A aplicação é composta por 4 componentes distintos: Indexer, Server, Dashboard e Agent. No cenário, o Indexer, Server e o Dashboard foram instalados da máquina virtual de SIEM.

6.1 Wazuh Indexer

É o componente responsável por armazenar e indexar alertas gerados pelo Server Wazuh. Baseado em OpenSearch, permite consultas rápidas para análises e relatórios. Para sua instalação, foram criados certificados SSL com o script `wazuh-certs-tool.sh`, para permitir comunicação segura entre os componentes do sistema. Um arquivo de configuração é utilizado para definir o nome do nó e outros parâmetros necessários antes da geração dos certificados.

Em seguida, o Wazuh Indexer é instalado no nó. O arquivo de configuração principal, `opensearch.yml`, é ajustado para especificar as definições de cluster e os caminhos dos certificados SSL. Após a configuração, o Indexer é iniciado, e o script `indexer-security-init.sh` é executado para carregar as informações dos certificados e finalizar a configuração e deixando o Indexer pronto para o uso. Os passos descritos em ([Wazuh, Inc., 2024b](#)) foram seguidos sem desvios, apenas alterando os valores de IP `<indexer-node-ip>`, `<wazuh-manager-ip>` e `<dashboard-node-ip>` para 10.0.102.40.

6.2 Wazuh Server

Atua como o núcleo do Wazuh, gerenciando os agentes, processando os dados coletados e aplicando regras de análise para identificar ameaças e anomalias. Sua configuração começa com a instalação do Wazuh Manager e do Filebeat. Após adicionar o repositório adequado e importar a chave GPG, os pacotes necessários são instalados usando o gerenciador de pacotes do sistema. O Filebeat, responsável por encaminhar alertas e eventos ao Wazuh Indexer, é configurado com um arquivo pré-definido, onde os endereços dos nós do Indexer e as credenciais de autenticação são ajustados.

Certificados SSL gerados previamente são aplicados para permitir comunicação segura. Além disso, as credenciais para conexão com o Indexer são armazenadas no keystore do Wazuh Manager, enquanto o Filebeat é configurado para operar com segurança, utilizando os certificados apropriados. Por fim, o Wazuh Manager e o Filebeat são inicializados ([Wazuh, Inc., 2024c](#)).

6.3 Wazuh Dashboard

É a interface gráfica baseada em web que permite aos usuários visualizar os dados de segurança, configurar políticas, monitorar eventos e gerar relatórios de forma amigável e interativa. Sua configuração inicia-se com a instalação do pacote `wazuh-dashboard` através do gerenciador de pacotes do sistema. Após a instalação, é necessário editar o arquivo de configuração `opensearch_dashboards.yml` para definir o `server.host` como `10.0.102.40`, permitindo conexões remotas, e especificar os endereços dos nós do Wazuh Indexer no parâmetro `opensearch.hosts`.

Em seguida, os certificados SSL previamente gerados são implantados no diretório `/etc/wazuh-dashboard/certs`, garantindo comunicações seguras. Após ajustar as permissões dos certificados, o serviço do Wazuh Dashboard é habilitado e iniciado utilizando comandos do `systemd`. Finalmente, o arquivo `wazuh.yml` é configurado para apontar para o endereço do Wazuh Server, permitindo que o dashboard se comunique corretamente com o servidor (Wazuh, Inc., 2024a). Com esses passos concluídos, o Wazuh Dashboard estará acessível via navegador web, proporcionando uma interface gráfica para monitorar e visualizar eventos de segurança (Figura 5).

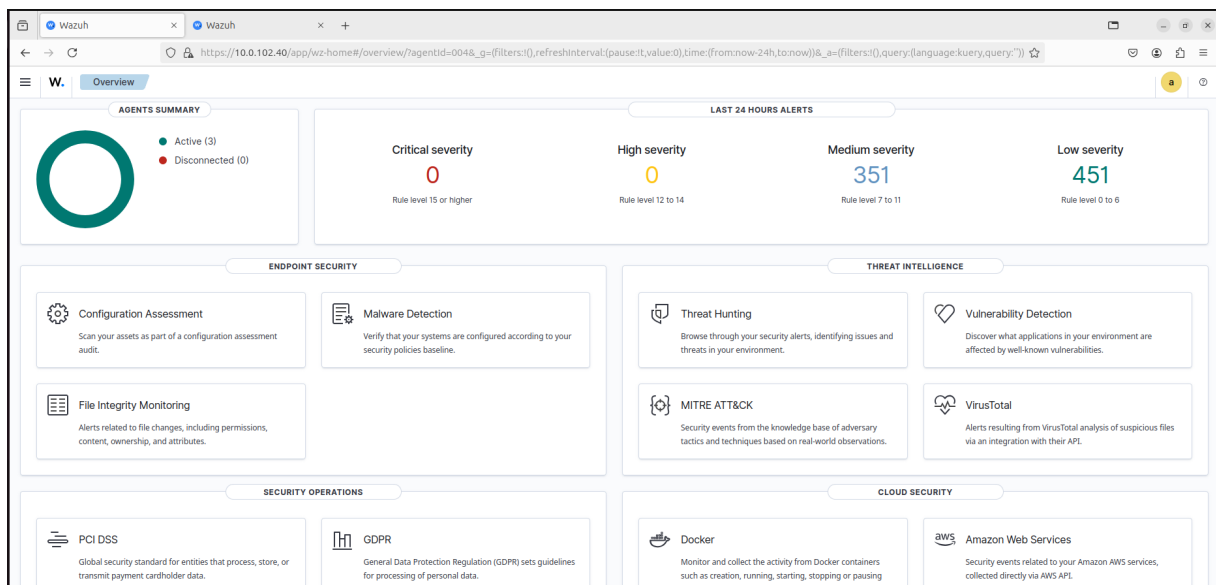


Figura 5 – Wazuh Endpoints

6.4 Wazuh Agent

Instalado em dispositivos monitorados, coleta dados de segurança como logs, arquivos de configuração e métricas do sistema, enviando-os ao Server Wazuh para análise.

Sua configuração em sistemas Linux inicia com a adição do repositório oficial do Wazuh, importando a chave GPG e configurando o repositório apropriado para o gerenciador de pacotes do sistema. Após atualizar as informações dos pacotes, define-se a

variável `WAZUH_MANAGER` com o endereço IP ou hostname do servidor Wazuh, nesse caso, `10.0.102.40`, proporcionando a configuração automática do agente durante a instalação.

Em seguida, instala-se o pacote `wazuh-agent` utilizando o gerenciador de pacotes. Após a instalação, o serviço do agente é habilitado e iniciado com os comandos do `systemd`. Para garantir compatibilidade entre as versões do agente e do servidor, foram desabilitadas as atualizações automáticas do Wazuh, ajustando o repositório para evitar upgrades não intencionais. Com esses passos, o Wazuh Agent estará ativo no sistema Linux, pronto para monitorar o endpoint e comunicar-se com o servidor Wazuh.

Esses passos foram executados no proxy, servidor legítimo e Honeypot. Por padrão, de acordo com a configuração presente em `/var/ossec/etc/ossec.conf`, o Wazuh Agent realiza uma ampla gama de monitoramentos no endpoint para detectar alterações e atividades suspeitas. Ele analisa logs do sistema, como syslog, logs de serviços (exemplo: Apache/Nginx) e logs de pacotes como `dpkg.log`.

Monitora a integridade de arquivos e diretórios críticos, como `/etc`, `/usr/bin` e `/boot`, detectando modificações, exclusões e criações. Além disso, verifica políticas de segurança para identificar rootkits, trojans e outras ameaças, analisando processos em execução, portas abertas e arquivos do sistema.

O agente também coleta informações detalhadas do inventário do sistema, incluindo hardware, sistema operacional, pacotes instalados, processos, portas abertas e configurações de rede, atualizando esses dados periodicamente. Verifica a conformidade com políticas de segurança baseadas em padrões como os benchmarks CIS e executa respostas automáticas para mitigar ameaças detectadas, como o bloqueio de IPs maliciosos ou encerramento de processos suspeitos.

Também realiza análises de comandos como `df`, `netstat` e `last`, monitorando uso de disco, portas em escuta e sessões de login recentes, enquanto permite a inclusão de monitoramentos personalizados. Essas funcionalidades são configuradas para verificações periódicas e em tempo real, garantindo vigilância contínua e detalhada do endpoint.

Particularmente, no proxy, foi adicionado ao arquivo de configuração `/var/ossec/etc/ossec.conf` o trecho de Código 1 para que o agente monitorasse também o arquivo de log gerado pelo ModSecurity.

```
1 <localfile>
2   <log_format>json</log_format>
3   <location>/var/log/modsec_audit.log</location>
4 </localfile>
```

Arquivo 1 – Adição do arquivo de log do ModSecurity aos arquivos monitorados pelo Wazuh.

Já no servidor legítimo e no Honeypot foi adicionado ao arquivo de configuração o trecho de Código 2 para que o agente monitorasse também os arquivos de log gerados

pelos servidores MySQL e Apache PHP.

```

1 <localfile>
2   <log_format>apache</log_format>
3   <location>/home/admin/docker/apache-php/logs/access.log</location>
4 </localfile>
5
6 <localfile>
7   <log_format>apache</log_format>
8   <location>/home/admin/docker/apache-php/logs/error.log</location>
9 </localfile>
10
11 <localfile>
12   <log_format>syslog</log_format>
13   <location>/home/admin/docker/mysql-logs/error.log</location>
14 </localfile>

```

Arquivo 2 – Adição do arquivo de log do ModSecurity aos arquivos monitorados pelo Wazuh.

Com todos os elementos configurados, é possível monitorar cada agente pelo Wazuh Dashboard, como na Figura 6.

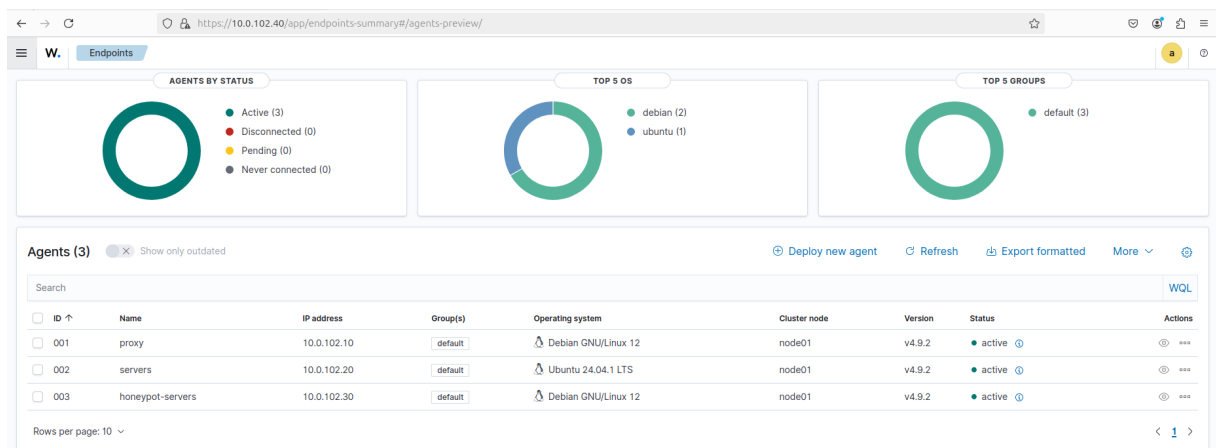
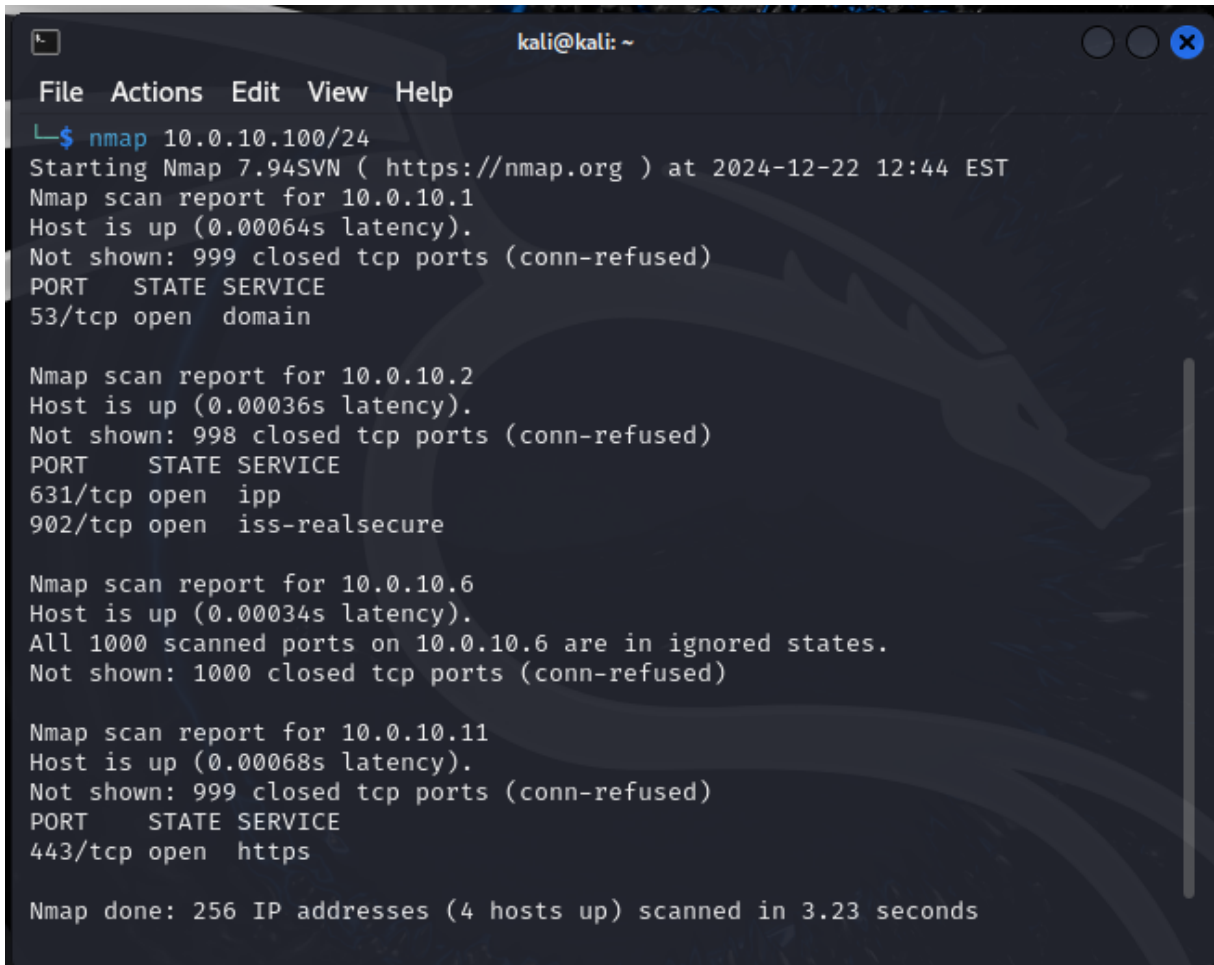


Figura 6 – Wazuh Dashboard

7 Ataques e monitoramento

Para realizar testes de segurança, utilizou-se uma máquina virtual com Kali Linux. Como já explicado, a máquina do atacante só consegue se comunicar com a interface do proxy exposta na rede NAT, portanto na visão do atacante ele sempre está acessando o IP 10.0.10.11. Para início da análise, foi executado um escaneamento com o NMAP na rede 10.0.10.0/24 para identificar os IPs presentes (Figura 7).

Ao observar a porta HTTPS aberta, decide-se acessar o endereço pelo navegador para analisar seu conteúdo. Caso o usuário tenha login e senha, ele poderá interagir com



```
kali@kali: ~
File Actions Edit View Help
└─$ nmap 10.0.10.100/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-22 12:44 EST
Nmap scan report for 10.0.10.1
Host is up (0.00064s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE
53/tcp    open  domain

Nmap scan report for 10.0.10.2
Host is up (0.00036s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE
631/tcp    open  ipp
902/tcp    open  iss-realsecure

Nmap scan report for 10.0.10.6
Host is up (0.00034s latency).
All 1000 scanned ports on 10.0.10.6 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Nmap scan report for 10.0.10.11
Host is up (0.00068s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE
443/tcp    open  https

Nmap done: 256 IP addresses (4 hosts up) scanned in 3.23 seconds
```

Figura 7 – Resultado do nmap na máquina do atacante

o sistema normalmente, caso contrário, é necessário tentar descobrir um usuário e senha válidos. Supondo que o atacante resolva testar alguns usuários e senhas comuns, ele se deparará com uma tela de credenciais incorretas (Figura 8). Nessas situações, logs sobre tentativas de acesso serão registradas pelo Wazuh, como o que pode ser visto no [Anexo D - Log SIEM de tentativa de login incorreto](#).

Uma vez que foi configurado um limite máximo de 5 tentativas inválidas para um único IP, o Nginx fará o redirecionamento do tráfego para o servidor do Honeypot. No Honeypot, o usuário pode tentar diversas credenciais, quantas vezes quiser, até que encontre alguma válida, pode ser feito o uso da ferramenta Hydra, por exemplo. Caso o usuário encontre credenciais válidas, ele terá acesso aos dados falsos do Honeypot, e enquanto isso, as requisições realizadas são registradas ([Anexo E - Log SIEM descoberta de invasor via brute force](#)).

Um próximo passo pode ser a tentativa de realizar SQL Injection no campo de busca. O atacante logo percebe que é possível executar comandos SQL sem muitos problemas, desde que o número de valores retornados pelo comando seja igual ao número de colunas da tabela. Ao testar campos de tabela comuns em uma tabela de usuário, por

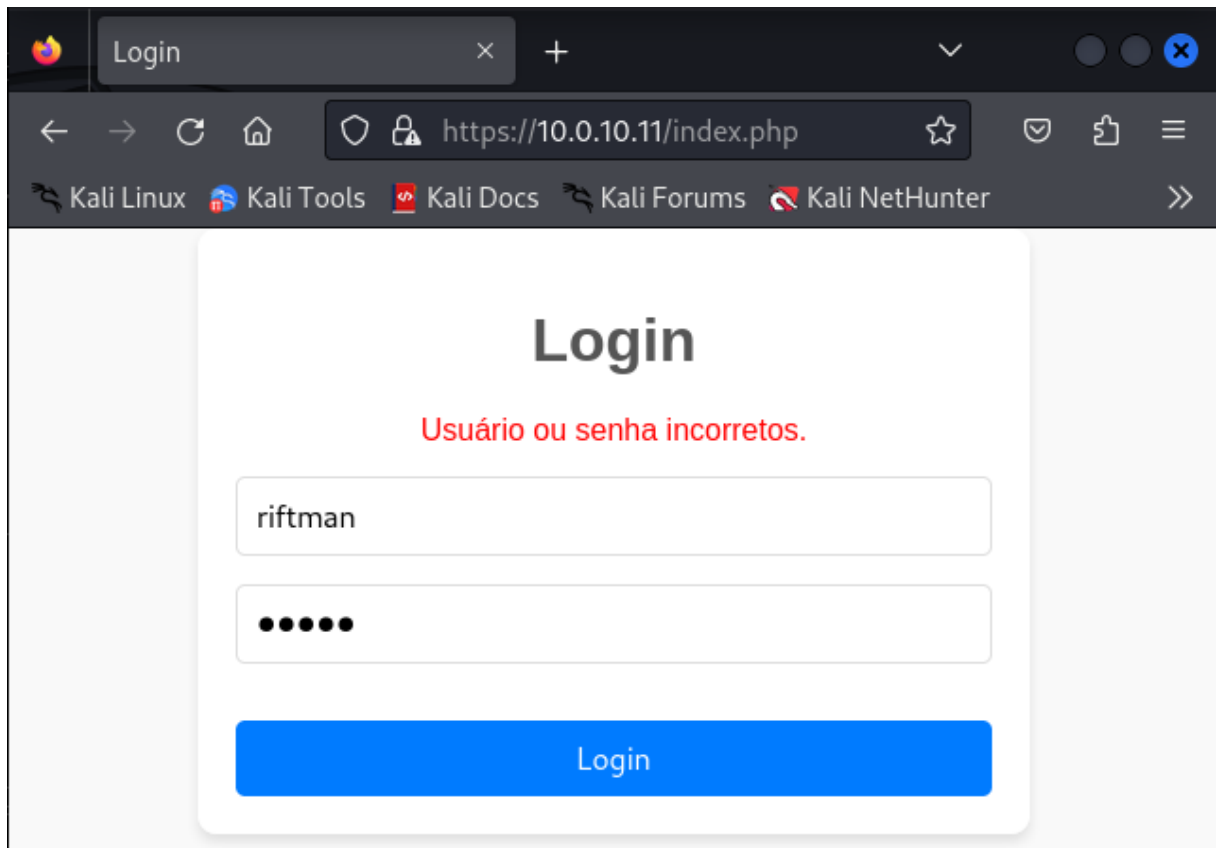


Figura 8 – Aviso de credenciais incorretas.

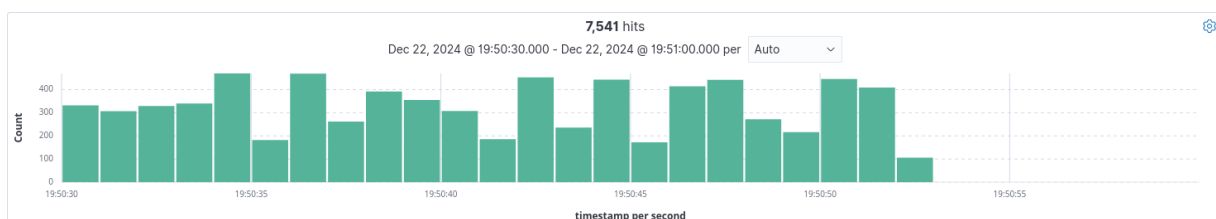


Figura 9 – Número de requisições elevado em um curto período de tempo.

exemplo, é possível revelar todas as informações confidenciais dos usuários registrados (Figura 10). Entretanto, ao executar uma busca com padrões de SQL Injection, as regras do ModSecurity captam a requisição e acusam a anomalia ([Anexo F - Log SIEM SQL Injection](#)).

Idealmente, o honeypot deve incluir mecanismos básicos de segurança, como criptografia e medidas de hardening, para simular de forma mais realista um sistema em produção. Em um cenário onde a aplicação retorna senhas em texto puro (**plaintext**), torna-se evidente para o atacante que o alvo não é um sistema real (ao menos espera-se que não seja). A implementação dessas medidas contribui para aumentar a credibilidade do honeypot e melhorar a eficácia do ambiente controlado.

Com essas estratégias, o atacante que interagir com o sistema terá maior dificuldade em distinguir entre o servidor real e o honeypot, o que o incentivará a continuar

executando ações maliciosas. Durante esse processo, todas as suas interações podem ser registradas e enviadas para análise.

8 Conclusão

Notou-se que a configuração de um ambiente que em teoria não é muito complexo já levanta uma série de questões sobre cada detalhe que deve ser considerado ao tentar manter cada elemento verdadeiramente seguro. São muitas variáveis, e é necessário ter o domínio de diversas áreas da Ciência da Computação para realizar um bom trabalho.

9 Referências

ModSecurity Project. *ModSecurity: Open Source Web Application Firewall*. 2024. Accessed: 2024-12-22. Disponível em: <<https://modsecurity.org/>><https://modsecurity.org/>. Citado na página 4.

NGINX. *NGINX: High Performance Load Balancer, Web Server, Reverse Proxy*. 2024. Accessed: 2024-12-22. Disponível em: <<https://nginx.org/en/>><https://nginx.org/en/>. Citado na página 4.

OWASP, C. R. S. *Core Rule Set*. 2024. Accessed: 2024-12-21. Disponível em: <<https://github.com/coreruleset/coreruleset>><https://github.com/coreruleset/coreruleset>. Citado na página 5.

Wazuh, Inc. *Wazuh Dashboard Installation Guide: Step by Step*. 2024. Accessed: 2024-12-22. Disponível em: <<https://documentation.wazuh.com/current/installation-guide/wazuh-dashboard/step-by-step.html>><https://documentation.wazuh.com/current/installation-guide/wazuh-dashboard/step-by-step.html>. Citado na página 9.

Wazuh, Inc. *Wazuh Indexer Installation Guide: Step by Step*. 2024. Accessed: 2024-12-22. Disponível em: <<https://documentation.wazuh.com/current/installation-guide/wazuh-indexer/step-by-step.html>><https://documentation.wazuh.com/current/installation-guide/wazuh-indexer/step-by-step.html>. Citado na página 8.

Wazuh, Inc. *Wazuh Server Installation Guide: Step by Step*. 2024. Accessed: 2024-12-22. Disponível em: <<https://documentation.wazuh.com/current/installation-guide/wazuh-server/step-by-step.html>><https://documentation.wazuh.com/current/installation-guide/wazuh-server/step-by-step.html>. Citado na página 8.

Wazuh, Inc. *Wazuh: The Open Source Security Platform*. 2024. Accessed: 2024-12-22. Disponível em: <<https://wazuh.com/>><https://wazuh.com/>. Citado na página 7.

Wazuh Team. *Wazuh Documentation*. [S.l.], 2024. Accessed: 2024-12-21. Disponível em: <<https://documentation.wazuh.com/current/getting-started/index.html>><https://documentation.wazuh.com/current/getting-started/index.html>. Citado na página 8.

XIAO, G. *How to Enable ModSecurity with Nginx on Debian/Ubuntu*. 2022. Accessed: 2024-12-22. Disponível em: <<https://www.linuxbabe.com/security/modsecurity-nginx-debian-ubuntu>><https://www.linuxbabe.com/security/modsecurity-nginx-debian-ubuntu>. Citado na página 4.

Anexo A - Configuração de rede das máquinas

No caso da VM do proxy, a interface `enp0s3` está conectada à rede NAT com IP fixo `10.0.10.11`, com rota padrão configurada para o IP `10.0.10.1`, que é o IP atribuído pelo VirtualBox ao *gateway* da rede. Cada uma das demais interfaces conecta a uma rede diferente, a de produção, do Honeypot e a de monitoramento e gestão.

```
1 network:
2   version: 2
3   ethernets:
4     enp0s3: # rede NAT
5       dhcp4: no
6       addresses:
7         - 10.0.10.11/24
8       routes:
9         - to: default
10         via: 10.0.10.1
11       nameservers:
12         addresses:
13           - 8.8.8.8
14           - 8.8.4.4
15
16     enp0s8: # rede prod
17       dhcp4: no
18       addresses:
19         - 10.0.100.10/24
20
21     enp0s9: # rede honey
22       dhcp4: no
23       addresses:
24         - 10.0.101.10/24
25
26     enp0s10: # rede mgmt
27       dhcp4: no
28       addresses:
29         - 10.0.102.10/24
```

Arquivo 3 – Configuração das interfaces do proxy

Para efetivar as mudanças, deve-se usar o comando `netplan apply`. Abaixo estão as configurações das demais máquinas.

```
1 network:
2   version: 2
3   ethernets:
4     enp0s8: # rede prod
5       dhcp4: no
6       addresses:
7         - 10.0.100.20/24
```



```
8
9     enp0s9: # rede mgmt
10         dhcp4: no
11         addresses:
12             - 10.0.102.20/24
```

Arquivo 4 – Configuração das interfaces da VM do servidor legítimo

```
1 network:
2     version: 2
3     ethernets:
4         enp0s8: # rede honey
5             dhcp4: no
6             addresses:
7                 - 10.0.101.30/24
8
9         enp0s9: # rede mgmt
10             dhcp4: no
11             addresses:
12                 - 10.0.102.30/24
```

Arquivo 5 – Configuração das interfaces da VM do Honeypot

```
1 network:
2     version: 2
3     ethernets:
4         enp0s8: # rede mgmt
5             dhcp4: no
6             addresses:
7                 - 10.0.102.40/24
```

Arquivo 6 – Configuração das interfaces do SIEM

Anexo B - Configuração do Nginx e ModSecurity

```

1 server {
2     listen 443 ssl default_server;
3     server_name _;
4
5     ssl_certificate      /etc/nginx/ssl/certificate.crt;
6     ssl_certificate_key  /etc/nginx/ssl/certificate.key;
7
8     log_format custom_combined '$remote_addr - $remote_user [$time_local
9 ] '
10                                '"$request" $status $body_bytes_sent '
11                                '"$http_referer" "$http_user_agent"';
12
13     access_log /var/log/nginx/access.log custom_combined;
14     error_log /var/log/nginx/error.log warn;
15
16     error_page 403 = @Honeypot;
17
18     location @Honeypot {
19         proxy_pass https://10.0.101.30;
20     }
21
22     location / {
23         proxy_pass https://10.0.100.20;
24         proxy_ssl_verify off;
25     }
26 }

```

Arquivo 7 – Configuração do proxy Reverso

```

1 # /etc/nginx/modsec/coreruleset-4.9.0/rules/REQUEST-949-BLOCKING-
2   EVALUATION.conf
3 # if early blocking is active, check threshold in phase 1
4 SecRule TX:BLOCKING_INBOUND_ANOMALY_SCORE "@ge %{tx.
5 inbound_anomaly_score_threshold}" \
6     "id:949111,\
7     phase:1,\
8     deny,\
9     t:none,\
10    msg:'Inbound Anomaly Score Exceeded in phase 1 (Total Score: %{TX.
11 BLOCKING_INBOUND_ANOMALY_SCORE})',\
12    tag:'anomaly-evaluation',\
13    tag:'OWASP_CRS',\
14    ver:'OWASP_CRS/4.9.0',\
15    chain"
16     SecRule TX:EARLY_BLOCKING "@eq 1"
17
18 # always check threshold in phase 2

```

```

16 SecRule TX:BLOCKING_INBOUND_ANOMALY_SCORE "@ge %{tx.
    inbound_anomaly_score_threshold}" \
17     "id:949110,\
18     phase:2,\
19     deny,\
20     t:none,\
21     msg:'Inbound Anomaly Score Exceeded (Total Score: %{TX.
BLOCKING_INBOUND_ANOMALY_SCORE})',\
22     tag:'anomaly-evaluation',\
23     tag:'OWASP_CRS',\
24     ver:'OWASP_CRS/4.9.0'"

```

Arquivo 8 – Regra de bloqueio a partir de limiar padrão do CRS

```

1 # /etc/nginx/modsec/coreruleset-4.9.0/rules/REQUEST-999-CUSTOM.conf
2 SecRuleUpdateActionById 949110 "setvar:ip.invasor=1"
3 SecRuleUpdateActionById 949111 "setvar:ip.invasor=1"

```

Arquivo 9 – Arquivo adicionado para salvar IP que passou do limiar de malícia como invasor.

```

1 # /etc/nginx/modsec/coreruleset-4.9.0/rules/RESPONSE-999-CUSTOM.conf
2 SecRule RESPONSE_STATUS "@streq 401" \
3     "id:999021,\
4     phase:4,\
5     pass,\
6     t:none,\
7     nolog,\
8     setvar:ip.login_falhas+=1,\
9     msg:'[Custom] Falha de login detectada (401) => +1 login_falhas'"
10
11 SecRule IP:login_falhas "@ge 5" \
12     "id:999022,\
13     phase:2,\
14     pass,\
15     t:none,\
16     setvar:ip.invasor=1,\
17     msg:'[Custom] IP marcado como invasor por 5 falhas de login'"
18
19 SecRule IP:invasor "@eq 1" \
20     "id:999023,\
21     phase:2,\
22     deny,\
23     status:403,\
24     t:none,\
25     msg:'[Custom] IP invasor => 403 => Honeypot'"

```

Arquivo 10 – Regras para contar tentativas de login e bloquear a partir de 5 seguidas.

Neste caso, se a resposta da aplicação for 401 (Login Inválido), incrementamos o contador de falhas para esse IP que tentou fazer o login. Caso esse IP chegue à somatória

de 5 falhas, ele é identificado como invasor e passa a ter o acesso negado à aplicação real, sendo interceptado pelo Nginx e redirecionado para o Honeypot.

Anexo C - Configurações dos contêineres de produção

```

1 version: '3.8'
2
3 services:
4   apache-php:
5     container_name: apache_php
6     build: ./apache-php
7     ports:
8       - "80:80"
9       - "443:443"
10    volumes:
11      - ./apache-php/src:/var/www/html
12      - ./apache-php/logs:/var/logs/apache2
13    depends_on:
14      - mysql
15    restart: unless-stopped
16    networks:
17      - servers_net
18
19    mysql:
20      container_name: mysql_server
21      image: mysql:8.0
22      environment:
23        MYSQL_ROOT_PASSWORD: "root"
24        MYSQL_DATABASE: "secure_db"
25        MYSQL_USER: "user"
26        MYSQL_PASSWORD: "user"
27      volumes:
28        - mysql_data:/var/lib/mysql
29        - ./mysql-init:/docker-entrypoint-initdb.d
30        - ./mysql-logs:/var/log/mysql
31      ports:
32        - "3306:3306"
33      restart: unless-stopped
34      networks:
35        - servers_net
36      command: --local-infile=1 --secure-file-priv=/var/lib/mysql-files
37
38 volumes:
39   mysql_data:
40
41 networks:
42   servers_net:
43     driver: bridge

```

Arquivo 11 – docker-compose.yml

```

1 FROM php:8.1-apache

```

```
2
3 RUN apt-get update && apt-get install -y openssl \
4     && docker-php-ext-install mysqli pdo_mysql
5
6 COPY ssl/server.crt /etc/ssl/certs/server.crt
7 COPY ssl/server.key /etc/ssl/private/server.key
8 COPY ssl/000-default.conf /etc/apache2/sites-available/000-default.conf
9 COPY ssl/000-default-ssl.conf /etc/apache2/sites-available/000-default-
    ssl.conf
10
11 RUN a2enmod ssl rewrite
12 RUN a2ensite 000-default
13 RUN a2ensite 000-default-ssl
14
15 WORKDIR /var/www/htmlSD
```

Arquivo 12 – apache-php/Dockerfile

Os demais arquivos utilizados para a criação da aplicação podem ser visualizados no [GitHub](#)

Anexo D - Log SIEM de tentativa de login incorreto

```
1 {
2   "_index": "wazuh-alerts-4.x-2024.12.22",
3   "_id": "NHqI75MBcXYNosmW3Xh0",
4   "_score": 1,
5   "_source": {
6     "agent": {
7       "ip": "10.0.102.10",
8       "name": "proxy",
9       "id": "001"
10    },
11    "manager": {
12      "name": "siem"
13    },
14    "data": {
15      "protocol": "POST",
16      "srcip": "10.0.10.6",
17      "id": "401",
18      "url": "/index.php"
19    },
20    "rule": {
21      "firedtimes": 1,
22      "mail": false,
23      "level": 5,
24      "pci_dss": [
25        "6.5",
26        "11.4"
27      ],
28      "tsc": [
29        "CC6.6",
30        "CC7.1",
31        "CC8.1",
32        "CC6.1",
33        "CC6.8",
34        "CC7.2",
35        "CC7.3"
36      ],
37      "description": "Web server 400 error code.",
38      "groups": [
39        "web",
40        "accesslog",
41        "attack"
42      ],
43      "id": "31101",
44      "nist_800_53": [
45        "SA.11",
46        "SI.4"
```

```
47     ],
48     "gdpr": [
49         "IV_35.7.d"
50     ]
51 },
52 "decoder": {
53     "name": "web-accesslog"
54 },
55 "full_log": "10.0.10.6 - - [22/Dec/2024:18:02:13 +0000] \"POST /
index.php HTTP/1.1\" 401 629 \"https://10.0.10.11/index.php\" \"
Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox
/115.0\"",
56 "input": {
57     "type": "log"
58 },
59 "@timestamp": "2024-12-22T18:02:13.538Z",
60 "location": "/var/log/nginx/access.log",
61 "id": "1734890533.1546403",
62 "timestamp": "2024-12-22T18:02:13.538+0000"
63 },
64 "fields": {
65     "@timestamp": [
66         "2024-12-22T18:02:13.538Z"
67     ],
68     "timestamp": [
69         "2024-12-22T18:02:13.538Z"
70     ]
71 }
72 }
```


Anexo E - Log SIEM descoberta de invasor via brute force

```
1 {
2   "_index": "wazuh-alerts-4.x-2024.12.22",
3   "_id": "Q3qZ75MBcXYNosmWDnge",
4   "_version": 1,
5   "_score": null,
6   "_source": {
7     "input": {
8       "type": "log"
9     },
10    "agent": {
11      "ip": "10.0.102.10",
12      "name": "proxy",
13      "id": "001"
14    },
15    "manager": {
16      "name": "siem"
17    },
18    "data": {
19      "srcip": "10.0.10.6"
20    },
21    "rule": {
22      "firedtimes": 1,
23      "mail": false,
24      "level": 7,
25      "description": "ModSecurity rejected a query",
26      "groups": [
27        "nginx",
28        "web",
29        "modsecurity"
30      ],
31      "mitre": {
32        "technique": [
33          "File and Directory Discovery"
34        ],
35        "id": [
36          "T1083"
37        ],
38        "tactic": [
39          "Discovery"
40        ]
41      },
42      "id": "31333",
43      "gpg13": [
44        "10.1"
45      ]
46    },

```

```
47     "location": "/var/log/nginx/error.log",
48     "decoder": {
49         "parent": "nginx-errorlog",
50         "name": "nginx-errorlog"
51     },
52     "id": "1734891602.1548455",
53     "full_log": "2024/12/22 18:20:02 [error] 1834#1834: *52 [client
10.0.10.6] ModSecurity: Access denied with code 403 (phase 2).
Matched \"0operator 'Eq' with parameter '1' against variable 'IP
:10.0.10.6_ef5e07785cb550c27cafab8bc95a0c4c7556def1::::invasor' (
Value: '1' ) [file \"/etc/nginx/modsec/coreruleset-4.9.0/rules/
RESPONSE-999-CUSTOM.conf\"] [line \"27\"] [id \"999023\"] [rev \"\"]
[msg \"[Custom] IP invasor => 403 => honeypot\"] [data \"\"] [
severity \"0\"] [ver \"\"] [maturity \"0\"] [accuracy \"0\"] [
hostname \"10.0.10.11\"] [uri \"/index.php\"] [unique_id
\"173489160274.219452\"] [ref \"\"], client: 10.0.10.6, server: _,
request: \"POST /index.php HTTP/1.1\", host: \"10.0.10.11\", referrer
: \"https://10.0.10.11/index.php\"\",
54     "timestamp": "2024-12-22T18:20:02.818+0000"
55 },
56 "fields": {
57     "timestamp": [
58         "2024-12-22T18:20:02.818Z"
59     ]
60 },
61 "sort": [
62     1734891602818
63 ]
64 }
```

Anexo F - Log SIEM SQL Injection

```
1 {
2   "_index": "wazuh-alerts-4.x-2024.12.22",
3   "_id": "TXqd75MBcXYNosmWLHj6",
4   "_version": 1,
5   "_score": null,
6   "_source": {
7     "input": {
8       "type": "log"
9     },
10    "agent": {
11      "ip": "10.0.102.10",
12      "name": "proxy",
13      "id": "001"
14    },
15    "manager": {
16      "name": "siem"
17    },
18    "data": {
19      "srcip": "10.0.10.6"
20    },
21    "rule": {
22      "firedtimes": 10,
23      "mail": false,
24      "level": 7,
25      "description": "ModSecurity rejected a query",
26      "groups": [
27        "nginx",
28        "web",
29        "modsecurity"
30      ],
31      "mitre": {
32        "technique": [
33          "File and Directory Discovery"
34        ],
35        "id": [
36          "T1083"
37        ],
38        "tactic": [
39          "Discovery"
40        ]
41      },
42      "id": "31333",
43      "gpg13": [
44        "10.1"
45      ]
46    },

```

```

47     "location": "/var/log/nginx/error.log",
48     "decoder": {
49         "parent": "nginx-errorlog",
50         "name": "nginx-errorlog"
51     },
52     "id": "1734891867.1557326",
53     "full_log": "2024/12/22 18:24:26 [error] 1834#1834: *57 [client
10.0.10.6] ModSecurity: Access denied with code 403 (phase 2).
Matched \"0operator 'Eq' with parameter '1' against variable 'IP
:10.0.10.6_ef5e07785cb550c27cafab8bc95a0c4c7556def1:::invasor' (
Value: '1' ) [file \"/etc/nginx/modsec/coreruleset-4.9.0/rules/
RESPONSE-999-CUSTOM.conf\" [line \"27\"] [id \"999023\"] [rev \"\"]
[msg \"[Custom] IP invasor => 403 => honeypot\"] [data \"\"] [
severity \"0\"] [ver \"\"] [maturity \"0\"] [accuracy \"0\"] [
hostname \"10.0.10.11\"] [uri \"/dex.php\"] [unique_id
\"173489186669.664331\"] [ref \"\"], client: 10.0.10.6, server: _,
request: \"GET /dex.php?search=1%3D1+UNION+SELECT+1%2C+username%2C+
password%2C+NULL%2C+NULL%2C+NULL%2C+NULL%2C+NULL%2C+NULL%2C+
NULL%2C+NULL%2C+NULL%2C+NULL%2C+NULL%2C+NULL%2C+NULL+FROM+users+--
HTTP/1.1\", host: \"10.0.10.11\", referer: \"https://10.0.10.11/dex.
php\"\",
54     "timestamp": "2024-12-22T18:24:27.213+0000"
55 },
56 "fields": {
57     "timestamp": [
58         "2024-12-22T18:24:27.213Z"
59     ]
60 },
61 "sort": [
62     1734891867213
63 ]
64 }

```