UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS REITOR JOÃO DAVID FERREIRA LIMA
PROGRAMA DE GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Yuri Ferreira Bittencourt

ORCH-IDS - UM SISTEMA MODULAR DE DETECÇÃO DE INTRUSOS NA REDE BASEADO EM REGRAS:

Suportando multiagentes coletores e analisadores, regras customizáveis e flexíveis

Florianópolis, Santa Catarina – Brasil
2023

Yuri Ferreira Bittencourt

ORCH-IDS - UM SISTEMA MODULAR DE DETECÇÃO DE INTRUSOS NA REDE BASEADO EM REGRAS:

Suportando multiagentes coletores e analisadores, regras customizáveis e flexíveis

Trabalho de Conclusão de Curso submetido ao Programa de Graduação em Ciências da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientador(a): Jean Everson Martina, Dr.

Florianópolis, Santa Catarina – Brasil
2023

Notas legais:

Não há garantia para qualquer parte do software documentado. Os autores tomaram cuidado na preparação desta tese, mas não fazem nenhuma garantia expressa ou implícita de qualquer tipo e não assumem qualquer responsabilidade por erros ou omissões. Não se assume qualquer responsabilidade por danos incidentais ou consequentes em conexão ou decorrentes do uso das informações ou programas aqui contidos.

Catalogação na fonte pela Biblioteca Universitária da Universidade Federal de Santa Catarina.
Arquivo compilado às 02:38h do dia 12 de julho de 2023.

Yuri Ferreira Bittencourt
ORCH-IDS - Um sistema modular de Detecção de Intrusos na rede baseado em regras : Suportando multiagentes coletores e analisadores, regras customizáveis e flexíveis / Yuri Ferreira Bittencourt; Orientador(a), Jean Everson Martina, Dr. – Florianópolis, Santa Catarina – Brasil, 05 de julho de 2023.
114 p.

Trabalho de Conclusão de Curso – Universidade Federal de Santa Catarina, INE – Departamento de Informática e Estatística, CTC – Centro Tecnológico, Programa de Graduação em Ciências da Computação.

Inclui referências

1. Segurança de Redes, 2. Sistema de detecção de intrusão na rede baseado em regras, 3. IDS, 4. NIDS, I. Jean Everson Martina, Dr. II. Programa de Graduação em Ciências da Computação III. ORCH-IDS - Um sistema modular de Detecção de Intrusos na rede baseado em regras

CDU 02:141:005.7

Yuri Ferreira Bittencourt

ORCH-IDS - UM SISTEMA MODULAR DE DETECÇÃO DE INTRUSOS NA REDE BASEADO EM REGRAS: Suportando multiagentes coletores e analisadores, regras customizáveis e flexíveis

Este(a) Trabalho de Conclusão de Curso foi julgado adequado(a) para obtenção do Título de Bacharel em Ciências da Computação, e foi aprovado em sua forma final pelo Programa de Graduação em Ciências da Computação do INE – Departamento de Informática e Estatística, CTC – Centro Tecnológico da Universidade Federal de Santa Catarina.

Florianópolis, Santa Catarina – Brasil, 05 de julho de 2023.

Lúcia Helena Martins Pacheco, Dr.
Coordenador(a) do Programa de
Graduação em Ciências da Computação

Banca Examinadora:

Jean Everson Martina, Dr.
Orientador(a)
Universidade Federal de Santa
Catarina – UFSC

Guilherme Arthur Geronimo, Dr.
Universidade Federal de Santa
Catarina – UFSC

Henrique Ribeiro
Universidade Federal de Santa
Catarina – UFSC

Este trabalho é dedicado à todos que acreditaram em mim,
que apesar das minhas dificuldades, não me permitiram desistir.

AGRADECIMENTOS

Os agradecimentos principais são direcionados aos meus pais por me guiarem até aqui, aos meus irmãos, meus tios e primos em Florianópolis, meus professores que me inspiraram tanto na PUCRS quanto na UFSC e também aos meus amigos que fiz antes e durante esta jornada.

Agradeço também à toda a comunidade científica que se não fossem por eles, não estaria neste ponto e espero que assim como aqueles que vieram antes me conduziram, eu faça o mesmo aos próximos.

“Somos como anões aos ombros de gigantes, pois podemos ver mais coisas do que eles e mais distantes, não devido à acuidade da nossa vista ou à altura do nosso corpo, mas porque somos mantidos e elevados pela estatura de gigantes.”
Bernardus Carnotensis

“Nós só podemos ver um pouco do futuro, mas o suficiente para perceber que há muito a fazer.”
Alan Turing

RESUMO

A expansão e popularização da Internet trouxeram benefícios significativos à sociedade, mas também aumentaram a incidência de ataques cibernéticos. Entre os tipos mais comuns de ataques segundo a CrowdStrike (BAKER, 2023), destaca-se o ataque de Negação de Serviço (DoS), incluindo a variante distribuída (DDoS). Um sistema de detecção de intrusão na rede baseado em regras pode detectar esses ataques. O objetivo deste trabalho é desenvolver um protótipo escalável de um sistema de detecção de intrusos na rede baseado em regras, capaz de lidar com múltiplos sensores de captura de pacotes de rede e fornecer uma interface web para configuração e visualização de dados e alertas gerados pelo sistema.

Palavras-chaves: Segurança de Redes. Sistema de detecção de intrusão na rede baseado em regras. IDS. NIDS.

ABSTRACT

The expansion and popularization of the Internet have brought many benefits to our society, however, it has become commonplace for cyber-attacks. according to CrowdStrike(BAKER, 2023), the second most common cyber-attack is the Denial of Service attack, also known as DoS, and that includes its distributed variant, DDoS, and this is merely an example of an attack that can be detected by a rule-based network intrusion detection system. The main objective of this study is to develop a scalable prototype of a rule-based network intrusion detection system that supports multiple network packet-capture sensors and has a web interface for system configurations and data visualization of generated alerts.

Keywords: Network Security. Rule-based Network Intrusion Detection System. IDS. NIDS.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

IDS	*Intrusion Detection System*
NIDS	*Network-based Intrusion Detection System*
TCC	Trabalho de Conclusão de Curso
IP	*Internet Protocol*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
DoS	*Denial of Service*
DDoS	*Distributed Denial of Service*

SUMÁRIO

1 INTRODUÇÃO

Os ataques e tentativas de ataques cibernéticos estão cada vez mais frequentes e especificamente no Brasil em 2022 houve um aumento de 94% dos incidentes comparando o primeiro semestre de 2022 (31,5bi de incidentes) com o mesmo período no ano anterior (16,2bi de incidentes) segundo a CNN Brasil(OLIVEIRA, 2022) e é esperado que com o tempo os incidentes de cibersegurança tendam a aumentar.

No começo do ano de 2023, a CrowdStrike compilou um documento (BAKER, 2023) relatando os 10 tipos de ciberataques mais comuns, nesta listagem os ataques de Negação de serviço (DoS) e Negação de serviço distribuída (DDoS) figuram em segundo lugar e a importância de mencioná-los é que esse tipo de ataque pode ser detectado por regras configuradas em um IDS, nessa lista também há outros tipos de ataques passíveis dessa análise.

É evidente que, com base nas informações previamente apresentadas, existe a necessidade de melhorar e ampliar os sistemas de segurança cibernética e isso vai de encontro com a proposta deste trabalho que apresentará um protótipo (prova de conceito) de um sistema de detecção de intrusão na rede distribuído baseado em regras com múltiplos agentes sendo altamente customizável e com interface web.

1.1 JUSTIFICATIVA

Buscando implementar um sistema de detecção de intrusão na rede baseado em regras de forma descomplicada, com múltiplos sensores e com boa escalabilidade (processamento distribuído) foi percebido uma lacuna devido a sistemas similares como o Snort (SNORT, 2023) que permite múltiplos sensores, ainda que a configuração dos mesmos não seja tão simples, tem o processamento totalmente centralizado e isso permite a realização deste projeto como prova de conceito. Como mencionado na seção anterior deste mesmo capítulo existe uma crescente demanda para ampliação e melhoramento de sistemas de segurança cibernética e o protótipo desenvolvido neste trabalho pode ser útil neste objetivo.

Este trabalho também é motivado por interesses pessoais e é esperado que o sistema desenvolvido neste trabalho possa ampliar o conhecimento deste tipo de tecnologia.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O Objetivo geral deste trabalho é o desenvolvimento de um protótipo de sistema de detecção de intrusão na rede baseado em regras, assim como o Snort (SNORT, 2023) com múltiplos sensores, modularizado com suporte a interface web e que possa

ser facilmente implantado e altamente escalável afim de que possa atuar detectando possíveis ataques em uma rede e sendo possível configurar suas regras de detecção e IPs a serem monitorados.

1.2.2 Objetivos Específicos

Os Objetivos específicos deste trabalho incluem características do desenvolvimento da ferramenta como: independência entre os módulos de forma que o sistema possa funcionar independentemente da quantidade de agentes de sensoreamento como também de unidades de processamento de regras, o protótipo deve estar funcional e suportando a criação de regras simples e complexas que são regras que envolvem um período e número de casos para gerar alertas.

1.3 ESTRUTURA DOS CAPÍTULOS

Os capítulos deste trabalho estão dispostos da seguinte forma:

- Capítulo 2: Discorre sobre a fundamentação teórica dos Sistemas de Detecção de Intrusão (IDS), dando foco nos sistemas que operam na rede (NIDS) e são baseados em regras;
- Capítulo 3: Apresenta trabalhos correlatos;
- Capítulo 4: Introduz o sistema desenvolvido neste trabalho, implementação, funcionalidades e limitações;
- Capítulo 5: Apresenta as considerações finais e possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados conceitos e sistemas necessários para a compreensão deste trabalho e seu sistema proposto.

2.1 SISTEMA DE DETECÇÃO DE INTRUSÃO

O Sistema de detecção de Intrusão (do inglês: *Intrusion Detection System*) é uma tecnologia de segurança em rede pensada em detectar possíveis ataques contra uma aplicação ou computador alvo. Diferentemente de um sistema de prevenção, o IDS apenas detecta e alerta, funcionando assim de forma passiva apenas monitorando tráfego e reportando dados a um administrador ou sistema, segundo (PALO ALTO NETWORKS, 2023)

2.2 SISTEMA DE DETECÇÃO DE INTRUSÃO BASEADO EM REDE

Esta variação do IDS, também conhecido como NIDS, consiste em um sistema que diferentemente do IDS que era voltado a uma aplicação ou computador específico é direcionado operar sobre a rede e que irá monitorar o seu tráfego, conforme (STALLINGS, 2012, Capítulo 8.5), neste mesmo capítulo é citado o funcionamento típico de um NIDS consistindo em:

- vários sensores para capturar pacotes de rede;
- um ou mais servidores para funções de gerenciamento do sistema;
- um ou mais consoles de gerenciamento com interface para o operador.

Ainda neste mesmo assunto, é importante mencionar sobre os tipos de sensores que podem ser *Inline* ou passivos e conforme (STALLINGS, 2012, Capítulo 8.5): o *inline* é um sensor que está interceptando o tráfego e pelo seu funcionamento pode ser usado para prevenção de ataques, entretanto ele pode causar latência na rede; O tipo passivo é mais comumente usado para IDS pois não afeta a rede diretamente uma vez que ele apenas monitora uma cópia do tráfego e não intercepta pacotes.

2.3 SISTEMA DE DETECÇÃO DE INTRUSÃO BASEADO EM REGRAS

Existem várias formas para um IDS detectar algum evento, a forma interessante para este trabalho é a detecção baseada em regras assim como funciona o sistema Snort (SNORT, 2023) onde segundo a própria documentação do Snort (MANSOUR, 2020) as regras seguem o formato:

O primeiro campo *action* seria qual a ação para ser tomada e isso é interessante no contexto de um sistema de prevenção de Intrusão (IPS), em nosso caso todas as

Quadro 1 – Formato de regras do snort

Action Protocol Networks Ports Direction Operator Networks Ports

Fonte: (MANSOUR, 2020)

ações do sistema proposto seriam equivalentes a *action alert*, os demais campos em tradução aberta são:

- *Protocol*: Protocolo do remetente;
- *Networks*: Rede do remetente, este seria o endereço;
- *Ports*: Porta do remetente;
- *Direction Operator*: Direção da regra, unidirecional ou bidirecional;
- *Networks*: Rede do destinatário, endereço do destinatário;
- *Ports*: Porta do destinatário

Perceba que uma regra bidirecional perde um pouco do sentido de remetente e destinatário mas foi mantido na explicação para diferenciar os campos, contudo uma regra bidirecional pode ser vista como duas regras unidirecionais com os campos *Networks* e *Ports* invertidos

3 TRABALHOS RELACIONADOS

Neste capítulo serão apresentados trabalhos com propostas similares a este trabalho e também o diferencial de cada um e o motivo de não ser aplicável ao caso deste trabalho.

Existem trabalhos que apresentam a integração entre IDS e a arquitetura SDN, o autor (FRANCISCO, 2019) propõe um estudo sobre a arquitetura SDN com o protocolo OpenFlow para integrar com um sistema de detecção de intrusão apresentando um ambiente simulado integrando essas tecnologias para mitigação de ataques, não apresenta um IDS propriamente mas uma integração com, para efeitos de teste e validação num ambiente controlado este trabalho é adequado.

O próprio projeto do Snort (SNORT, 2023) é um trabalho correlato por ser um sistema de detecção e prevenção de intrusão na rede baseado em regras *open source* e foi uma forte inspiração para este trabalho, o Snort tem vários modos de operação e isso permite ele atuar como um IPS ou IDS dependendo das configurações, o Snort permite múltiplos sensores mas no entanto o seu processamento é sempre centralizado.

No trabalho (KUMAR *et al.*, 2020) foi apresentado um NIDS baseado em regras e o sistema proposto utiliza uma abordagem de detecção de assinatura para identificar diferentes tipos de ameaças em uma rede. Este trabalho também fez um comparativo de desempenho com outros modelos existentes, entretanto aqui o IDS proposto possui processamento centralizado.

O trabalho (JÚNIOR, 2016) é um trabalho fortemente relacionado à este, a proposta é similar e caso tivesse o encontrado antes seria base para este trabalho pois apresenta uma estrutura similar, multi-agentes coletores e processamento podendo ser distribuído. Também é um IDS em rede baseado em regras.

4 DESENVOLVIMENTO DO ORCH-IDS

Neste capítulo será apresentado os requisitos funcionais e não-funcionais, arquitetura do sistema, esquema do banco de dados, e funcionamento.

4.1 REQUISITOS FUNCIONAIS

1. Deve poder capturar os pacotes de rede e armazená-los para posterior análise;
2. Deve aceitar a criação e exclusão de regras customizáveis que analisarão os pacotes pois sem as regras o sistema não poderá detectar ameaças;
3. Deve aceitar a inclusão e exclusão de endereços IP na lista de monitoramento para detectar ameaças de endereços suspeitos conhecidos;
4. Deve analisar os pacotes com todas as regras aplicáveis se possível, uma vez que o atraso no processamento pode gerar detecções muito tardias;
5. Deve gerar alerta caso uma regra detecte uma ameaça, essa é a função principal de um IDS;
6. Deve ser possível consultar e apagar estes alertas, uma vez que alguns alertas podem ser ignorados ou ultrapassados;

4.2 REQUISITOS NÃO-FUNCIONAIS

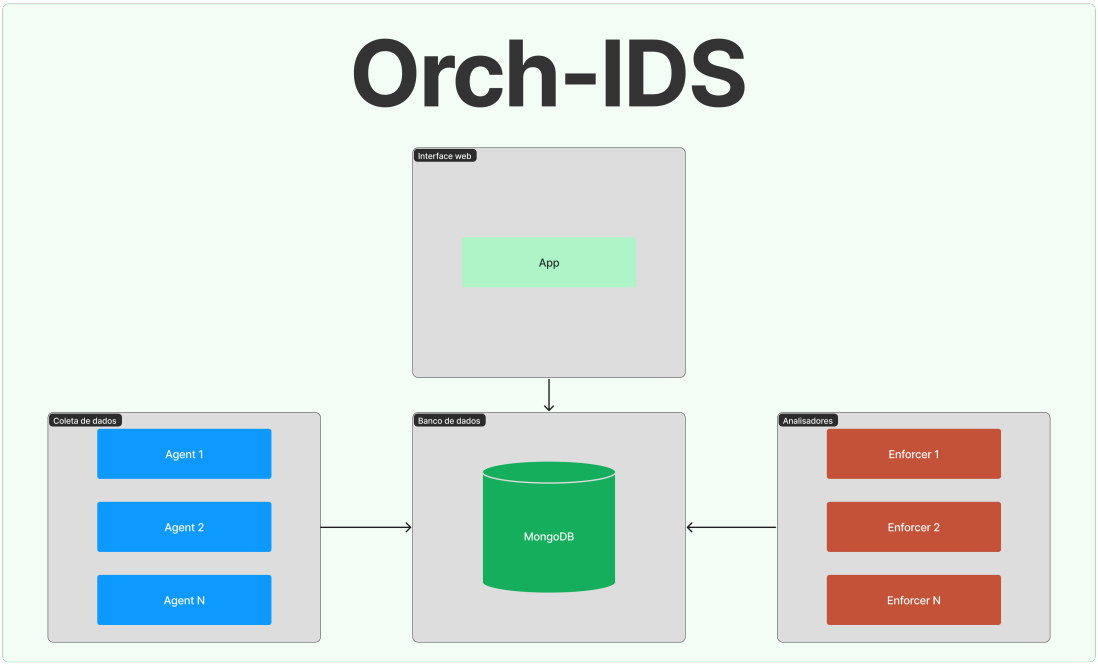
1. Deve armazenar os pacotes com pouco atraso, uma vez que a quantidade de pacotes de rede pode ser exageradamente grande;
2. Deve analisar os pacotes de forma eficiente e rápida, afim de evitar atrasos entre a chegada de um pacote e sua análise;
3. Deve ter regras de exemplo para auxiliar novos operadores.

4.3 ARQUITETURA

O sistema é dividido em 3 serviços que se comunicam apenas com uma instância ou cluster do MongoDB, dessa forma qualquer parte do sistema pode ser substituída ou refeita desde que respeitada as entradas e saídas desse serviço para com o banco.

Perceba que na figura 1 há múltiplos *Agents* e *Enforcers*, isso é só um exemplo de que o sistema aceita quantas instâncias desses serviços forem desejadas, o mesmo vale para o *App* apesar de não haver muita utilidade em mais de uma instância para a interface gráfica.

Figura 1 – Arquitetura do Sistema ORCH-IDS



Fonte: O autor.

4.3.1 *Agent*

Este serviço é responsável pela coleta de dados para o sistema, funciona como um *sniffer*, esse sistema só captura pacotes encapsulado por IP (versão 4 ou 6) por motivos de simplificação para prototipação, envia esses dados em grupos de 10 para a coleção de pacotes no banco de dados, a configuração desse serviço é feita no arquivo *.env* que tem o seguinte formato:

Código 1 – *.env* do *Agent*

```
1  MONGO_HOST='mongodb://127.0.0.1'
2  MONGO_PORT=27017
3  MONGO_DB='NIDS'
4  MONGO_COLLECTION\_QUEUE='packets'
5  NETWORK_INTERFACE='Ethernet'
```

Este é um arquivo de exemplo que basicamente tem o endereço do banco e a porta, nome da *database*, coleção do banco e principalmente qual a interface de rede que deve ser usada para monitorar o tráfego, importante salientar que esta interface deve ser configurada em modo promíscuo caso queira monitorar todo tráfego da rede ou não caso queira apenas monitorar o tráfego para esta máquina.

4.3.2 *App*

Este serviço é responsável por prover uma interface web para o sistema desenvolvido em *Flask*, aqui também estão os utilitários que criam o banco de dados, índices e populam ou esvaziam as coleções, assim como o *Agent*, também possui um arquivo de configuração chamado *.env*, no entanto ele possui mais informações:

Código 2 – *.env* do *App*

```
1   MONGO_HOST='mongodb://127.0.0.1'
2   MONGO_PORT=27017
3   MONGO_DB='NIDS'
4   MONGO_COLLECTION_QUEUE='packets'
5   MONGO_COLLECTION_RULES='rules'
6   MONGO_COLLECTION_BLACKLISTED='blacklist'
7   MONGO_COLLECTION_ALERTS='alerts'
8   MONGO_COLLECTION_OCCURRENCES='occurrences'
9   ADMIN_USER='admin'
10  ADMIN_PASSWORD='admin'
11  ENVIRONMENT='DEV'
12  PORT='5000'
13  SECRET_KEY='super-secret-key'
```

Assim como no código 1 há a configuração de conexão ao banco de dados, aqui também temos todas as coleções, nome e senha de usuário da aplicação, porta, chave de encriptação e principalmente o ambiente (*DEV* ou *PROD*)

4.3.3 *Enforcer*

Este serviço é responsável por pegar dados da coleção de pacotes (que funciona como fila) e verificar se esses pacotes violam alguma regra ou o IP (destino ou origem) desses pacotes constam na lista negra, gera alertas em casos positivos.

A cada 5 minutos este serviço acessa o banco de dados para verificar se houveram alterações no conjunto de regras ou conjunto de IPs monitorados (lista negra) e a cada 1 minuto existe uma verificação na coleção de *occurrences* que testa por regras 'complexas' que são regras que só geram alertas se ocorrerem *x* vezes num intervalo de *y* segundos.

Assim como os outros serviços, as configurações desse serviço ficam em um arquivo *.env*:

Código 3 – *.env* do *Enforcer*

```
1   MONGO_HOST='mongodb://127.0.0.1'
2   MONGO_PORT=27017
3   MONGO_DB='NIDS'
```

```
4       MONGO_COLLECTION_QUEUE='packets'
5       MONGO_COLLECTION_RULES='rules'
6       MONGO_COLLECTION_BLACKLISTED='blacklist'
7       MONGO_COLLECTION_ALERTS='alerts'
8       MONGO_COLLECTION_OCCURRENCES='occurrences'
```

Assim como no código 1 e 2, há a configuração de conexão ao banco de dados e as coleções que o *Enforcer* utiliza (todas).

4.4 ESQUEMA DO BANCO DE DADOS

4.4.1 *Rules*

Esta coleção guardará todas as regras do sistema, o esquema da coleção é:

- name: nome da regra, é uma string obrigatória que não repete;
- description: descrição da regra, é uma string obrigatória;
- severity: severidade, é um número obrigatório e positivo;
- direction: direção da regra (unidirecional ou bidirecional igual ao Snort 1), é um booleano obrigatório, false é bidirecional;
- source_ip: endereço IP do remetente, é uma string;
- destination_ip: endereço IP do destinatário, é uma string;
- ip_version: versão do IP, é número;
- max_length: tamanho máximo do payload do pacote, é número positivo;
- min_length: tamanho mínimo do payload do pacote, é número positivo;
- protocol: protocolo do pacote, é uma string;
- source_port: porta do remetente, é um número no intervalo 0-65536;
- destination_port: porta do destinatário, é um número no intervalo 0-65536;
- count: contagem de ocorrências, é um número positivo;
- interval: intervalo de tempo em segundos para count ser levado em consideração, é um número positivo;
- track: rastreamento, pode ser por IP de destino, IP de origem ou ambos, é uma string;
- flags: flags do *TCP*, é uma string.

Para todos os campos que não estão marcados como obrigatórios, qualquer valor vazio é desconsiderado na aplicação da regra assim como um 'any' nas regras do Snort (SNORT, 2023)

4.4.2 *Blacklist*

Esta coleção guardará todos os IPs monitorados, é a lista negra, o esquema da coleção é:

- ip: endereço ip, é uma string obrigatória e não repete;
- ip_version: versão do IP, é um número obrigatório (4 ou 6);
- reason: motivo do monitoramento, é uma string obrigatória;
- severity: severidade, é um número obrigatório e positivo.

4.4.3 *Alerts*

Esta coleção guardará todos os alertas do sistema, o esquema da coleção é:

- name: nome da regra que disparou o alerta, é uma string obrigatória;
- severity: severidade, é um número obrigatório e positivo;
- timestamp: milissegundos de quando ocorreu o alerta, é um número obrigatório;
- protocol: protocolo, é uma string obrigatória;
- source_ip: endereço IP do remetente, é uma string obrigatória;
- destination_ip: endereço IP do destinatário, é uma string obrigatória;
- length: tamanho do payload do pacote, é número positivo obrigatório;
- source_port: porta do remetente, é um número no intervalo 0-65536;
- destination_port: porta do destinatário, é um número no intervalo 0-65536.

4.4.4 *Packets*

Esta coleção funciona como uma fila que os *Agents* inserem dados (pacotes) e os *Enforcers* consomem para validar com as regras, o esquema da coleção é:

- timestamp: milissegundos de quando ocorreu o pacote, é um número obrigatório;
- source_ip: endereço IP do remetente, é uma string obrigatória;
- destination_ip: endereço IP do destinatário, é uma string obrigatória;

- ip_version: versão do IP, é um número obrigatório (4 ou 6);
- length: tamanho do payload do pacote, é número positivo obrigatório;
- protocol: protocolo, é uma string obrigatória;
- source_port: porta do remetente, é um número no intervalo 0-65536;
- destination_port: porta do destinatário, é um número no intervalo 0-65536;
- flags: flags do *TCP*, é um array de strings;
- captured_by: IP do *Agent* que capturou.

4.4.5 *Occurrences*

Esta coleção é similar à coleção de alertas, basicamente ela guardará as ocorrências de regras que possuem o atributou *count* e *interval*, o esquema da coleção é

- name: nome da regra, é uma string obrigatória que não repete;
- timestamp: milisegundos de quando ocorreu, é um número obrigatório;
- protocol: protocolo do pacote, é uma string;
- source_ip: endereço IP do remetente, é uma string;
- destination_ip: endereço IP do destinatário, é uma string;
- length: tamanho do payload do pacote, é número positivo;
- source_port: porta do remetente, é um número no intervalo 0-65536;
- destination_port: porta do destinatário, é um número no intervalo 0-65536.

4.5 FUNCIONAMENTO

O funcionamento da aplicação como um todo foi brevemente explicado anteriormente em cada subseção da arquitetura do sistema, aqui entrarei em detalhes de algumas decisões de desenvolvimento e o motivo disso.

Uma decisão tomada pensando em otimização foi que a verificação das regras testará as regras na ordem de severidade (maior para menor) e irá parar na primeira regra mais grave a ter correspondência, isso pode ser facilmente alterado mas para efeitos de prototipagem está funcional e eficiente.

Uma limitação deste sistema são os possíveis pacotes duplicados que é um problema passível de ocorrer quando múltiplos *agents* estão monitorando a rede e existe

alguma sobreposição no alcance do monitoramento, dessa forma irão ser capturados pacotes duplicados e para remediar isso foi incluído o atributo *'captured_by'* na coleção dos pacotes, desta forma é mais fácil de diferenciar se realmente foi um pacote duplicado por algum possível ataque ou monitoramento sobreposto mas isto ficará para um trabalho futuro e pode ser evitado com o correto posicionamento dos *agents*.

O *Enforcer* ao buscar as regras da coleção, irá dividir as regras em regras normais e regras complexas assim fazendo uso da coleção de *occurrences*, desta forma é possível que mesmo que várias instâncias desse serviço estejam trabalhando concorrentemente, essas regras que necessitam de contagem possam ser feitas sem problemas uma vez que toda ocorrência dessa regra é salva na coleção e a cada tanto tempo uma dessas instâncias irá verificar essa coleção para gerar alerta caso passe do valor de contagem no intervalo estipulado.

O *Enforcer* ainda faz um parse de regras bidirecionais para dividir em duas unidirecionais para simplificar a verificação das regras, a verificação das regras consiste em:

Código 4 – Trecho da verificação de regras simples do Enforcer

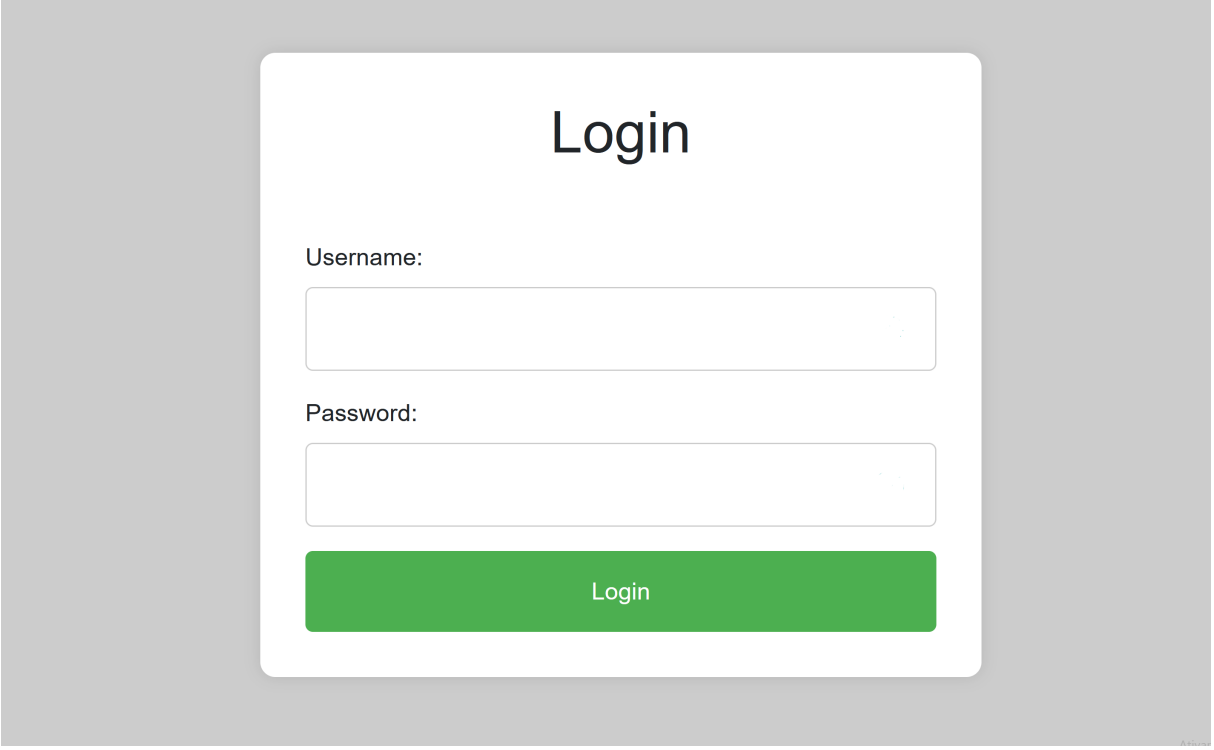
```
1     def verify(self, packet, rule):
2         for k, v in rule.items():
3             if k in self.ignored_fields: #ignored_fielsd = ['name',
                  'severity', 'description', 'direction', 'count',
                  'interval', 'track']
4                 continue
5
6             if k == 'flags':
7                 if 'flags' not in packet or v not in packet['flags']:
8                     return False
9             elif k == 'min_length':
10                if v < packet["length"]:
11                    return False
12            elif k == 'max_length':
13                if v > packet["length"]:
14                    return False
15            elif v != packet[k]:
16                return False
17        return True
```

Basicamente no Código 4 é feita a verificação de cada campo da regra que está definido ignorando alguns campos que não são necessários para a validação e então verifica especificamente campos que possuem lógicas diferentes como as flags que verifica se o valor está contido no conjunto, tamanho mínimo e máximo do

payload que faz a verificação de intervalo e depois a verificação genérica se os valores correspondem, esse método retorna verdadeiro ou falso para a regra, se a regra a ser verificada for uma regra simples, ou seja, que não envolve múltiplas ocorrências em um período de tempo, irá gerar um alerta, caso contrário irá salvar essa ocorrência na coleção de ocorrências que é verificada a cada minuto se extrapola o número de ocorrências no intervalo de tempo de cada regra e assim gera um alerta.

Quanto a interface gráfica, ao subir o módulo *App* e acessar no navegador, o operador será redirecionado à tela de login como na figura abaixo:

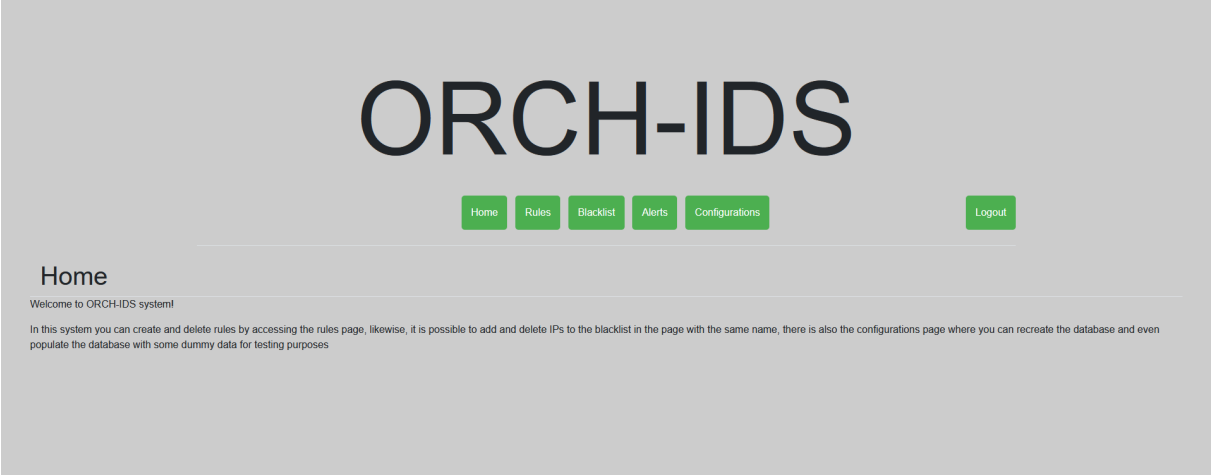
Figura 2 – Página de Login



Fonte: O autor.

Após o login o operador é redirecionado à página *Home* que contém uma breve descrição das outras páginas:

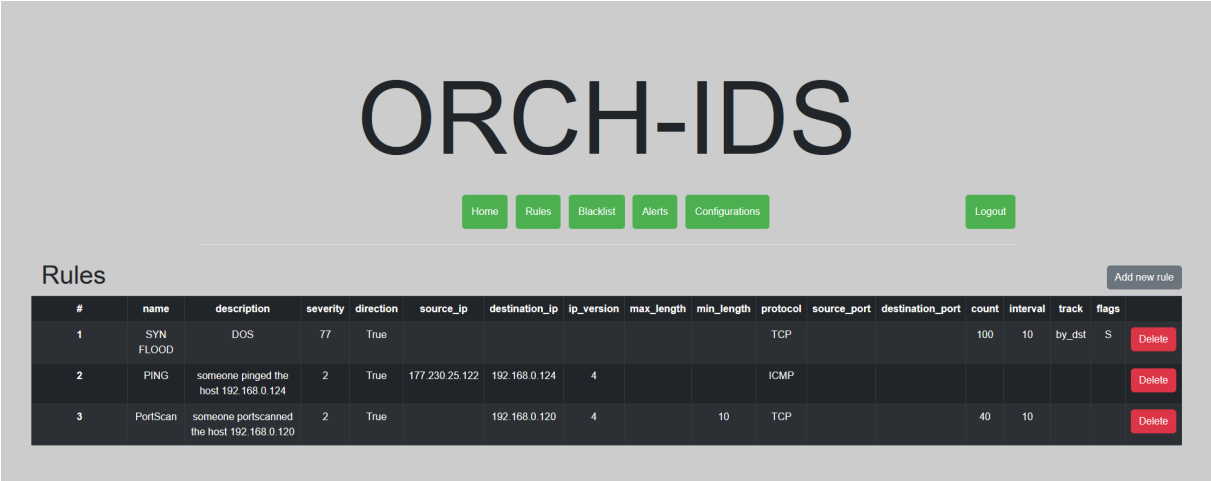
Figura 3 – Página Home



Fonte: O autor.

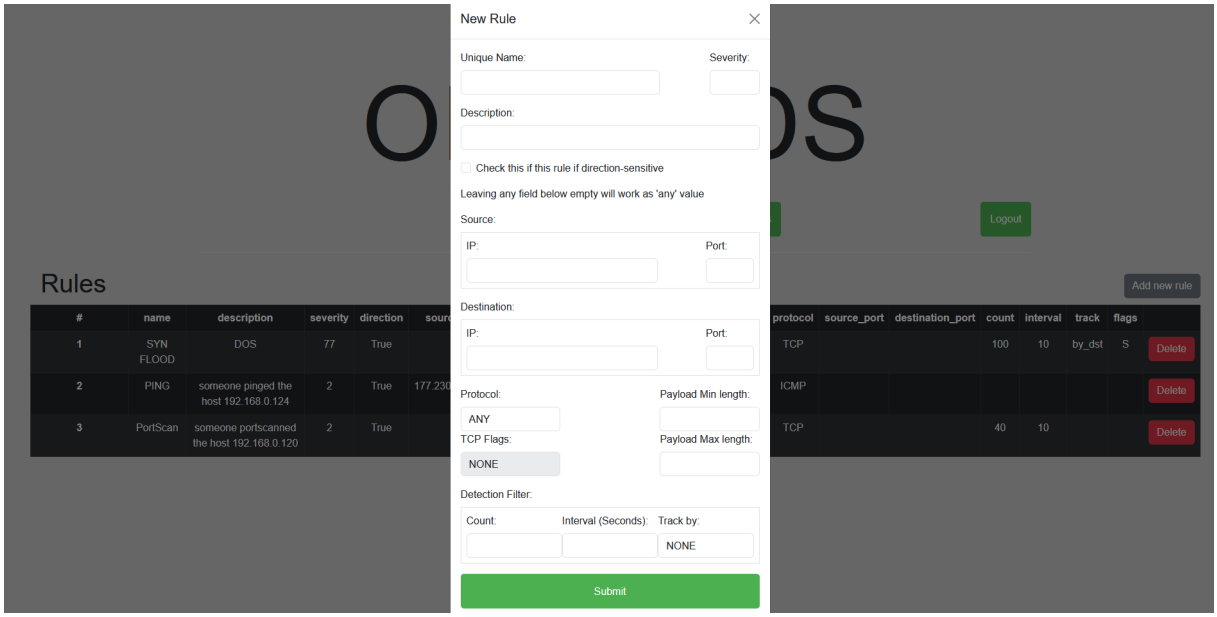
Na página *Rules* temos a lista de regras e também o formulário para adicionar uma nova regra:

Figura 4 – Página Rules



Fonte: O autor.

Figura 5 – Formulário de novas regras



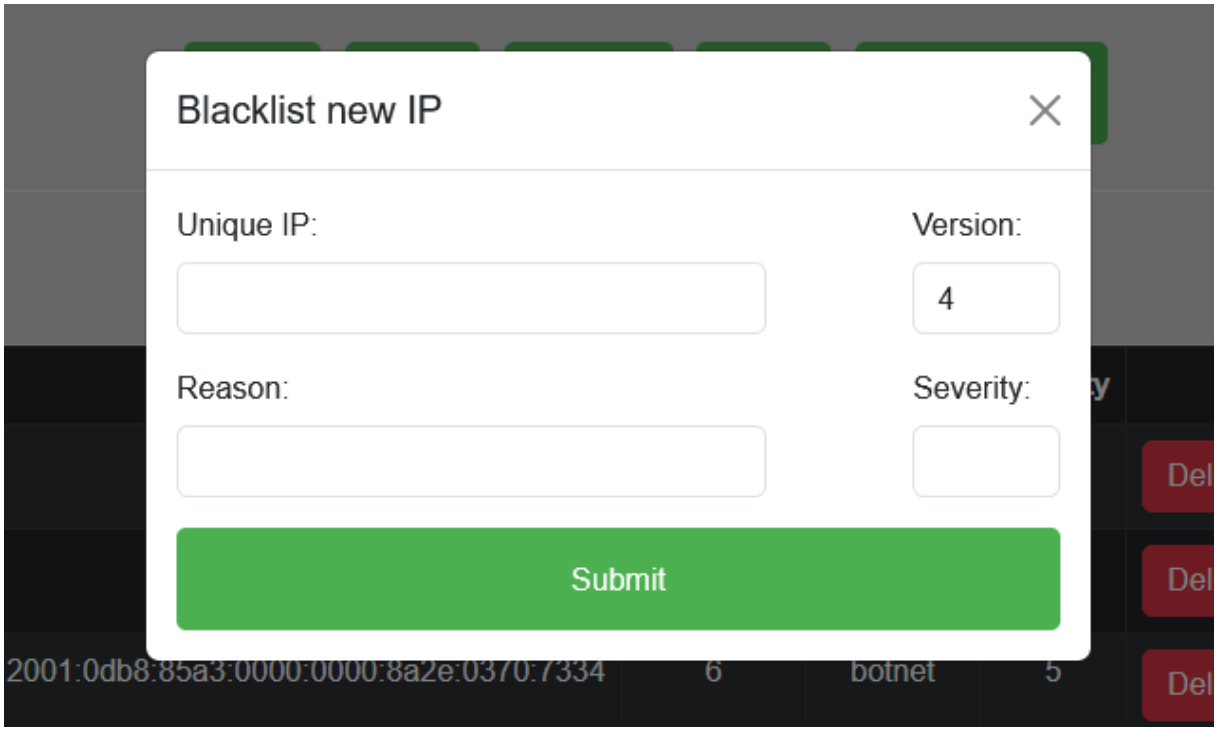
Fonte: O autor.

Assim como na página *Rules* temos também a página *Blacklist* que lista os IPs monitorados e também o formulário para adicionar um novo IP à essa lista:

Figura 6 – Página Blacklist

Fonte: O autor.

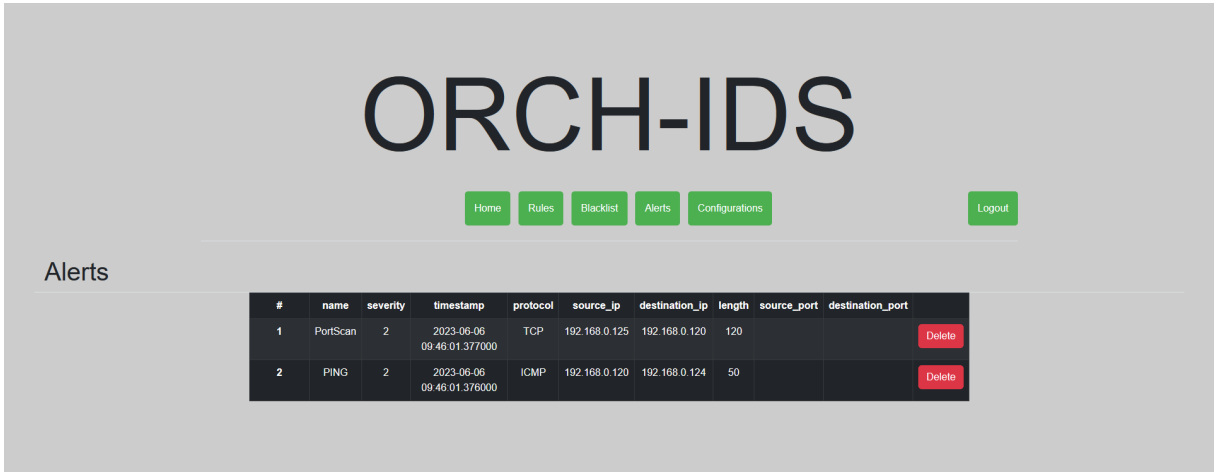
Figura 7 – Formulário de novas regras



Fonte: O autor.

Há também a página de *Alerts* que lista os alertas gerados pelas instâncias dos *Enforcers*:

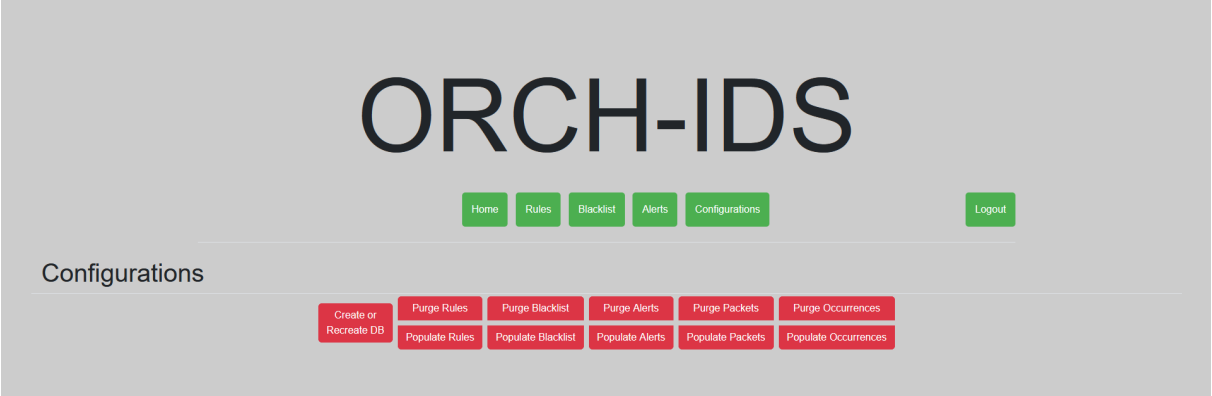
Figura 8 – Página Alerts



Fonte: O autor.

E por fim temos a página de *Configurations* que é a página de configurações:

Figura 9 – Página Configurations



Fonte: O autor.

Nesta página há várias configurações úteis, a primeira de todas é a mais importante para o funcionamento da aplicação que é o botão *Create or Recreate DB* que cria todo o esquema do banco de dados, após isso existe um botão para cada coleção, tabela, do banco de dados um para popular com dados de teste e outro para limpar a tabela.

Os botões para popular as tabelas foi mantido para serem usados no contexto de testes, uma vez que é possível realizar testes sem ser necessário ter todo o sistema em funcionamento, popular com regras básicas ou IPs monitorados, alertas ou até mesmo pacotes para testar o *Enforcer* sem precisar de um *Agent* para coletar dados. Estes dados populados estão definidos dentro do *App* no módulo *utils* especificamente no arquivo *populate_db.py*, consulte o Repositório deste projeto (BITTENCOURT, 2023) (Apêndice B) para mais informações e também como implantar os sistemas

5 CONSIDERAÇÕES FINAIS

Ao longo deste trabalho foi desenvolvido um protótipo de um IDS funcional que conseguiu satisfazer seus requisitos funcionais, permitindo regras customizáveis e monitoramento de IPs suspeitos, visualização e exclusão de alertas, permite múltiplos *agents* coletores e *Enforcers* para validar as regras, sendo assim um Sistema de Detecção de Intrusão na Rede baseado em regras e distribuído, um DRNIDS (*Distributed Rule-based NIDS*), além disso o sistema permite a substituição de qualquer um de seus serviços (*Agent, Enforcer, App*) tornando assim um sistema agnóstico a tecnologia e de certa forma vinculando apenas banco que opera sobre, o MongoDB.

5.1 TRABALHOS FUTUROS

Para os trabalhos futuros, seria possível ampliar o *Enforcer* para que ele tenha suporte à *Machine Learning* permitindo assim uma análise mais complexa e completa; Analisar o comportamento do sistema num cenário real (empresarial) para ter métricas da escalabilidade horizontal do sistema.

REFERÊNCIAS

BAKER, Kurt. **10 Most Common Types of Cyber Attacks**. [*S.l.*], 2023. Disponível em: <https://www.crowdstrike.com/cybersecurity-101/cyberattacks/most-common-types-of-cyberattacks>. Acesso em: 12 jun. 2023. Citado nas pp. 7, 8, 14.

BITTENCOURT, Yuri. **ORCH-IDS github repository**. [*S.l.*: *s.n.*], 2023. Disponível em: <https://github.com/YuriBittencourt/ORCH-IDS>. Acesso em: 14 jun. 2023. Citado na p. 30.

FRANCISCO, Clailton. Integração entre Rede Definida por Software e Sistema de Detecção de Intrusão para Mitigação de Ataques, 2019. Disponível em: <https://repositorio.ufsc.br/handle/123456789/202491>. Acesso em: 14 jun. 2023. Citado na p. 18.

JÚNIOR, João Francisco Gonçalves Sobrinho. Storm IDS: um Sistema de Detecção de Intrusão Escalável e Distribuído, 2016. Disponível em: <https://bdm.unb.br/bitstream/10483/13197/1/2016_JoaoFranciscoGoncalvesSobrinhoJunior.pdf>. Acesso em: 10 jul. 2023. Citado na p. 18.

KUMAR, Vikash *et al.* An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset, 2020. Disponível em: <https://doi.org/10.1007/s10586-019-03008-x>. Acesso em: 14 jun. 2023. Citado na p. 18.

MANSOUR, Yaser. **Rules Authors Introduction to Writing Snort 3 Rules**. [*S.l.*], 2020. Disponível em: <https://www.snort.org/documents/rules-writers-guide-to-snort-3-rules>. Acesso em: 12 jun. 2023. Citado nas pp. 16, 17.

OLIVEIRA, Ingrid. **Levantamento mostra que ataques cibernéticos no Brasil cresceram 94%**. [*S.l.*: *s.n.*], 2022. Disponível em: <https://www.cnnbrasil.com.br/tecnologia/levantamento-mostra-que-ataques-ciberneticos-no-brasil-cresceram-94>. Acesso em: 12 jun. 2023. Citado na p. 14.

PALO ALTO NETWORKS. **What is an Intrusion Detection System?** [*S.l.*]. Disponível em: <https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-detection-system-ids>. Acesso em: 12 jun. 2023. Citado na p. 16.

SNORT. **The Snort Project**. [*S.l.*: *s.n.*]. Disponível em: <https://snort.org/>. Acesso em: 12 jun. 2023. Citado nas pp. 14, 16, 18, 23.

STALLINGS, WILLIAM. **Segurança de Computadores**. 2ª Edição. [*S.l.*]: Elsevier Editora Ltda., 2012. ISBN 8535264493; 9788535264494. Citado na p. 16.

Anexos

ANEXO A – MANUAL DE IMPLANTAÇÃO

Manual de Implantação

Para uma fácil implantação, será listado tudo que na ordem que for preciso e é esperado que o usuário seja familiar com docker e tenha o mesmo configurado e instalado, em caso de dúvidas consulte a documentação oficial.

1. ### Banco de dados (Mongo)

   Você pode facilmente criar sua própria instância do MongoDB localmente com docker:

   ```
   docker run -dp 27017:27017 -v local-mongo:/data/db --name local-mongo --restart=always mongo:latest
   ```

   Este comando foi o utilizado para criar uma instância localmente do MongoDB, será explicado cada parâmetro:

   - `-dp 27017:27017` : Cria um container destacado do terminal e expõe a porta `27017`
   - `-v local-mongo:/data/db` : Monta o volume para persistir os dados (será criado um volume caso não exista)
   - `--name local-mongo` : Irá nomear a instância como `local-mongo`
   - `--restart-always` : Este parâmetro fará com que o container sempre inicie quando a máquina host esteja ligada
   - `mongo:latest` : A imagem do container, irá procurar localmente primeiro e depois no dockerhub

   Em caso de dúvidas sobre esse processo, consule o Docker docs run

2. ### App

   Primeiro de tudo é necessário que tenha instalado o `python3`, `pip` e o `venv` instalado, se você usa um linux baseado em debian poderá instalar os pacotes assim:

   ```
   sudo apt install python3 python3-pip python3.8-venv
   ```

   Dentro da pasta do App, você pode criar um virtual environment do python se você quiser mais isolamento, para criar e ativar execute:

   ```
   python -m venv .
   source bin/activate
   ```

   Instale todas as bibliotecas requeridas:

   ```
   pip install -r requirements.txt
   ```

   copie `.env.example` para `.env` e configure-o, você pode copiar com:

   ```
   cp .env.example .env
   ```

   Agora apenas tenha certeza que os valores estão corretos e depois disso pode subir a aplicação web com:

   ```
   python3 app.py
   ```

3. ### Agent

   Primeiro de tudo é necessário que tenha instalado o `pcap`, `python3`, `pip` e `venv` instalado, se você usa um linux baseado em debian poderá instalar os pacotes assim:

   ```
   sudo apt install python3 python3-pip python3.8-venv libpcap-dev
   ```

   Se você estiver usando Windows use o Npcap ao invés do pcap.

   Dentro da pasta do Agent, você pode criar um virtual environment do python se você quiser mais isolamento, para criar e ativar execute:

   ```
   python -m venv .
   source bin/activate
   ```

   Instale todas as bibliotecas requeridas:

   ```
   pip install -r requirements.txt
   ```

   Copie `.env.example` para `.env` e configure-o, você pode copiar com:

   ```
   cp .env.example .env
   ```

   IMPORTANTE: Você deve por a sua interface de rede em modo promíscuo se você desejar capturar todos os pacotes da rede e não apenas os endereçados a sua interface de rede, para fazer isso no linux:

```
#change [interface] with the same value as NETWORK_INTERFACE inserted in .env file
ip link set [interface] promisc on
```

Agora apenas tenha certeza que os valores estão corretos e depois disso pode subir o agent com `sudo`:

```
sudo python3 agent.py
```

Observação: Você pode executar quantos agentes desejar na rede, apenas tome cuidado para não sobrepor a àrea de cobertura dos mesmos ou ocorrerá registro duplicado de pacotes e isso poderá afetar a geração de alertas

4. ### Enforcer

Primeiro de tudo é necessário que tenha instalado o `python3`, `pip` e `venv` instalado, se você usa um linux baseado em debian poderá instalar os pacotes assim:

```
sudo apt install python3 python3-pip python3.8-venv
```

Dentro da pasta do Enforcer, você pode criar um virtual environment do python se você quiser mais isolamento, para criar e ativar execute:

```
python -m venv .
source bin/activate
```

Instale todas as bibliotecas requeridas:

```
pip install -r requirements.txt
```

Copie `.env.example` para `.env` e configure-o, você pode copiar com:

```
cp .env.example .env
```

Agora apenas tenha certeza que os valores estão corretos e depois disso pode subir o enforcer com:

```
python3 enforcer.py
```

ANEXO B – CÓDIGO FONTE

File - D:\ORCH-IDS\app\app.py

```python
from flask import Flask, render_template, request, redirect, session, url_for
from dotenv import dotenv_values
from bson import ObjectId

from utils.mongo import mongo_instance as mongo
from utils.auth_decorator import authenticated_resource
from utils.setup_db import setup_db
import utils.populate_db as pop
from utils.schema import schemas

app = Flask(__name__)
config = dotenv_values()
app.secret_key = config['SECRET_KEY']
name = 'ORCH-IDS'

credentials = {'username': config['ADMIN_USER'], 'password': config['ADMIN_PASSWORD']}


@app.route('/')
@authenticated_resource
def home():
    return render_template('home.jinja2', title=name)


@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        if username == credentials['username'] and password == credentials['password']:
            session['username'] = username
            return redirect(url_for('home'))
        else:
            return render_template('login.jinja2', error='Invalid Credentials!')

```

```
37      return render_template('login.jinja2', title=name
    )
38
39
40  @app.route('/logout')
41  @authenticated_resource
42  def logout():
43      session.pop('username', None)
44      return redirect(url_for('login'))
45
46
47  @app.route('/rules', methods=['GET', 'POST', 'DELETE'
    ])
48  @authenticated_resource
49  def rules():
50      # Add new rules
51      if request.method == 'POST':
52          try:
53              new = {k: v if schemas['rules']['
    properties'][k]['type'] == 'string'
54              else int(v) if schemas['rules']['
    properties'][k]['type'] == 'number'
55              else bool(v) for k, v in request.form.
    items() if v}
56
57              if 'direction' not in new:
58                  new['direction'] = False
59
60              mongo.db[mongo.collections['rules']].
    insert_one(new)
61
62          except Exception as e:
63              print(e)
64
65      # Delete rule
66      if request.method == 'DELETE':
67          try:
68              mongo.db[mongo.collections['rules']].
    delete_one({'_id': ObjectId(request.json['_id'])})
69          except Exception as e:
70              print(e)
```

```
 71
 72     # Retrieve rules
 73     rules_list = list(mongo.db[mongo.collections['
    rules']].find())
 74     keys = list(schemas['rules']['properties'].keys
    ())
 75
 76     return render_template('rules.jinja2', list=
    rules_list, keys=keys, title=name, route='/rules')
 77
 78
 79 @app.route('/blacklist', methods=['GET', 'POST', '
    DELETE'])
 80 @authenticated_resource
 81 def blacklist():
 82     # Blacklist new IP
 83     if request.method == 'POST':
 84         try:
 85             new = {k: v if schemas['blacklist']['
    properties'][k]['type'] == 'string'
 86             else int(v) if schemas['blacklist']['
    properties'][k]['type'] == 'number'
 87             else bool(v) for k, v in request.form.
    items() if v}
 88
 89             mongo.db[mongo.collections['blacklist'
    ]].insert_one(new)
 90
 91         except Exception as e:
 92             print(e)
 93
 94     # Remove IP from blacklist
 95     if request.method == 'DELETE':
 96         try:
 97             mongo.db[mongo.collections['blacklist'
    ]].delete_one({'_id': ObjectId(request.json['_id'
    ])})
 98         except Exception as e:
 99             print(e)
100
101     # Retrieve blacklisted IPs
```

```
102     ban_list = list(mongo.db[mongo.collections['
    blacklist']].find())
103     keys = list(schemas['blacklist']['properties'].
    keys())
104 
105     return render_template('blacklist.jinja2', list=
    ban_list, keys=keys, title=name, route='/blacklist')
106 
107 
108 @app.route('/alerts', methods=['GET', 'DELETE'])
109 @authenticated_resource
110 def alerts():
111     # Delete alerts
112     if request.method == 'DELETE':
113         try:
114             mongo.db[mongo.collections['alerts']].
    delete_one({'_id': ObjectId(request.json['_id'])})
115         except Exception as e:
116             print(e)
117 
118     # Retrieve alerts
119     alerts_list = list(mongo.db[mongo.collections['
    alerts']].aggregate([
120         {
121             '$addFields': {
122                 'timestamp': {
123                     '$toDate': '$timestamp'
124                 }
125             }
126         }, {
127             '$sort': {
128                 'timestamp': -1,
129                 'severity': -1
130             }
131         }
132     ]))
133     keys = list(schemas['alerts']['properties'].keys
    ())
134 
135     return render_template('alerts.jinja2', list=
    alerts_list, keys=keys, title=name, route='/alerts')
```

```
136
137
138 @app.route('/configurations/')
139 @app.route('/configurations/<action>')
140 @authenticated_resource
141 def configurations(action=None):
142     execute = None
143     if action == 'setup_db':
144         execute = setup_db
145
146     elif action == 'drop_rules':
147         execute = pop.drop_rules
148
149     elif action == 'populate_rules':
150         execute = pop.populate_rules
151
152     elif action == 'drop_blacklist':
153         execute = pop.drop_blacklist
154
155     elif action == 'populate_blacklist':
156         execute = pop.populate_blacklist
157
158     elif action == 'drop_packets':
159         execute = pop.drop_packets
160
161     elif action == 'populate_packets':
162         execute = pop.populate_packets
163
164     elif action == 'drop_alerts':
165         execute = pop.drop_alerts
166
167     elif action == 'populate_alerts':
168         execute = pop.populate_alerts
169
170     elif action == 'drop_occurrences':
171         execute = pop.drop_occurrences
172
173     elif action == 'populate_occurrences':
174         execute = pop.populate_occurrences
175
176     if execute:
```

```
177         try:
178             execute()
179         except Exception as e:
180             print(e)
181 
182     return render_template('configurations.jinja2',
    title=name)
183 
184 
185 if __name__ == '__main__':
186     if config['ENVIRONMENT'] == 'DEV':
187         app.run(host='0.0.0.0', port=config['PORT'
    ], debug=True)
188 
189     elif config['ENVIRONMENT'] == 'PROD':
190         from waitress import serve
191 
192         serve(app, host='0.0.0.0', port=config['PORT
    '])
193 
194     else:
195         raise ValueError('WRONG environment variable
    ')
196 
```

```
MONGO_HOST='mongodb://127.0.0.1'
MONGO_PORT=27017
MONGO_DB='NIDS'
MONGO_COLLECTION_QUEUE='packets'
MONGO_COLLECTION_RULES='rules'
MONGO_COLLECTION_BLACKLISTED='blacklist'
MONGO_COLLECTION_ALERTS='alerts'
MONGO_COLLECTION_OCCURRENCES='occurrences'
ADMIN_USER='admin'
ADMIN_PASSWORD='admin'
ENVIRONMENT='DEV'
PORT=5000
SECRET_KEY='super-secret-key'
```

```
blinker==1.6.2
click==8.1.3
colorama==0.4.6
dnspython==2.3.0
Flask==2.3.2
itsdangerous==2.1.2
Jinja2==3.1.2
MarkupSafe==2.1.3
pymongo==4.3.3
python-dotenv==1.0.0
Werkzeug==2.3.4
waitress==2.1.2
```

```python
from pymongo import MongoClient
from dotenv import dotenv_values


class Mongo:
    def __init__(self):
        config = dotenv_values()
        client = MongoClient(host=config['MONGO_HOST'], port=int(config['MONGO_PORT']))
        self.db = client[config['MONGO_DB']]
        self.collections = {
            'packets': config['MONGO_COLLECTION_QUEUE'],
            'rules': config['MONGO_COLLECTION_RULES'],
            'blacklist': config['MONGO_COLLECTION_BLACKLISTED'],
            'alerts': config['MONGO_COLLECTION_ALERTS'],
            'occurrences': config['MONGO_COLLECTION_OCCURRENCES']
        }


mongo_instance = Mongo()

```

File - D:\ORCH-IDS\app\utils\schema.py

```
class Schema:
    def __init__(self):
        self.schemas = {
            'blacklist': {
                'bsonType': 'object',
                'required':
                    [
                        'ip',
                        'ip_version',
                        'reason',
                        'severity'
                    ],
                'properties': {
                    'ip': {
                        'type': 'string'
                    },
                    'ip_version': {
                        'type': 'number',
                        'enum': [4, 6]
                    },
                    'reason': {
                        'type': 'string'
                    },
                    'severity': {
                        'type': 'number',
                        'minimum': 0
                    }
                }
            },

            'packets': {
                'bsonType': 'object',
                'required':
                    [
                        'timestamp',
                        'source_ip',
                        'destination_ip',
                        'ip_version',
                        'length',
                        'protocol',
                        'captured_by'
```

```
                    ],
                'properties': {
                    'timestamp': {
                        'type': 'number'
                    },
                    'source_ip': {
                        'type': 'string'
                    },
                    'destination_ip': {
                        'type': 'string'
                    },
                    'ip_version': {
                        'type': 'number',
                        'enum': [4, 6]
                    },
                    'length': {
                        'type': 'number'
                    },
                    'protocol': {
                        'type': 'string',
                        'enum': ['TCP', 'UDP', 'ICMP']
                    },
                    'source_port': {
                        'type': 'number'
                    },
                    'destination_port': {
                        'type': 'number'
                    },

                    'flags': {
                        'type': 'array',
                        'items': {
                            'type': 'string'
                        }
                    },
                    'captured_by': {
                        'type': 'string'
                    }
                }
            },
```

```
 82
 83              'rules': {
 84                  'bsonType': 'object',
 85                  'required':
 86                      [
 87                          'name',
 88                          'description',
 89                          'severity',
 90                          'direction'
 91                      ],
 92                  'properties': {
 93                      'name': {
 94                          'type': 'string'
 95                      },
 96                      'description': {
 97                          'type': 'string'
 98                      },
 99                      'severity': {
100                          'type': 'number',
101                          'minimum': 0
102                      },
103                      'direction': {
104                          'type': 'boolean',
105                      },
106                      'source_ip': {
107                          'type': 'string',
108                      },
109                      'destination_ip': {
110                          'type': 'string',
111                      },
112                      'ip_version': {
113                          'type': 'number',
114                          'enum': [4, 6]
115                      },
116                      'max_length': {
117                          'type': 'number',
118                          'minimum': 0
119                      },
120                      'min_length': {
121                          'type': 'number',
122                          'minimum': 0
```

```
123                         },
124                         'protocol': {
125                             'type': 'string',
126                             'enum': ['TCP', 'UDP', 'ICMP
    ']
127                         },
128                         'source_port': {
129                             'type': 'number',
130                             'minimum': 0,
131                             'maximum': 65536
132                         },
133                         'destination_port': {
134                             'type': 'number',
135                             'minimum': 0,
136                             'maximum': 65536
137                         },
138                         'count': {
139                             'type': 'number',
140                             'minimum': 1
141                         },
142                         'interval': {
143                             'type': 'number',
144                             'minimum': 1
145                         },
146                         'track': {
147                             'type': 'string',
148                             'enum': ['by_src', 'by_dst'
    , 'both']
149                         },
150                         'flags': {
151                             'type': 'string'
152                         }
153                     }
154 
155                 },
156 
157                 'alerts': {
158                     'bsonType': 'object',
159                     'required':
160                         [
161                             'name',
```

```
                            'severity',
                            'timestamp',
                            'protocol',
                            'source_ip',
                            'destination_ip',
                            'length'
                        ],
                    'properties': {
                        'name': {
                            'type': 'string'
                        },
                        'severity': {
                            'type': 'number',
                            'minimum': 0
                        },
                        'timestamp': {
                            'type': 'number',
                            'minimum': 0
                        },
                        'protocol': {
                            'type': 'string',
                            'enum': ['TCP', 'UDP', 'ICMP
    ']
                        },
                        'source_ip': {
                            'type': 'string'
                        },
                        'destination_ip': {
                            'type': 'string'
                        },
                        'length': {
                            'type': 'number',
                        },
                        'source_port': {
                            'type': 'number',
                            'minimum': 0,
                            'maximum': 65536
                        },
                        'destination_port': {
                            'type': 'number',
                            'minimum': 0,
```

```
202                             'maximum': 65536
203                         }
204                     }
205                 },
206                 'occurrences': {
207                     'bsonType': 'object',
208                     'required':
209                         [
210                             'name',
211                             'timestamp',
212                             'protocol',
213                             'source_ip',
214                             'destination_ip',
215                             'length'
216                         ],
217                     'properties': {
218                         'name': {
219                             'type': 'string'
220                         },
221                         'timestamp': {
222                             'type': 'number',
223                             'minimum': 0
224                         },
225                         'protocol': {
226                             'type': 'string',
227                             'enum': ['TCP', 'UDP', 'ICMP
    ']
228                         },
229                         'source_ip': {
230                             'type': 'string'
231                         },
232                         'destination_ip': {
233                             'type': 'string'
234                         },
235                         'length': {
236                             'type': 'number',
237                         },
238                         'source_port': {
239                             'type': 'number',
240                             'minimum': 0,
241                             'maximum': 65536
```

```
242                     },
243                     'destination_port': {
244                         'type': 'number',
245                         'minimum': 0,
246                         'maximum': 65536
247                     }
248                 }
249 
250             }
251         }
252 
253         self.indexes = {
254             'blacklist': [{'value': 'ip', 'unique': True}],
255             'packets': [{'value': [('timestamp', 1)], 'unique': False}],
256             'rules': [{'value': 'name', 'unique': True}, {'value': [('severity', -1)], 'unique': False}],
257             'alerts': [{'value': [('severity', -1), ('timestamp', -1)], 'unique': False}],
258             'occurrences': [{'value': [('severity', -1), ('timestamp', -1)], 'unique': False}]
259         }
260 
261 schemas = Schema().schemas
262 
```

```python
from .mongo import mongo_instance as mongo
from .schema import Schema


def setup_db():
    schema = Schema()

    # Drop collections if they do exist and create:
    for collection in mongo.collections:
        mongo.db.drop_collection(collection)
        mongo.db.create_collection(collection, validator={'$jsonSchema': schema.schemas[collection]})

        # create index if there is a index
        if collection in schema.indexes:
            for index in schema.indexes[collection]:
                mongo.db[collection].create_index(index['value'], unique=index['unique'])

    # List created collections:
    print(mongo.db.list_collection_names())
```

File - D:\ORCH-IDS\app\utils\populate_db.py

```python
from .mongo import mongo_instance as mongo


def drop_blacklist():
    mongo.db[mongo.collections['blacklist']].delete_many({})


def populate_blacklist():
    documents =[
        {'ip': '192.164.0.32', 'ip_version': 4, 'reason': 'botnet', 'severity': 5},
        {'ip': '192.164.2.32', 'ip_version': 4, 'reason': 'suspicious', 'severity': 5},
        {'ip': '2001:0db8:85a3:0000:0000:8a2e:0370:7334', 'ip_version': 6, 'reason': 'botnet', 'severity': 5},
        {'ip': '2001:0db8:85a3:0000:0000:8a2e:0370:7335', 'ip_version': 6, 'reason': 'suspicious', 'severity': 5},
    ]
    mongo.db[mongo.collections['blacklist']].insert_many(documents, ordered=False)


def drop_packets():
    mongo.db[mongo.collections['packets']].delete_many({})


def populate_packets():
    documents =[
        {
            'timestamp': 1686044761376,
            'source_ip': '192.168.0.114',
            'destination_ip': '66.22.246.136',
            'ip_version': 4,
            'length': 300,
            'protocol': 'UDP',
            'source_port': 54380,
            'destination_port': 50002,
```

```
33              'captured_by': 'dummy'
34          },
35          {
36              'timestamp': 1686044761380,
37              'source_ip': '192.168.0.114',
38              'destination_ip': '66.22.246.136',
39              'ip_version': 4,
40              'length': 300,
41              'protocol': 'UDP',
42              'source_port': 54380,
43              'destination_port': 50002,
44              'captured_by': 'dummy'
45          },
46          {
47              'timestamp': 1686044761381,
48              'source_ip': '192.168.0.114',
49              'destination_ip': '66.22.246.136',
50              'ip_version': 4,
51              'length': 300,
52              'protocol': 'UDP',
53              'source_port': 54380,
54              'destination_port': 50002,
55              'captured_by': 'dummy'
56          },
57          {
58              'timestamp': 1686044761434,
59              'source_ip': '192.168.0.114',
60              'destination_ip': '162.159.129.235',
61              'ip_version': 4,
62              'length': 127,
63              'protocol': 'TCP',
64              'source_port': 58964,
65              'destination_port': 443,
66              'flags': [
67                  'P',
68                  'A'
69              ],
70              'captured_by': 'dummy'
71          },
72          {
73              'timestamp': 1686044761447,
```

```
 74                 'source_ip': '192.168.0.114',
 75                 'destination_ip': '35.215.218.63',
 76                 'ip_version': 4,
 77                 'length': 71,
 78                 'protocol': 'UDP',
 79                 'source_port': 55612,
 80                 'destination_port': 50002,
 81                 'captured_by': 'dummy'
 82             },
 83             {
 84                 'timestamp': 1686044761452,
 85                 'source_ip': '162.159.129.235',
 86                 'destination_ip': '192.168.0.114',
 87                 'ip_version': 4,
 88                 'length': 40,
 89                 'protocol': 'TCP',
 90                 'source_port': 443,
 91                 'destination_port': 58964,
 92                 'flags': [
 93                     'A'
 94                 ],
 95                 'captured_by': 'dummy'
 96             },
 97             {
 98                 'name': 'SYN flood',
 99                 'timestamp': 1686044761376,
100                 'protocol': 'TCP',
101                 'source_ip': '192.168.0.120',
102                 'destination_ip': '192.168.0.124',
103                 'length': 10,
104                 'ip_version': 4,
105                 'captured_by': 'dummy',
106                 'flags': [
107                     'S'
108                 ],
109             },
110             {
111                 'name': 'SYN flood',
112                 'timestamp': 1686044761377,
113                 'protocol': 'TCP',
114                 'source_ip': '192.168.0.120',
```

```
115                 'destination_ip': '192.168.0.124',
116                 'length': 10,
117                 'ip_version': 4,
118                 'captured_by': 'dummy',
119                 'flags': [
120                     'S'
121                 ]
122             },
123             {
124                 'name': 'SYN flood',
125                 'timestamp': 1686044761378,
126                 'protocol': 'TCP',
127                 'source_ip': '192.168.0.120',
128                 'destination_ip': '192.168.0.124',
129                 'length': 10,
130                 'ip_version': 4,
131                 'captured_by': 'dummy',
132                 'flags': [
133                     'S'
134                 ]
135             },
136             {
137                 'name': 'SYN flood',
138                 'timestamp': 1686044761379,
139                 'protocol': 'TCP',
140                 'source_ip': '192.168.0.120',
141                 'destination_ip': '192.168.0.124',
142                 'length': 10,
143                 'ip_version': 4,
144                 'captured_by': 'dummy',
145                 'flags': [
146                     'S'
147                 ]
148             },
149             {
150                 'name': 'SYN flood',
151                 'timestamp': 1686044761380,
152                 'protocol': 'TCP',
153                 'source_ip': '192.168.0.120',
154                 'destination_ip': '192.168.0.124',
155                 'length': 10,
```

```
            'ip_version': 4,
            'captured_by': 'dummy',
            'flags': [
                'S'
            ]
        },
        {
            'name': 'SYN flood',
            'timestamp': 1686044761381,
            'protocol': 'TCP',
            'source_ip': '192.168.0.120',
            'destination_ip': '192.168.0.124',
            'length': 10,
            'ip_version': 4,
            'captured_by': 'dummy',
            'flags': [
                'S'
            ]
        },
        {
            'name': 'SYN flood',
            'timestamp': 1686044761382,
            'protocol': 'TCP',
            'source_ip': '192.168.0.120',
            'destination_ip': '192.168.0.124',
            'length': 10,
            'ip_version': 4,
            'captured_by': 'dummy',
            'flags': [
                'S'
            ]
        },
        {
            'name': 'SYN flood',
            'timestamp': 1686044761383,
            'protocol': 'TCP',
            'source_ip': '192.168.0.120',
            'destination_ip': '192.168.0.124',
            'length': 10,
            'ip_version': 4,
            'captured_by': 'dummy',
```

```
197                 'flags': [
198                     'S'
199                 ]
200             },
201             {
202                 'name': 'SYN flood',
203                 'timestamp': 1686044761384,
204                 'protocol': 'TCP',
205                 'source_ip': '192.168.0.120',
206                 'destination_ip': '192.168.0.124',
207                 'length': 10,
208                 'ip_version': 4,
209                 'captured_by': 'dummy',
210                 'flags': [
211                     'S'
212                 ]
213             },
214             {
215                 'name': 'SYN flood',
216                 'timestamp': 1686044761385,
217                 'protocol': 'TCP',
218                 'source_ip': '192.168.0.120',
219                 'destination_ip': '192.168.0.124',
220                 'length': 10,
221                 'ip_version': 4,
222                 'captured_by': 'dummy',
223                 'flags': [
224                     'S'
225                 ]
226             },
227             {
228                 'name': 'SYN flood',
229                 'timestamp': 1686044761386,
230                 'protocol': 'TCP',
231                 'source_ip': '192.168.0.120',
232                 'destination_ip': '192.168.0.124',
233                 'length': 10,
234                 'ip_version': 4,
235                 'captured_by': 'dummy',
236                 'flags': [
237                     'S'
```

```
238             ]
239         },
240     ]
241     mongo.db[mongo.collections['packets']].
    insert_many(documents, ordered=False)
242
243
244 def drop_rules():
245     mongo.db[mongo.collections['rules']].delete_many
    ({})
246
247
248 def populate_rules():
249     documents = [
250         {
251             'name': 'PING',
252             'description': 'someone pinged the host
    192.168.0.124',
253             'severity':  2,
254             'direction': True,
255             'source_ip': '177.230.25.122',
256             'destination_ip': '192.168.0.124',
257             'ip_version': 4,
258             'protocol': 'ICMP',
259         },
260         {
261             'name': 'PortScan',
262             'description': 'someone portscanned the
    host 192.168.0.120',
263             'severity': 2,
264             'direction': True,
265             'destination_ip': '192.168.0.120',
266             'min_length': 10,
267             'ip_version': 4,
268             'protocol': 'TCP',
269             'count': 40,
270             'interval': 10
271         },
272     ]
273     mongo.db[mongo.collections['rules']].insert_many
    (documents, ordered=False)
```

```
274
275
276 def drop_alerts():
277     mongo.db[mongo.collections['alerts']].
    delete_many({})
278
279
280 def populate_alerts():
281     documents = [
282         {
283             'name': 'PING',
284             'severity': 2,
285             'timestamp': 1686044761376,
286             'protocol': 'ICMP',
287             'source_ip': '192.168.0.120',
288             'destination_ip': '192.168.0.124',
289             'length': 50
290         },
291
292         {
293             'name': 'PortScan',
294             'severity': 2,
295             'timestamp': 1686044761377,
296             'protocol': 'TCP',
297             'source_ip': '192.168.0.125',
298             'destination_ip': '192.168.0.120',
299             'length': 120
300         },
301
302     ]
303     mongo.db[mongo.collections['alerts']].
    insert_many(documents, ordered=False)
304
305
306 def drop_occurrences():
307     mongo.db[mongo.collections['occurrences']].
    delete_many({})
308
309
310 def populate_occurrences():
311     documents = [
```

```
312         {
313             'name': 'SYN flood',
314             'timestamp': 1686044761376,
315             'protocol': 'TCP',
316             'source_ip': '192.168.0.120',
317             'destination_ip': '192.168.0.124',
318             'length': 10
319         },
320         {
321             'name': 'SYN flood',
322             'timestamp': 1686044761377,
323             'protocol': 'TCP',
324             'source_ip': '192.168.0.120',
325             'destination_ip': '192.168.0.124',
326             'length': 10
327         },
328         {
329             'name': 'SYN flood',
330             'timestamp': 1686044761378,
331             'protocol': 'TCP',
332             'source_ip': '192.168.0.120',
333             'destination_ip': '192.168.0.124',
334             'length': 10
335         },
336         {
337             'name': 'SYN flood',
338             'timestamp': 1686044761379,
339             'protocol': 'TCP',
340             'source_ip': '192.168.0.120',
341             'destination_ip': '192.168.0.124',
342             'length': 10
343         },
344         {
345             'name': 'SYN flood',
346             'timestamp': 1686044761380,
347             'protocol': 'TCP',
348             'source_ip': '192.168.0.120',
349             'destination_ip': '192.168.0.124',
350             'length': 10
351         },
352         {
```

```
            'name': 'SYN flood',
            'timestamp': 1686044761381,
            'protocol': 'TCP',
            'source_ip': '192.168.0.120',
            'destination_ip': '192.168.0.124',
            'length': 10
        },
        {
            'name': 'SYN flood',
            'timestamp': 1686044761382,
            'protocol': 'TCP',
            'source_ip': '192.168.0.120',
            'destination_ip': '192.168.0.124',
            'length': 10
        },
        {
            'name': 'SYN flood',
            'timestamp': 1686044761383,
            'protocol': 'TCP',
            'source_ip': '192.168.0.120',
            'destination_ip': '192.168.0.124',
            'length': 10
        },
        {
            'name': 'SYN flood',
            'timestamp': 1686044761384,
            'protocol': 'TCP',
            'source_ip': '192.168.0.120',
            'destination_ip': '192.168.0.124',
            'length': 10
        },
        {
            'name': 'SYN flood',
            'timestamp': 1686044761385,
            'protocol': 'TCP',
            'source_ip': '192.168.0.120',
            'destination_ip': '192.168.0.124',
            'length': 10
        },
        {
            'name': 'SYN flood',
```

```
            'timestamp': 1686044761386,
            'protocol': 'TCP',
            'source_ip': '192.168.0.120',
            'destination_ip': '192.168.0.124',
            'length': 10
        },
    ]
    mongo.db[mongo.collections['occurrences']].
insert_many(documents, ordered=False)
```

File - D:\ORCH-IDS\app\utils\auth_decorator.py

```python
from functools import wraps
from flask import session, redirect, url_for


def authenticated_resource(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        if 'username' in session:
            return f(*args, **kwargs)

        return redirect(url_for('login'))

    return decorated
```

File - D:\ORCH-IDS\app\static\css\main.css

```css
body {

    align-items: center;
    justify-content: center;
    margin-top: 10vh;
    padding: 0;
    font-family: Arial, sans-serif;
    background-color: #cccccc;
}

/* Chrome, Safari, Edge, Opera */
input::-webkit-outer-spin-button,
input::-webkit-inner-spin-button {
  -webkit-appearance: none;
  margin: 0;
}

/* Firefox */
input[type=number] {
  -moz-appearance: textfield;
}

.login{
    display: flex;
}

table.table-fit {
    width: auto;
    table-layout: auto;
}

table.table-fit thead th, table.table-fit tfoot th {
    width: auto;
}

table.table-fit tbody td, table.table-fit tfoot td {
    width: auto;
}

.title{
    text-align: center;
```

```
    font-size: 10em;
    font-family: Arial, sans-serif;
}

.login-container {
    max-width: 30em;
    width: 90%;
    padding: 1.875em;
    background-color: #fff;
    border-radius: 0.625em;
    box-shadow: 0 0 0.625em rgba(0, 0, 0, 0.1);
}

.login-container h1 {
    text-align: center;
    margin-bottom: 1.25em;
}

.login-container input[type="text"],
.login-container input[type="password"] {
    width: 100%;
    padding: 0.9375em;
    margin-bottom: 0.625em;
    border: 0.0625em solid #ccc;
    border-radius: 0.3125em;
}

ul li{
    margin-left: 0.4rem;
    margin-right: 0.4rem;
}
.green-btn {
    width: 100%;
    padding: 0.9375em;
    background-color: #4CAF50;
    color: #fff;
    border: none;
    border-radius: 0.3125em;
    cursor: pointer;
}

```

```
.green-btn:hover {
    background-color: #45a049;
}

@media (max-width: 23.4375em) {
    .login-container {
        max-width: 22em;
        padding: 1.25em;
    }
}
```

File - D:\ORCH-IDS\app\templates\home.jinja2

```
{% extends 'components/base.jinja2' %}

{% block content %}
    <div class="d-flex align-items-center justify-content-between border-bottom">
        <h1 class="mx-3">Home</h1>
    </div>
    <div class="content">
        <p> Welcome to {{title}} system!</p>
        <p> In this system you can create and delete rules by accessing the rules page, likewise, it is possible to add and delete IPs to the blacklist in the page with the same name, there is also the configurations page where you can recreate the database and even populate the database with some dummy data for testing purposes </p>
    </div>
{% endblock %}


```

File - D:\ORCH-IDS\app\templates\login.jinja2

```
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='css/main.css') }}">
</head>
<body class="login">
<div class="login-container">
    <h1>Login</h1>
    {% if error %}
        <div class="alert alert-danger">{{ error }}</div>
    {% endif %}
    <form method="POST" action="{{ url_for('login')}}">
        <div class="mb-3">
            <label for="username" class="form-label">Username:</label>
            <input type="text" class="form-control" id="username" name="username" required>
        </div>
        <div class="mb-3">
            <label for="password" class="form-label">Password:</label>
            <input type="password" class="form-control" id="password" name="password" required>
        </div>
        <button type="submit" class="green-btn">Login</button>
    </form>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.min.js"></script>
</body>
</html>

```

```
{% extends 'components/base.jinja2' %}

{% block content %}
    <div class="d-flex align-items-center justify-content-between border-bottom">
        <h1 class="mx-3">Rules</h1>
        <button type="button" class="btn btn-secondary" data-bs-toggle="modal" data-bs-target="#modal"> Add new rule </button>
    </div>


    {% include 'components/modal/rules.jinja2' %}

    {% if list %}
        {% include 'components/table.jinja2' %}

    {% else %}
        <p> There are no rules! </p>
    {% endif %}

{% endblock %}

```

```
{% extends 'components/base.jinja2' %}

{% block content %}
    <div class="d-flex align-items-center justify-content-between border-bottom">
        <h1 class="mx-3">Alerts</h1>
    </div>

    {% if list %}
        {% include 'components/table.jinja2' %}

    {% else %}
        <p> There are no alerts! </p>
    {% endif %}

{% endblock %}

```

```
{% extends 'components/base.jinja2' %}

{% block content %}
    <div class="d-flex align-items-center justify-content-between border-bottom">
        <h1 class="mx-3">Blocked IPs</h1>
        <button type="button" class="btn btn-secondary" data-bs-toggle="modal" data-bs-target="#modal"> Add new IP </button>
    </div>


    {% include 'components/modal/blacklist.jinja2' %}


    {% if list %}
        {% include 'components/table.jinja2' %}

    {% else %}
        <p> There are no IPs blacklisted! </p>
    {% endif %}

{% endblock %}

```

File - D:\ORCH-IDS\app\templates\configurations.jinja2

```
{% extends 'components/base.jinja2' %}

{% block content %}
    <div class="d-flex align-items-center justify-content-between border-bottom">
        <h1 class="mx-3">Configurations</h1>
    </div>

    <div class="d-flex justify-content-center align-items-center">
        <div class="btn-group-vertical mx-1 my-1">
            <a class="btn btn-danger align-self-center" href="{{url_for('configurations')}}/setup_db" >Create or <br> Recreate DB</a>
        </div>
        <div class="btn-group-vertical mx-1">
            <a class="btn btn-danger my-1" href="{{ url_for('configurations')}}/drop_rules">Purge Rules</a>
            <a class="btn btn-danger my-1" href="{{ url_for('configurations')}}/populate_rules">Populate Rules</a>
        </div>
        <div class="btn-group-vertical mx-1">
            <a class="btn btn-danger my-1" href="{{ url_for('configurations')}}/drop_blacklist">Purge Blacklist</a>
            <a class="btn btn-danger my-1" href="{{ url_for('configurations')}}/populate_blacklist">Populate Blacklist</a>
        </div>
        <div class="btn-group-vertical mx-1">
            <a class="btn btn-danger my-1" href="{{ url_for('configurations')}}/drop_alerts">Purge Alerts</a>
            <a class="btn btn-danger my-1" href="{{ url_for('configurations')}}/populate_alerts">Populate Alerts</a>
        </div>
        <div class="btn-group-vertical mx-1">
            <a class="btn btn-danger my-1" href="{{
```

```
25 url_for('configurations')}}/drop_packets">Purge
   Packets</a>
26             <a class="btn btn-danger my-1" href="{{
   url_for('configurations')}}/populate_packets">
   Populate Packets</a>
27         </div>
28         <div class="btn-group-vertical mx-1">
29             <a class="btn btn-danger my-1" href="{{
   url_for('configurations')}}/drop_occurrences">Purge
   Occurrences</a>
30             <a class="btn btn-danger my-1" href="{{
   url_for('configurations')}}/populate_occurrences">
   Populate Occurrences</a>
31         </div>
32     </div>
33 {% endblock %}
34
35
```

File - D:\ORCH-IDS\app\templates\components\base.jinja2

```
<!DOCTYPE html>
<html>
<head>
    <title>{{ title }}</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='css/main.css') }}">
</head>
<body>
<div class="container">
    <h1 class="title">{{ title }} </h1>
    <header class="d-flex align-items-center justify-content-between py-3 mb-4 border-bottom">
        <span class="col-md-1"></span>
        <nav class="col-12 col-md-auto mb-2 d-flex justify-content-center">
            <ul class="nav">
                <li><a class="btn btn-primary green-btn" href="{{url_for('home')}}">Home</a></li>
                <li><a class="btn btn-primary green-btn" href="{{url_for('rules')}}">Rules</a></li>
                <li><a class="btn btn-primary green-btn" href="{{url_for('blacklist')}}">Blacklist</a></li>
                <li><a class="btn btn-primary green-btn" href="{{url_for('alerts')}}">Alerts</a></li>
                <li><a class="btn btn-primary green-btn" href="{{url_for('configurations')}}">Configurations</a></li>
            </ul>
        </nav>
        <div class="col-1 col-md-auto mb-2 d-flex justify-content-center">
            <a class="btn btn-primary green-btn" href="{{url_for('logout')}}">Logout</a>
        </div>
    </header>

</div>

```

```
29 <div class="align-items-center justify-content-
   between px-4 mx-4">
30     {% block content %}{% endblock %}
31 </div>
32 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5
   .3.0/dist/js/bootstrap.min.js"></script>
33 </body>
34 </html>
35
```

File - D:\ORCH-IDS\app\templates\components\table.jinja2

```
<div class="d-flex justify-content-center">
    <table class="table table-striped table-dark table-bordered table-responsive table-hover text-center table-fit">
        <thead>
        <tr>
            <th scope="col">#</th>
            {% for key in keys %}
                <th scope="col">{{key}}</th>
            {% endfor %}
            <th scope="col-md-1"></th>
        </tr>
        </thead>
        <tbody>

        {% for item in list %}
            <tr>
                <th class="col-md-1">{{loop.index}}</th>
                {% for key in keys %}
                    <td class="col-md-3">{{ item[key] }}</td>
                {% endfor %}
                <td class="col-md-1 align-items-center">
                    <button class="btn btn-danger col-md-auto align-items-center" value="{{item['_id']}}" onclick="deleteEntry(this)">Delete</button>
                </td>
            </tr>
        {% endfor %}
        </tbody>
    </table>
</div>

<script>
    function deleteEntry(btn){
        let row = btn.parentNode.parentNode;
        fetch('{{route}}', {
            method: 'DELETE',
            headers: {
```

```
35                  'Content-Type': 'application/json'
36              },
37              body: JSON.stringify({ '_id': btn.value
    })
38          })
39          row.remove();
40      }
41  </script>
```

```
<div class="modal fade" id="modal" tabindex="-1" aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">{{ modal_title }}</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
            </div>
            {% block form %}

            {% endblock %}
        </div>
    </div>
</div>
{% block validateForm %}

{% endblock %}
```

File - D:\ORCH-IDS\app\templates\components\modal\rules.jinja2

```
{% set modal_title='New Rule' %}

{% extends 'components/modal/modal.jinja2' %}

{% block form %}
    <div class="modal-body">
        <form id="newRule" method="POST" action="{{ url_for('rules')}}" onsubmit="return validateForm()">
            <div class="form-row d-flex justify-content-between mb-3">
                <div class="form-group col-md-8">
                    <label for="name" class="form-label">Unique Name:</label>
                    <input type="text" class="form-control" id="name" name="name" required>
                </div>
                <div class="form-group col-md-2">
                    <label for="severity" class="form-label">Severity:</label>
                    <input type="number" class="form-control" id="severity" name="severity" required min="0">
                </div>
            </div>

            <div class="form-group col-md-auto mb-3">
                <label for="description" class="form-label">Description:</label>
                <input type="text" class="form-control" id="description" name="description" required>
            </div>

            <div class="form-group form-check mb-3">
                <label class="form-check-label" for="direction">Check this if this rule if direction-sensitive</label>
                <input type="checkbox" class="form-check-input" id="direction" name="direction" value=true>
            </div>
```

```
28
29              <p>Leaving any field below empty will
    work as 'any' value</p>
30
31              <label for="source" class="form-label">
    Source:</label>
32              <div id="source" class="form-row d-flex
    justify-content-between mb-3 p-2 border">
33                  <div class="form-group col-md-8">
34                      <label for="source_ip" class="
    form-label">IP:</label>
35                      <input type="text" class="form-
    control" id="source_ip" name="source_ip">
36                  </div>
37                  <div class="form-group col-md-2">
38                      <label for="source_port" class="
    form-label">Port:</label>
39                      <input type="number" class="form-
    control" id="source_port" name="source_port" min="0"
    max="65536">
40                  </div>
41              </div>
42
43              <label for="destination" class="form-
    label">Destination:</label>
44              <div id="destination" class="form-row d-
    flex justify-content-between mb-3 p-2 border">
45                  <div class="form-group col-md-8">
46                      <label for="destination_ip" class
    ="form-label">IP:</label>
47                      <input type="text" class="form-
    control" id="destination_ip" name="destination_ip">
48                  </div>
49                  <div class="form-group col-md-2">
50                      <label for="destination_port"
    class="form-label">Port:</label>
51                      <input type="number" class="form-
    control" id="destination_port" name="destination_port
    " min="0" max="65536">
52                  </div>
53              </div>
```

```
54
55              <div class="form-row d-flex justify-
content-between mb-3">
56                  <div class="form-group col-md-4">
57                      <label for="protocol" class="form
-label">Protocol:</label>
58                      <select name="protocol" class="
form-control" id="protocol" form="newRule" onchange="
disableFlags(this)">
59                          <option selected value="">ANY
</option>
60                          <option value="TCP">TCP</
option>
61                          <option value="UDP">UDP</
option>
62                          <option value="ICMP">ICMP</
option>
63                      </select>
64
65                      <label for="flags" class="form-
label">TCP Flags:</label>
66                      <select name="flags" class="form-
control" id="flags" form="newRule" disabled="">
67                          <option selected value="">
NONE</option>
68                          <option value="S">SYN</option
>
69                          <option value="F">FIN</option
>
70                          <option value="R">RST</option
>
71                          <option value="P">PSH</option
>
72                          <option value="A">ACK</option
>
73                          <option value="U">URG</option
>
74                          <option value="E">ECE</option
>
75                          <option value="C">CWR</option
>
```

```
                    </select>
                </div>

                <div id="length" class="form-group
col-md-4">
                    <label for="minimum" class="form
-label">Payload Min length:</label>
                    <input type="number" class="form
-control" id="minimum" name="min_length" min="0">
                    <label for="maximum" class="form
-label">Payload Max length:</label>
                    <input type="number" class="form
-control" id="maximum" name="max_length" min="0">
                </div>
            </div>

            <label for="detection_filter" class="
form-label">Detection Filter:</label>
            <div id="detection_filter" class="form-
row d-flex justify-content-between mb-3 p-2 border">
                <div class="form-group col-md-4">
                    <label for="count" class="form-
label">Count:</label>
                    <input type="text" class="form-
control" id="count" name="count"  min="1">
                </div>
                <div class="form-group col-md-4">
                    <label for="interval" class="
form-label">Interval (Seconds):</label>
                    <input type="number" class="form
-control" id="interval" name="interval"  min="1">
                </div>

                <div class="form-group col-md-4">
                    <label for="track" class="form-
label">Track by:</label>
                    <select name="track" class="form
-control" id="track" form="newRule">
                        <option selected value="">
NONE</option>
                        <option value="by_src">
```

```
102 Source</option>
103                             <option value="by_dst">
    Destination</option>
104                             <option value="both">Both</
    option>
105                         </select>
106                     </div>
107                 </div>
108                 <button type="submit" class="btn green-
    btn">Submit</button>
109             </form>
110         </div>
111 {% endblock %}
112 {% block validateForm %}
113     <script>
114 
115         function disableFlags(protocol) {
116             let flags = document.forms["newRule"]["
    flags"];
117             if(protocol.value === 'TCP'){
118                 flags.disabled = false
119                 return
120             }
121             flags.disabled = true
122             flags.value=""
123 
124         }
125 
126         function validateForm() {
127 
128             let name = document.forms["newRule"]["
    name"].value
129             {% if list %}
130                 {% for i in list %}
131                     if (name === "{{i['name']}}"){
132                         alert("Invalid Name!")
133                         return false
134                     }
135                 {% endfor %}
136             {%endif %}
137 
```

```
138             let src_ip = document.forms["newRule"]["
    source_ip"].value.trim();
139             let dst_ip = document.forms["newRule"]["
    destination_ip"].value.trim();
140 
141             let ip_regex = /^((?:[0-9a-fA-F]{1,4}:){
    7}[0-9a-fA-F]{1,4})|((\d{1,3}\.){3}\d{1,3})$/;
142 
143             let valid_src_ip = src_ip === "" ||
    ip_regex.test(src_ip);
144             let valid_dst_ip = dst_ip === "" ||
    ip_regex.test(dst_ip);
145             if (!valid_src_ip || !valid_dst_ip){
146                 alert("Invalid IP format");
147                 return false;
148             }
149 
150             let min_length = parseInt(document.forms
    ["newRule"]["min_length"].value);
151             let max_length = parseInt(document.forms
    ["newRule"]["max_length"].value);
152             if (max_length < min_length){
153                 alert("Payload max length can't be
    lesser than min length")
154                 return false;
155             }
156 
157             let count = document.forms["newRule"]["
    count"].value;
158             let interval = document.forms["newRule"
    ]["interval"].value;
159             console.log(count, interval)
160             if ((!count && interval) || (count && !
    interval))
161             {
162                 alert("Count and Interval Must both
    have a value or none at all")
163                 return false;
164             }
165 
166             let track = document.forms["newRule"]["
```

```
166 track"].value;
167             console.log(track)
168             if(track && !count){
169                 alert("Track can only be set if the
    interval and count are set as well")
170                 return false;
171             }
172         }
173     </script>
174 {% endblock %}
```

```
{% set modal_title='Blacklist new IP' %}

{% extends 'components/modal/modal.jinja2' %}

{% block form %}
    <div class="modal-body">
        <form id="addToBlacklist" method="POST" action="{{ url_for('blacklist')}}" onsubmit="return validateForm()">
            <div class="form-row d-flex justify-content-between mb-3">
                <div class="form-group col-md-8">
                    <label for="ip" class="form-label">Unique IP:</label>
                    <input type="text" class="form-control" id="ip" name="ip" required>
                </div>
                <div class="form-group col-md-2">
                    <label for="ip_version" class="form-label">Version:</label>
                    <select name="ip_version" class="form-control" id="ip_version" required form="addToBlacklist">
                        <option selected value=4>4</option>
                        <option value=6>6</option>
                    </select>
                </div>
            </div>
            <div class="form-row d-flex justify-content-between mb-3">
                <div class="form-group col-md-8">
                    <label for="reason" class="form-label">Reason:</label>
                    <input type="text" class="form-control" id="reason" name="reason" required>
                </div>

                <div class="form-group col-md-2">
                    <label for="severity" class="form-label">Severity:</label>
```

```
29                      <input type="number" class="form-
   control" id="severity" name="severity" required min="
   0">
30                  </div>
31              </div>
32              <button type="submit" class="btn green-
   btn">Submit</button>
33          </form>
34      </div>
35 {% endblock %}
36 {% block validateForm %}
37      <script>
38          function validateForm() {
39
40              let ip = document.forms["addToBlacklist"
   ]["ip"].value
41              {% if list %}
42                  {% for i in list %}
43                      if (ip === "{{i['ip']}}"){
44                          alert("IP already blacklisted
   !")
45                          return false
46                      }
47                  {% endfor %}
48              {%endif %}
49
50              let version = document.forms["
   addToBlacklist"]["ip_version"].value;
51
52              let regex = /^(?:[0-9a-fA-F]{1,4}:){7}[0-
   9a-fA-F]{1,4}$/;
53              if (version === "4") {
54                  regex =  /^(\d{1,3}\.){3}\d{1,3}$/;
55              }
56
57              // Verificar se o valor do campo
   corresponde ao formato do IPv6
58              if (!regex.test(ip)){
59                  alert("Invalid IP format");
60                  return false;
61              }
```

```
        }
    </script>
{% endblock %}
```

File - D:\ORCH-IDS\agent\agent.py

```python
from pymongo import MongoClient
from dotenv import dotenv_values
from scapy.all import *
from datetime import datetime

config = dotenv_values()

mongo_db = config['MONGO_DB']
client = MongoClient(host=config['MONGO_HOST'], port=int(config['MONGO_PORT']))
db = client[mongo_db]
collection = db[config['MONGO_COLLECTION_QUEUE']]
queue = []

interface = config['NETWORK_INTERFACE']

# IANA protocol codes to names, refer to https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml
protocols = {
    1: 'ICMP',
    6: 'TCP',
    17: 'UDP'
}


def process_packet(packet):

    # We only want IP packets
    if IP not in packet:
        return

    # Ethernet header
    packet_info = {
        "captured_by": get_if_addr(interface),
        "timestamp": int(datetime.now().timestamp() * 1000),
    }

    # Unpack
    packet = packet.payload
```

```
38
39      # extract protocol number, IPv6 calls this field
    as nextHeader, that is why packet.nh
40      protocol = packet.proto if packet.version == 4
    else packet.nh
41
42      # We only support ICMP, TCP and UDP
43      if protocol not in protocols:
44          return
45
46      # IPv4/IPv6
47      packet_info["source_ip"] = packet.src
48      packet_info["destination_ip"] = packet.dst
49      packet_info["ip_version"] = packet.version
50      packet_info["protocol"] = protocols[protocol]
51
52      # IPv4 has a len attribute, IPv6 has a plen
    attribute
53      packet_info["length"] = packet.len if packet.
    version == 4 else packet.plen
54
55      # Unpack
56      packet = packet.payload
57
58      # Extract ports if there are
59      if hasattr(packet, "sport") and hasattr(packet, "
    dport"):
60          packet_info["source_port"] = packet.sport
61          packet_info["destination_port"] = packet.
    dport
62
63      # Extract flags from TCP
64      if TCP in packet:
65          packet_info["flags"] = [flag for flag in
    packet.flags]
66
67      # Enqueue and save queue
68      queue.append(packet_info)
69      save_packet(queue)
70
71
```

```python
def save_packet(queue):
    if len(queue) == 10:
        collection.insert_many(queue)
        queue.clear()


def capture_packets():

    # Iniciando a captura de pacotes
    sniff(iface=interface, prn=process_packet, store=False)


# Executando a captura de pacotes
capture_packets()

```

```
MONGO_HOST='mongodb://127.0.0.1'
MONGO_PORT=27017
MONGO_DB='NIDS'
MONGO_COLLECTION_QUEUE='packets'
NETWORK_INTERFACE='Ethernet'
```

```
1 scapy==2.5.0
2 pymongo==4.3.3
3 python-dotenv==1.0.0
4
```

```python
from datetime import datetime

from pymongo import MongoClient
from dotenv import dotenv_values

from apscheduler.schedulers.background import BackgroundScheduler


def create_incident(name, packet, severity=None):
    new_alert = {
        'name': name,
        'timestamp': packet['timestamp'],
        'protocol': packet['protocol'],
        'source_ip': packet['source_ip'],
        'destination_ip': packet['source_ip'],
        'length': packet['length']
    }

    if severity:
        new_alert['severity'] = severity

    if 'source_port' in packet:
        new_alert['source_port'] = packet['source_port']
        new_alert['destination_port'] = packet['destination_port']

    return new_alert


class Enforcer:
    def __init__(self):
        config = dotenv_values()

        mongo_db = config['MONGO_DB']
        self.client = MongoClient(host=config['MONGO_HOST'], port=int(config['MONGO_PORT']))
        db = self.client[mongo_db]

        self.rules = db[config['
```

```
37 MONGO_COLLECTION_RULES']]
38         self.alerts = db[config['
   MONGO_COLLECTION_ALERTS']]
39         self.packets = db[config['
   MONGO_COLLECTION_QUEUE']]
40         self.blacklist = db[config['
   MONGO_COLLECTION_BLACKLISTED']]
41         self.occurrences = db[config['
   MONGO_COLLECTION_OCCURRENCES']]
42 
43         # Should not evaluate these fields:
44         self.ignored_fields = ['name', 'severity', '
   description', 'direction', 'count', 'interval', '
   track']
45 
46         self.rule_set = []
47         self.complex_rule_set = {}  # for rules that
   have count, interval and track attributes
48 
49         self.blacklist_set = {}  # blacklisted IPs (
   key is Ip and value is severity)
50 
51     def fetch_rules(self):
52         rules_list = list(self.rules.aggregate([
53             {'$project': {'_id': 0}},
54             {'$sort': {'severity': -1}}
55         ]))
56 
57         new_complex_rules = {}
58         new_rules = []
59         for r in rules_list:
60             # check if a rule has count and interval
    (it can have a track attribute as well)
61             # if it has then it is a complex rule and
    should be evaluated a little different
62             if 'count' in r and 'interval' in r:
63                 new_complex_rules[r['name']] = {k: v
   for k, v in r.items() if k in ['name', 'count', '
   interval', 'track', 'severity']}
64 
65             if not r['direction']:
```

```
 66                 new_rule = r.copy()
 67
 68                 keys_to_delete = ['source_ip', '
    destination_ip', 'source_port', 'destination_port']
 69                 for k in keys_to_delete:
 70                     if k in new_rule:
 71                         del new_rule[k]
 72
 73                 if 'source_ip' in r:
 74                     new_rule['destination_ip'] = r['
    source_ip']
 75                 if 'destination_ip' in r:
 76                     new_rule['source_ip'] = r['
    destination_ip']
 77                 if 'source_port' in r:
 78                     new_rule['destination_port'] = r
    ['source_port']
 79                 if 'destination_port' in r:
 80                     new_rule['source_port'] = r['
    destination_port']
 81                 new_rules.append(new_rule)
 82
 83         self.rule_set = rules_list + new_rules
 84         self.complex_rule_set = new_complex_rules
 85
 86     def fetch_blacklist(self):
 87         self.blacklist_set = {
 88             x['ip']: x['severity'] for x in list(
 89                 self.blacklist.find(
 90                     {},
 91                     projection={
 92                         '_id': 0,
 93                         'ip': 1,
 94                         'severity': 1
 95                     }
 96                 )
 97             )
 98         }
 99
100     def fetch_packets(self, batch_size=10):
101         with self.client.start_session() as session:
```

```
102             with session.start_transaction():
103                 packet_list = list(self.packets.find
    ({}, sort=[('timestamp', 1)], limit=batch_size))
104                 self.packets.delete_many({"_id": {'
    $in': [x['_id'] for x in packet_list]}})
105                 return packet_list
106 
107     def save_alerts(self, alerts_list):
108         if len(alerts_list):
109             self.alerts.insert_many(alerts_list)
110 
111     def save_occurrences(self, occurrences_list):
112         if len(occurrences_list):
113             self.occurrences.insert_many(
    occurrences_list)
114 
115     def verify(self, packet, rule):
116         for k, v in rule.items():
117             if k in self.ignored_fields:
118                 continue
119 
120             if k == 'flags':
121                 if 'flags' not in packet or v not in
     packet['flags']:
122                     return False
123             elif k == 'min_length':
124                 if v < packet['length']:
125                     return False
126             elif k == 'max_length':
127 
128                 if v > packet['length']:
129                     return False
130             elif v != packet[k]:
131                 return False
132         return True
133 
134     def check_blacklist(self, packet):
135         blacklist_alert = []
136         if packet['source_ip'] in self.blacklist_set
    :
137             blacklist_alert.append(create_incident('
```

```
137 BLACKLISTED_IP_SRC',
138 
    packet,
139 
    self.blacklist_set[packet['source_ip']]))
140 
141         if packet['destination_ip'] in self.
    blacklist_set:
142             blacklist_alert.append(create_incident('
    BLACKLISTED_IP_DST',
143 
    packet,
144 
    self.blacklist_set[packet['destination_ip']]))
145 
146         return blacklist_alert
147 
148     def alert_complex_rules(self):
149         new_alerts = []
150         for rule in self.complex_rule_set.values():
151 
152             with self.client.start_session() as
    session:
153                 with session.start_transaction():
154 
155                     # Group by interval
156                     group = {'timestamp': {
157                         '$dateTrunc': {
158                             'date': {
159                                 '$toDate': '
    $timestamp'
160                             },
161                             'unit': 'second',
162                             'binSize': int(rule['
    interval'])
163                         }
164                     }
165                     }
166 
167                     if 'track' in rule:
168                         if rule['track'] == 'both'
```

```
168 or rule['track'] == 'by_dst':
169                                 group['destination_ip'
    ] = '$destination_ip'
170 
171                             elif rule['track'] == 'both'
     or rule['track'] == 'by_src':
172                                 group['source_ip'] = '
    $source_ip'
173 
174                         occurrences = self.occurrences.
    aggregate([
175                             {
176                                 '$match': {
177                                     'name': rule['name']
178                                 }
179                             }, {
180                                 '$group': {
181                                     '_id': group,
182                                     'count': {
183                                         '$sum': 1
184                                     },
185                                     'packet': {
186                                         '$last': '
    $$CURRENT'
187                                     }
188                                 }
189                             }, {
190                                 '$match': {
191                                     'count': {
192                                         '$gte': int(rule
    ['count'])
193                                     }
194                                 }
195                             }
196                         ])
197 
198                         for occurrence in occurrences:
199                             new_alerts.append(
    create_incident(occurrence['packet']['name'],
    occurrence['packet'], rule['severity']))
200 
```

```
201                     # We can simply delete everyone
    that has its timestamp lesser than (now - interval)
202                     self.occurrences.delete_many({'
    name': rule['name'], 'timestamp': {'$lte': int(
    datetime.now().timestamp() * 1000) - int(rule['
    interval']) * 1000}})
203             self.save_alerts(new_alerts)
204 
205 
206 if __name__ == '__main__':
207     enforcer = Enforcer()
208 
209     scheduler = BackgroundScheduler()
210     # Refresh rules and blacklist every 5 minutes
211     scheduler.add_job(enforcer.fetch_rules, '
    interval', minutes=5)
212     scheduler.add_job(enforcer.fetch_blacklist, '
    interval', minutes=5)
213 
214     scheduler.add_job(enforcer.alert_complex_rules,
    'interval', minutes=1)
215 
216     enforcer.fetch_rules()
217     enforcer.fetch_blacklist()
218 
219     scheduler.start()
220 
221     try:
222         while True:
223             new_alerts = []
224             new_occurrences = []
225             for packet in enforcer.fetch_packets():
226 
227                 # Check if any Ip (Source,
    Destination) is blacklisted
228                 new_alerts += enforcer.
    check_blacklist(packet)
229 
230                 # Try rules in a severity descending
     order stopping at first match
231                 for rule in enforcer.rule_set:
```

```
232
233                     if enforcer.verify(packet, rule
    ):
234
235                         # check if it is a complex
    rule or not:
236                         if rule['name'] not in
    enforcer.complex_rule_set:
237                             # In this case just
    alert
238                             new_alerts.append(
    create_incident(rule['name'], packet, rule['severity
    ']))
239                         else:
240                             # Complex rule case:
241                             new_occurrences.append(
    create_incident(rule['name'], packet))
242                         break
243
244             # Saving
245             enforcer.save_occurrences(
    new_occurrences)
246             enforcer.save_alerts(new_alerts)
247
248     except (KeyboardInterrupt, SystemExit):
249         scheduler.shutdown()
```

```
MONGO_HOST='mongodb://127.0.0.1'
MONGO_PORT=27017
MONGO_DB='NIDS'
MONGO_COLLECTION_QUEUE='packets'
MONGO_COLLECTION_RULES='rules'
MONGO_COLLECTION_BLACKLISTED='blacklist'
MONGO_COLLECTION_ALERTS='alerts'
MONGO_COLLECTION_OCCURRENCES='occurrences'

```

```
pymongo==4.3.3
python-dotenv==1.0.0
six==1.16.0
tzdata==2023.3
tzlocal==5.0.1
pytz==2023.3
APScheduler==3.10.1

```

ANEXO C – ARTIGO DO TCC

ORCH-IDS - Um sistema modular de Detecção de Intrusos na rede baseado em regras

Yuri Ferreira Bittencourt[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88040-370 – Florianópolis – SC – Brasil

Abstract. *The expansion and popularization of the Internet have brought many benefits to our society, however, it has become commonplace for cyber-attacks. according to CrowdStrike[Baker 2023], the second most common cyber-attack is the Denial of Service attack, also known as DoS, and that includes its distributed variant, DDoS, and this is merely an example of an attack that can be detected by a rule-based network intrusion detection system. The main objective of this study is to develop a scalable prototype of a rule-based network intrusion detection system that supports multiple network packet-capture sensors and has a web interface for system configurations and data visualization of generated alerts.*

Resumo. *A expansão e popularização da Internet trouxeram benefícios significativos à sociedade, mas também aumentaram a incidência de ataques cibernéticos. Entre os tipos mais comuns de ataques segundo a CrowdStrike [Baker 2023], destaca-se o ataque de Negação de Serviço (DoS), incluindo a variante distribuída (DDoS). Um sistema de detecção de intrusão na rede baseado em regras pode detectar esses ataques. O objetivo deste trabalho é desenvolver um protótipo escalável de um sistema de detecção de intrusos na rede baseado em regras, capaz de lidar com múltiplos sensores de captura de pacotes de rede e fornecer uma interface web para configuração e visualização de dados e alertas gerados pelo sistema.*

1. Introdução

Os ataques e tentativas de ataques cibernéticos estão cada vez mais frequentes e especificamente no Brasil em 2022 houve um aumento de 94% dos incidentes comparando o primeiro semestre de 2022 (31,5bi de incidentes) com o mesmo período no ano anterior (16,2bi de incidentes) segundo a CNN Brasil[Oliveira 2022] e é esperado que com o tempo os incidentes de cibersegurança tendam a aumentar.

No começo do ano de 2023, a CrowdStrike compilou um documento [Baker 2023] relatando os 10 tipos de ciberataques mais comuns, nesta listagem os ataques de Negação de serviço (DoS) e Negação de serviço distribuída (DDoS) figuram em segundo lugar e a importância de mencioná-los é que esse tipo de ataque pode ser detectado por regras configuradas em um IDS, nessa lista também há outros tipos de ataques passíveis dessa análise.

É evidente que, com base nas informações previamente apresentadas, existe a necessidade de melhorar e ampliar os sistemas de segurança cibernética e isso vai de encontro com a proposta deste trabalho que apresentará um protótipo (prova de conceito) de um sistema de detecção de intrusão na rede distribuído baseado em regras com múltiplos agentes sendo altamente customizável e com interface web.

2. Justificativa

Buscando implementar um sistema de detecção de intrusão na rede baseado em regras de forma descomplicada, com múltiplos sensores e com boa escalabilidade (processamento distribuído) foi percebido uma lacuna devido a sistemas similares como o Snort [SNORT] que permite múltiplos sensores, ainda que a configuração dos mesmos não seja tão simples, tem o processamento totalmente centralizado e isso permite a realização deste projeto como prova de conceito. Como mencionado na seção anterior deste mesmo capítulo existe uma crescente demanda para ampliação e melhoramento de sistemas de segurança cibernética e o protótipo desenvolvido neste trabalho pode ser útil neste objetivo.

Este trabalho também é motivado por interesses pessoais e é esperado que o sistema desenvolvido neste trabalho possa ampliar o conhecimento deste tipo de tecnologia.

3. Objetivos

3.1. Objetivo Geral

O Objetivo geral deste trabalho é o desenvolvimento de um protótipo de sistema de detecção de intrusão na rede baseado em regras, assim como o Snort [SNORT] com múltiplos sensores, modularizado com suporte a interface web e que possa ser facilmente implantado e altamente escalável afim de que possa atuar detectando possíveis ataques em uma rede e sendo possível configurar suas regras de detecção e IPs a serem monitorados.

3.2. Objetivos Específicos

Os Objetivos específicos deste trabalho incluem características do desenvolvimento da ferramenta como: independência entre os módulos de forma que o sistema possa funcionar independentemente da quantidade de agentes de sensoreamento como também de unidades de processamento de regras, o protótipo deve estar funcional e suportando a criação de regras simples e complexas que são regras que envolvem um período e número de casos para gerar alertas.

4. Fundamentação Teórica

4.1. Sistema de detecção de Intrusão

O Sistema de detecção de Intrusão (do inglês: *Intrusion Detection System*) é uma tecnologia de segurança em rede pensada em detectar possíveis ataques contra uma aplicação ou computador alvo. Diferentemente de um sistema de prevenção, o IDS apenas detecta e alerta, funcionando assim de forma passiva apenas monitorando tráfego e reportando dados a um administrador ou sistema, segundo [Palo Alto Networks]

4.2. Sistema de detecção de Intrusão baseado em Rede

Esta variação do IDS, também conhecido como NIDS, consiste em um sistema que diferentemente do IDS que era voltado a uma aplicação ou computador específico é direcionado operar sobre a rede e que irá monitorar o seu tráfego, conforme [STALLINGS 2012, Capítulo 8.5], neste mesmo capítulo é citado o funcionamento típico de um NIDS consistindo em:

- vários sensores para capturar pacotes de rede;

- um ou mais servidores para funções de gerenciamento do sistema;
- um ou mais consoles de gerenciamento com interface para o operador.

Ainda neste mesmo assunto, é importante mencionar sobre os tipos de sensores que podem ser *Inline* ou passivos e conforme [STALLINGS 2012, Capítulo 8.5]: o *inline* é um sensor que está interceptando o tráfego e pelo seu funcionamento pode ser usado para prevenção de ataques, entretanto ele pode causar latência na rede; O tipo passivo é mais comumente usado para IDS pois não afeta a rede diretamente uma vez que ele apenas monitora uma cópia do tráfego e não intercepta pacotes.

4.3. Sistema de detecção de Intrusão baseado em Regras

Existem várias formas para um IDS detectar algum evento, a forma interessante para este trabalho é a detecção baseada em regras assim como funciona o sistema Snort [SNORT] onde segundo a própria documentação do Snort [Mansour 2020] as regras seguem o formato: "Action Protocol Networks Ports Direction_Operator Networks Ports".

O primeiro campo *action* seria qual a ação para ser tomada e isso é interessante no contexto de um sistema de prevenção de Intrusão (IPS), em nosso caso todas as ações do sistema proposto seriam equivalentes a *action alert*, os demais campos em tradução aberta são:

- *Protocol*: Protocolo do remetente;
- *Networks*: Rede do remetente, este seria o endereço;
- *Ports*: Porta do remetente;
- *Direction_Operator*: Direção da regra, unidirecional ou bidirecional;
- *Networks*: Rede do destinatário, endereço do destinatário;
- *Ports*: Porta do destinatário

Perceba que uma regra bidirecional perde um pouco do sentido de remetente e destinatário mas foi mantido na explicação para diferenciar os campos, contudo uma regra bidirecional pode ser vista como duas regras unidirecionais com os campos *Networks* e *Ports* invertidos

5. Trabalhos Relacionados

Existem trabalhos que apresentam a integração entre IDS e a arquitetura SDN, o autor [Francisco 2019] propõe um estudo sobre a arquitetura SDN com o protocolo OpenFlow para integrar com um sistema de detecção de intrusão apresentando um ambiente simulado integrando essas tecnologias para mitigação de ataques, não apresenta um IDS propriamente mas uma integração com, para efeitos de teste e validação num ambiente controlado este trabalho é adequado.

O próprio projeto do Snort [SNORT] é um trabalho correlato por ser um sistema de detecção e prevenção de intrusão na rede baseado em regras *open source* e foi uma forte inspiração para este trabalho, o Snort tem vários modos de operação e isso permite ele atuar como um IPS ou IDS dependendo das configurações, o Snort permite múltiplos sensores mas no entanto o seu processamento é sempre centralizado.

No trabalho [Kumar et al. 2020] foi apresentado um NIDS baseado em regras e o sistema proposto utiliza uma abordagem de detecção de assinatura para identificar diferentes tipos de ameaças em uma rede. Este trabalho também fez um comparativo de desempenho com outros modelos existentes, entretanto aqui o IDS proposto possui processamento centralizado.

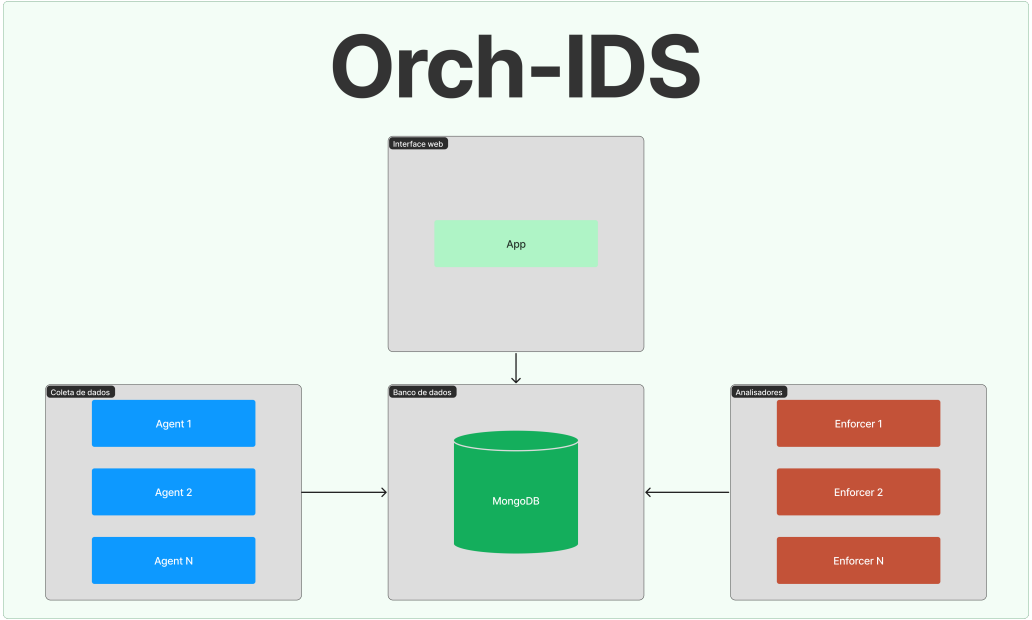
O trabalho [Júnior 2016] é um trabalho fortemente relacionado à este, a proposta é similar e caso tivesse o encontrado antes seria base para este trabalho pois apresenta uma estrutura similar, multi-agentes coletores e processamento podendo ser distribuído. Também é um IDS em rede baseado em regras.

6. ORCH-IDS

6.1. Arquitetura

O sistema é dividido em 3 serviços que se comunicam apenas com uma instância ou cluster do MongoDB, dessa forma qualquer parte do sistema pode ser substituída ou refeita desde que respeitada as entradas e saídas desse serviço para com o banco.

Figure 1. Arquitetura do Sistema ORCH-IDS



Perceba que na figura 1 há múltiplos *Agents* e *Enforcers*, isso é só um exemplo de que o sistema aceita quantas instâncias desses serviços forem desejadas, o mesmo vale para o *App* apesar de não haver muita utilidade em mais de uma instância para a interface gráfica.

6.2. *Agent*

Este serviço é responsável pela coleta de dados para o sistema, funciona como um *sniffer*, esse sistema só captura pacotes encapsulado por IP (versão 4 ou 6) por motivos de simplificação para prototipação, envia esses dados em grupos de 10 para a coleção de pacotes no banco de dados, a configuração desse serviço é feita no arquivo *.env* próprio.

No código fonte há um arquivo *.env.example* que é um arquivo de exemplo que basicamente tem o endereço do banco e a porta, nome da *database*, coleção do banco e principalmente qual a interface de rede que deve ser usada para monitorar o tráfego,

importante salientar que esta interface deve ser configurada em modo promíscuo caso queira monitorar todo tráfego da rede ou não caso queira apenas monitorar o tráfego para esta máquina.

6.3. *App*

Este serviço é responsável por prover uma interface web para o sistema desenvolvido em *Flask*, aqui também estão os utilitários que criam o banco de dados, índices e populam ou esvaziam as coleções, assim como o *Agent*, também possui um arquivo de configuração chamado *.env* próprio. Assim como no *Agent* há a configuração de conexão ao banco de dados, aqui também temos todas as coleções, nome e senha de usuário da aplicação, porta, chave de encriptação e principalmente o ambiente (*DEV* ou *PROD*)

6.4. *Enforcer*

Este serviço é responsável por pegar dados da coleção de pacotes (que funciona como fila) e verificar se esses pacotes violam alguma regra ou o IP (destino ou origem) desses pacotes constam na lista negra, gera alertas em casos positivos. A cada 5 minutos este serviço acessa o banco de dados para verificar se houveram alterações no conjunto de regras ou conjunto de IPs monitorados (lista negra) e a cada 1 minuto existe uma verificação na coleção de *occurrences* que testa por regras 'complexas' que são regras que só geram alertas se ocorrerem *x* vezes num intervalo de *y* segundos. Assim como os outros serviços, as configurações desse serviço ficam em um arquivo *.env*.

6.5. Funcionamento

O funcionamento da aplicação como um todo foi brevemente explicado anteriormente em cada subseção da arquitetura do sistema, aqui entrarei em detalhes de algumas decisões de desenvolvimento e o motivo disso. Uma decisão tomada pensando em otimização foi que a verificação das regras testará as regras na ordem de severidade (maior para menor) e irá parar na primeira regra mais grave a ter correspondência, isso pode ser facilmente alterado mas para efeitos de prototipagem está funcional e eficiente. Uma limitação deste sistema são os possíveis pacotes duplicados que é um problema passível de ocorrer quando múltiplos *agents* estão monitorando a rede e existe alguma sobreposição no alcance do monitoramento, dessa forma irão ser capturados pacotes duplicados e para remediar isso foi incluído o atributo *'captured_by'* na coleção dos pacotes, desta forma é mais fácil de diferenciar se realmente foi um pacote duplicado por algum possível ataque ou monitoramento sobreposto mas isto ficará para um trabalho futuro e pode ser evitado com o correto posicionamento dos *agents*. O *Enforcer* ao buscar as regras da coleção, irá dividir as regras em regras normais e regras complexas assim fazendo uso da coleção de *occurrences*, desta forma é possível que mesmo que várias instâncias desse serviço estejam trabalhando concorrentemente, essas regras que necessitam de contagem possam ser feitas sem problemas uma vez que toda ocorrência dessa regra é salva na coleção e a cada tanto tempo uma dessas instâncias irá verificar essa coleção para gerar alerta caso passe do valor de contagem no intervalo estipulado. O *Enforcer* ainda faz um parse de regras bidirecionais para dividir em duas unidirecionais para simplificar a verificação das regras. Basicamente no Código é feita a verificação de cada campo da regra que está definido ignorando alguns campos que não são necessários para a validação e então verifica especificamente campos que possuem lógicas diferentes como as flags que verifica se o valor está contido no conjunto, tamanho mínimo e máximo do payload que faz a verificação de intervalo e depois a

verificação genérica se os valores correspondem, esse método retorna verdadeiro ou falso para a regra, se a regra a ser verificada for uma regra simples, ou seja, que não envolve múltiplas ocorrências em um período de tempo, irá gerar um alerta, caso contrário irá salvar essa ocorrência na coleção de ocorrências que é verificada a cada minuto se extrapola o número de ocorrências no intervalo de tempo de cada regra e assim gera um alerta.

7. Considerações Finais

Ao longo deste trabalho foi desenvolvido um protótipo de um IDS funcional que conseguiu satisfazer seus requisitos funcionais, permitindo regras customizáveis e monitoramento de IPs suspeitos, visualização e exclusão de alertas, permite múltiplos *agents* coletores e *Enforcers* para validar as regras, sendo assim um Sistema de Detecção de Intrusão na Rede baseado em regras e distribuído, um DRNIDS (*Distributed Rule-based NIDS*), além disso o sistema permite a substituição de qualquer um de seus serviços (*Agent, Enforcer, App*) tornando assim um sistema agnóstico a tecnologia e de certa forma vinculando apenas banco que opera sobre, o MongoDB.

7.1. Trabalhos Futuros

Para os trabalhos futuros, seria possível ampliar o *Enforcer* para que ele tenha suporte à *Machine Learning* permitindo assim uma análise mais complexa e completa; Analisar o comportamento do sistema num cenário real (empresarial) para ter métricas da escalabilidade horizontal do sistema.

References

Baker, K. (2023). *10 Most Common Types of Cyber Attacks*. CrowdStrike.

Francisco, C. (2019). Integração entre rede definida por software e sistema de detecção de intrusão para mitigação de ataques.

Júnior, J. F. G. S. (2016). Storm ids: um sistema de detecção de intrusão escalável e distribuído.

Kumar, V., Sinha, D., Das, A. K., Pandey, S. C., and Goswami, R. T. (2020). An integrated rule based intrusion detection system: analysis on unsw-nb15 data set and the real time online dataset.

Mansour, Y. (2020). *Rules Authors Introduction to Writing Snort 3 Rules*.

Oliveira, I. (2022). Levantamento mostra que ataques cibernéticos no brasil cresceram 94%.

Palo Alto Networks. *What is an Intrusion Detection System?* Palo Alto Networks.

SNORT. The snort project.

STALLINGS, W. (2012). *Segurança de Computadores*. Elsevier Editora Ltda., 2ª edição edition.