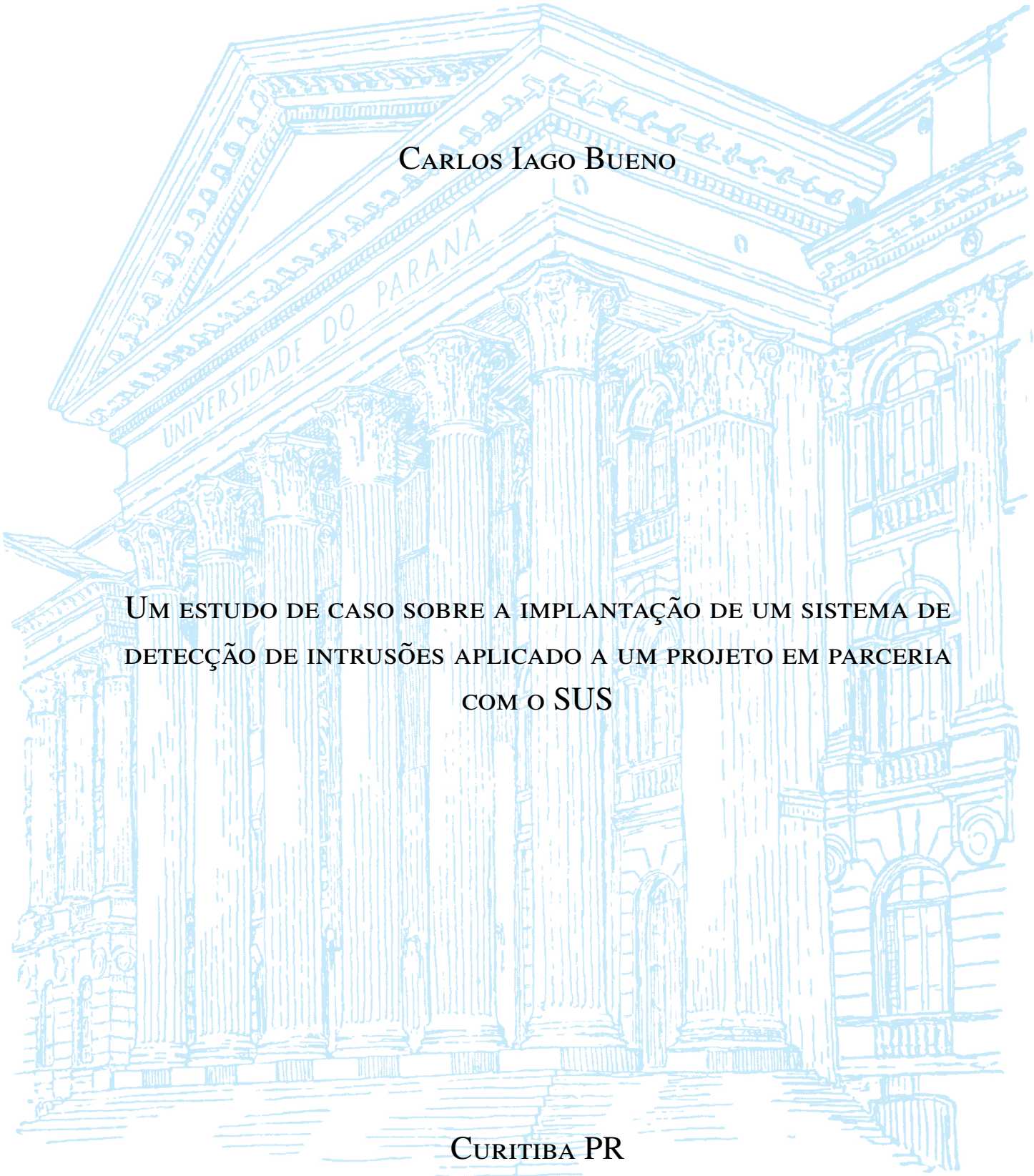
Universidade Federal do Paraná

Carlos Iago Bueno



Um estudo de caso sobre a implantação de um sistema de detecção de intrusões aplicado a um projeto em parceria com o SUS

Curitiba PR
2024

Carlos Iago Bueno

Um estudo de caso sobre a implantação de um sistema de detecção de intrusões aplicado a um projeto em parceria com o SUS

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Andre Ricardo Abed Gregio.

Curitiba PR
2024

Ficha catalográfica

Substituir o arquivo 0-iniciais/catalografica.pdf (PDF em formato A4) pela ficha catalográfica fornecida pela Biblioteca da UFPR a pedido da secretaria do PPGInf/ UFPR.

O conteúdo exato da ficha catalográfica é preparado pela Biblioteca Central da UFPR, a pedido da secretaria do PPGINF. Portanto, não "invente" um conteúdo para ela.

ATENÇÃO: por exigência da Biblioteca da UFPR, esta ficha deve ficar no verso da folha de rosto (que contém o nome do orientador e área de concentração). Cuide desse detalhe quando imprimir as cópias finais.

Ficha de aprovação

Substituir o arquivo 0-iniciais/aprovacao.pdf pela ficha de aprovação fornecida pela secretaria do programa, em formato PDF A4.

Que deus perdoe essas pessoas ruins...

Agradecimentos

Quero agradecer primeiro à Mylena, amor da minha vida. Obrigado por ter caminhado comigo nos últimos seis anos. Obrigado por dar sentido e significado às atribuições do dia a dia. Obrigado por sempre estar comigo, por nunca desistir de mim e por me amar de um jeito tão sincero. Espero passar o resto da minha vida junto contigo.

A minha mãe, que não apenas me presenteou com a vida, mas também abdicou de muitos sonhos dela para que eu pudesse viver os meus. Quero agradecer não apenas pelos grandes sacrifícios, mas pelos pequenos também. Agradecer os pequenos gestos de cuidado, pelo carinho e pelo zelo.

Ao meu pai, que não está mais conosco para presenciar este momento, mas sei que vibra por mim onde quer que esteja.

Aos meus irmãos, que sempre me deram força. Especialmente ao Andrey e a Ana que partilharam não somente a mesma casa comigo, mas as dificuldades, os momentos de alegria e a mesa. Sem vocês eu não teria chegado tão longe.

Ao Lüders e ao Vudala, os maiores companheiros que encontrei dentro da universidade. Estiveram ao meu lado nas piores e nas melhores fases da graduação. São duas pessoas que eu posso contar para qualquer situação e pretendo cultivar esta amizade para o resto da vida. Vocês são os irmãos que eu escolhi.

Aos amigos Tikara, Raul, Elson, Matheus, Christian e Fernando, que contribuíram de maneira direta ou indireta no desenvolvimento no projeto e na caminhada. Um agradecimento especial ao Tikara, por me ajudar inúmeras vezes ao revisar os textos e ao Fernando por me ajudar nas vezes que eu estava travado no projeto.

Ao Centro de Computação Científica e Software Livre, meu lar na universidade. O C3SL não somente é o maior responsável pela minha permanência na Universidade, como também é o ambiente que mais me desenvolveu como estudante, como pessoa e como cidadão. Agradeço também a todos os colegas de laboratório do C3SL que tornaram as manhãs mais leves e alegres.

Ao professor Grégio, pela orientação, pela paciência e pela confiança. Obrigado por não ter desistido. Agradeço também a todos os colegas da Secret que participaram de alguma forma.

Resumo

Empresas, órgãos do governo, organizações e instituições em todos os âmbitos investem recursos para empregar medidas de proteção às suas informações contra acesso não autorizado. Isso porque os dados foram se tornando ativos de grande importância para tomar decisões estratégicas, alocar recursos e até mesmo garantir soberania entre governos. O vazamento de dados pode comprometer a reputação, o serviço, os recursos e a conformidade com a regulamentação sobre o uso destes dados. No Brasil, o tratamento de dados em todas as instâncias precisa estar de acordo com as diretrizes redigidas pela Lei Geral de Proteção de Dados (LGPD). A Lei Geral de Proteção de Dados dita o que são dados sensíveis e atribui o papel à instituição que os opera de tratá-los de maneira adequada. No tema da saúde, há uma enorme preocupação com as medidas de proteção empregadas, pois as informações são consideradas sensíveis pela LGPD.

O Centro de Computação Científica e Software Livre(C3SL) está desenvolvendo um projeto em parceria com o Ministério da Saúde, que visa a troca de dados entre os profissionais da área da saúde. Portanto, é obrigação do C3SL adotar técnicas que protejam os princípios de segurança do projeto de modo a prevenir o vazamento destas informações sensíveis. Esta monografia detalha a implantação de uma técnica utilizada para este fim, um Sistema de Detecção de Intrusão. Além disso, explicamos a integração desta ferramenta com uma pilha de agregação de registros, que é usada para compactar o grande volume de alertas gerados e visualizar métricas sobre estes alertas. Desta forma, podemos criar filtros sobre o tráfego de rede que está sendo analisado e planejar estratégias de defesa para impedir possíveis ataques que levam a vazamento de dados.

Palavras-chave: Sistema de detecção de intrusão, Detecção por Assinatura, Suricata, Análise de Tráfego, Agregação de logs, Promtail, Loki, Grafana.

Abstract

In light of the growing threat of unauthorized access to sensitive information, a growing number of companies, government agencies, organizations, and institutions at all levels are allocating resources toward the implementation of measures designed to safeguard their data. This is because data has become a significant asset for making strategic decisions, allocating resources, and even guaranteeing sovereignty between governments. The occurrence of data leaks has the potential to negatively impact an organization's reputation, service quality, resource allocation, and compliance with regulations governing the utilization of such data. In Brazil, data processing must adhere to the guidelines set forth by the General Data Protection Law (LGPD), irrespective of the circumstances. The General Data Protection Law delineates the types of data that are considered sensitive and assigns the responsibility to the entity that handles such data to treat it in an appropriate manner. In the context of the health sector, there is a significant degree of concern surrounding the efficacy of the protective measures currently in place, given that the information in question is considered to be of a sensitive nature in accordance with the stipulations set forth by the LGPD.

The Center for Scientific Computing and Free Software (C3SL) is developing a project in collaboration with the Ministry of Health with the objective of facilitating the exchange of data between health professionals. Consequently, it is incumbent upon C3SL to implement measures that safeguard the project's security principles, thus preventing the unauthorized disclosure of this sensitive information. This monograph provides a detailed account of the deployment of an Intrusion Detection System, a technique employed for this purpose. Furthermore, we elucidate the incorporation of this tool into a log aggregation stack, which is employed to compress the copious volume of alerts generated and to visualize metrics pertaining to these alerts. Thus, filters may be created for the network traffic being analyzed, and defensive strategies may be formulated to prevent potential attacks that could result in data leakage. **Keywords:** Intrusion Detection System, Signature based detection systems, Suricata, Network based IDS, Log Aggregation, Promtail, Loki, Grafana.

Lista de Figuras

2.1 Operações comuns em um *firewall* 20

3.1 Roteamento de pacotes pela Roble 26
3.2 Máquinas SUS 29
3.3 Arquitetura do Snort 31
3.4 Arquitetura Multi-Thread do Suricata 32
3.5 A Pilha ELK 39
3.6 A Pilha PLG 40
3.7 Arquitetura 42

4.1 Distribuição dos Logs ao longo das semanas 43
4.2 Distribuição dos Eventos de Log 44
4.3 Categorias de alerta 45
4.4 Principais assinaturas casadas 46

Lista de Tabelas

3.1 Especificações técnicas Roble. 26
3.2 Comparação dos IDS de código aberto. 33
3.3 Comparação entre a Pilha ELK e a Pilha PLG 41

Lista de Acrônimos

AIDS	*Anomaly-based intrusion detection system*
AGTIC	Agência de Tecnologia da Informação e Comunicação
APS	Atenção Primária à Saúde
C3SL	Centro de Computação Científica e Software Livre
CADSUS	Cadastro Único do Sistema de Saúde
CVE	*Common Vulnerabilities and Exposures*
DHCP	*Dynamic Host Configuration Protocol*
DHCPv4	*Dynamic Host Configuration Protocol version 4*
DiD	*Defense in Depth*
DInf	Departamento de Informática
DoS	*Denial of Service*
DoS	*Distributed Denial of Service*
ELK	Elasticsearch, Logstash, Kibana
GCS	*Google Cloud Storage*
HIDS	*Host based IDS*
HTTP	*Hyper-Text Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IDES	*Intrusion Detection Expert System*
IDS	*Intrusion Detection System*
IP	*Internet Protocol*
LGPD	Lei Geral de Proteção de Dados
MitM	Man-in-the-middle attack
MS	Ministério da Saúde
NIDS	*Network Intrusion Detection System*
NIPS	*Network Intrusion Prevention System*

OISF	*Open Information Security Foundation*
OMFWD	*syslog Forwarding Output Module*
PCAP	*Packet CAPture*
PLG	Promtail, Loki, Grafana
PoP	Ponto de Presença
PoP-PR	Ponto de Presença no Paraná
RNP	Rede Nacional de Ensino e Pesquisa
RST	*Reset*
SAPS	Secretaria de Atenção Primária à Saúde
SCPA	Sistema de Cadastro e Permissão de Acesso
SIDS	*Signature-based intrusion detection system*
SQL	*Structured Query Language*
SQLi	*Structured Query Language injection*
SRI	*Stanford Research Institute*
SUS	Sistema Único de Saúde
TCP	*Transmission Control Protocol*
UFPR	Universidade Federal do Paraná
VLAN	*Virtual Local Area Network*

Lista de Símbolos

α	alfa, primeira letra do alfabeto grego
β	beta, segunda letra do alfabeto grego
γ	gama, terceira letra do alfabeto grego
ω	ômega, última letra do alfabeto grego
π	pi
τ	Tempo de resposta do sistema
θ	Ângulo de incidência do raio luminoso

Sumário

1 Introdução **14**
1.1 Justificativa . . . 15
1.2 Objetivos . . . 15

2 Fundamentação Teórica **16**
2.1 Ameaças . . . 16
2.2 Vulnerabilidades . . . 17
2.3 Tipos de Ataques . . . 17
2.3.1 Ataques de negação de serviço e de negação de serviço distribuído . . . 18
2.3.2 *Man-in-the-middle* (MitM) *attack* . . . 18
2.3.3 *Phishing* . . . 18
2.3.4 *Drive-by-download attack* . . . 18
2.3.5 Ataque de senhas . . . 18
2.3.6 Injeção SQL (SQLi) . . . 19
2.4 Defesa em profundidade . . . 19
2.5 *Firewall* . . . 19
2.6 Sistema de Detecção de Intrusão . . . 21
2.6.1 Classes de IDS . . . 22
2.6.2 Tipos de detecção . . . 22
2.6.3 Sistemas de Prevenção de Intrusão . . . 23
2.7 Revisão da Literatura . . . 23

3 Prova-de-Conceito **25**
3.1 Roteamento dos pacotes . . . 25
3.2 Espelhamento do tráfego de rede . . . 26
3.2.1 iptables: Tabelas, Cadeias e Regras . . . 27
3.2.2 Regra de espelhamento para o IDS . . . 27
3.3 Arquitetura do projeto SUS . . . 28
3.4 Software Livre . . . 29
3.5 Justificando a escolha da ferramenta de IDS . . . 30
3.5.1 Snort . . . 30
3.5.2 Suricata . . . 31
3.5.3 Snort VS Suricata . . . 33
3.6 Operando o Suricata . . . 33
3.6.1 Anatomia das Regras . . . 33
3.6.2 Configurações e Instalação . . . 35
3.7 Rsyslog . . . 37
3.8 Justificando a escolha da pilha de agregação e visualização de logs . . . 38

3.8.1 A Pilha ELK 38
3.8.2 A Pilha PLG 39
3.8.3 A Pilha ELK VS A Pilha PLG 40
3.8.4 Topologia do Trabalho 41

4 Resultados **43**
4.1 Registros do tipo Alerta 44
4.1.1 Filtrando por categoria de ataque 45
4.1.2 Filtrando por assinatura 46
4.2 Discussão 47

Referências Bibliográficas **49**

Capítulo 1

Introdução

Os dados ganham cada vez mais importância para empresas, instituições, organizações e órgãos em todas as esferas do governo, isto porque a análise de dados pode proporcionar um melhor entendimento da realidade. Desta forma, recursos podem ser alocados de maneira mais eficiente e decisões estratégicas podem ser tomadas. Um exemplo foi a abertura da Semana da Governança de Dados em 2024. No evento, a Ministra da Gestão e Inovação em Serviços Públicos, Esther Dweck, afirmou que um dos objetivos é unificar o enorme volume de dados produzidos sobre a realidade brasileira em um ecossistema seguro e robusto na intenção de garantir uma autonomia tecnológica nacional ((BRASIL, 2024)).

A medida com que os dados foram ganhando relevância, a preocupação com a segurança destes dados foi crescendo na mesma proporção. Surge então a necessidade da elaboração de uma regra jurídica que contemplasse as garantias e direitos sobre a liberdade e privacidade dos dados — a Lei Geral de Proteção de Dados (LGPD). A LGPD define o que são dados pessoais e quais cuidados devem ser tomados na hora de operar sob eles em todas as instâncias ((Ribacionka et al., 2022)).

Na área da saúde, o tema é de grande relevância pois as informações sobre a saúde de um paciente possuem forte relação com a sua intimidade. As informações podem ser sobre as condições do paciente, sobre etnia e convicção religiosa. Essas informações são consideradas como dados sensíveis pela LGPD e devem ser tratadas de acordo pela instituição responsável (Ribacionka et al., 2022). Apesar dos esforços para cumprir as diretrizes estabelecidas pela LGPD, não são incomuns as ocorrências de vazamentos de informações de saúde. De acordo com o Ministério da Saúde (MS), de 2019 à 2022 foram relatados pelo menos três grandes incidentes com dados pessoais envolvendo o Sistema de Cadastro e Permissão de Acesso (SCPA) e o Cadastro Único do Sistema de Saúde (CADSUS) (BRASIL, 2024).

A falha na prevenção das invasões, que podem levar a vazamentos de dados, não apenas fere as diretrizes previstas pela LGPD mas também levam a perca de credibilidade dos serviços, danos à reputação, exposição dos usuários, roubo de propriedade intelectual e dano à recursos físicos e monetários ((Ribacionka et al., 2022)). Dada esta relevância, esta monografia apresenta um estudo de caso sobre a implantação de uma técnica de prevenção de intrusões em rede em um ambiente de desenvolvimento de um projeto de saúde em parceria com o MS.

A técnica implantada é um Sistema de Detecção de Intrusão (IDS). Os sistemas de detecção de intrusão são programas de software ou hardware que identificam atividades maliciosas em sistemas de computação com a intenção de manter sua segurança. Um IDS pode monitorar o tráfego de rede para impedir que ataques comprometam os princípios de segurança de um sistema de informação e consequentemente a privacidade dos dados. ((Khraisat et al., 2019)).

1.1 Justificativa

O Centro de Computação Científica e Software Livre (C3SL), grupo vínculado ao Departamento de Informática (DInf) da Universidade Federal do Paraná, iniciou um projeto em parceria com o MS em 2024. O objetivo é compartilhar informações entre gestores e profissionais da saúde para divulgar os serviços do Sistema Único de Saúde (SUS) e coletar dados para avaliar a Atenção Primária à Saúde (APS).

O C3SL administra todo o parque computacional do DInf: as máquinas físicas e as máquinas virtuais; o sistema de e-mail; as contas para alunos da graduação, alunos pós graduação e professores acessarem os laboratórios; a rede física, cabeada e não cabeada (Wifi), e rede lógica. Dada a relevância do projeto, é responsabilidade do C3SL implementar medidas de segurança para proteger os dados de saúde de acordo com as diretrizes da LGPD.

Nesse contexto, o tráfego de rede foi segmentado logicamente para atender apenas este projeto através de uma VLAN. Desta forma, os dados do projeto são transmitidos apenas neste domínio e não se misturam com as demais áreas do departamento, sejam de administração, ensino ou pesquisa. O tráfego desta VLAN é redirecionado para um IDS que analisa e classifica os pacotes, gerando alertas quando necessário e permitindo a visualização deles.

Desta maneira, a implementação de um IDS poderá nos permitir descobrir o padrão de tráfego que chega à rede segmentada do projeto e auxilia o analisador a tomar medidas para mitigar os possíveis ataques no futuro.

1.2 Objetivos

O objetivo desta monografia é apresentar um estudo de caso sobre a implementação de um sistema de detecção de intrusões em um ambiente real. O ambiente da implantação será em uma instituição pública de ensino superior, em uma rede local virtual isolada para o desenvolvimento de um projeto da saúde em parceria com o Ministério da Saúde. Além da análise e classificação do tráfego que chega nessa rede, queremos atingir os seguintes objetivos com a implantação do sistema:

- Explicar conceitos base de segurança da computação.
- Caracterizar e categorizar sistemas de detecção de intrusão, justificar a escolha de nossa ferramenta, diferenciar técnicas de detecção, funcionamento das regras.
- Propor uma forma de agregação dos alertas de maneira robusta.
- Mostrar uma ferramenta de visualização dos alertas gerados para analista.
- Contribuir com outros analistas e administradores que eventualmente precisem realizar implantações semelhantes.

Capítulo 2

Fundamentação Teórica

A segurança é o ato ou efeito de segurar. É também sinônimo de confiança, garantia, firmeza, estabilidade e certeza (Significados, 2024). A partir dessa definição, podemos entender a "segurança computacional" como o ato de garantir ou segurar um conjunto de primitivas diretamente associadas a um sistema de computação. Em outras palavras, o termo "segurança computacional" faz referência à garantia de um conjunto de propriedades essenciais relacionadas às informações de um sistema e a todo o arcabouço tecnológico utilizado para a existência e manutenção delas. De acordo com Maziero (2019), podemos caracterizar essas propriedades como:

- **Confidencialidade:** os dados e recursos presentes no sistema devem ser consultados única e exclusivamente por pessoas devidamente autorizadas;
- **Integridade:** os ativos do sistema só podem ser modificados ou destruídos por pessoas devidamente credenciadas, que tenham o direito de executar tais ações;
- **Disponibilidade:** os recursos devem estar livres para serem acessados em qualquer momento por usuários que possuam este direito.
- **Autenticidade:** as informações relacionadas a uma entidade são legítimas, assim como as próprias entidades.
- **Não-repúdio:** as ações realizadas sob as informações ou recursos não podem ser omitidas ou declinadas.

Essas propriedades podem estar sujeitas a violações advindas de erros humanos ou indivíduos mal-intencionados. Ou seja, em diversos contextos, estas propriedades podem estar sob ameaça.

2.1 Ameaças

Para assegurar estas propriedades, a segurança precisa lidar com qualquer ação que as coloque em risco, em outras palavras: precisa mitigar ameaças. As ameaças podem ser intencionais: ataque, invasão ou qualquer acesso/entrada de uma pessoa que não possua as credenciais necessárias; e podem ser não intencionais: acidentes operacionais, falhas de hardware ou qualquer indisponibilidade de acesso aos recursos que não tenha sido gerada propositalmente. Para evitar ameaças, intencionais ou não, os desenvolvedores de software seguem padrões de qualidade de processo e de qualidade de software, a fim de evitar erros no código que levam a vulnerabilidades (Amoroso, 1994).

Ameaças podem surgir a partir de técnicas de extração de informações privilegiadas,como a engenharia social, ou através de brechas em um sistema. No primeiro caso, os criminosos analisam o comportamento humano e a falta de conhecimento de usuários para coletar dados e conseguir acesso não autorizado. Por exemplo, o atacante pode solicitar informações sigilosas por e-mail, alegando ser alguém legítimo, ou pode se passar por técnico de infraestrutura em uma empresa para ganhar acesso físico a uma sala ou conjunto de máquinas (Kaspersky, 2024a).

No segundo caso, erros inerentes do processo de desenvolvimento de software podem deixar o sistema vulnerável. Mesmo quando há padrões de engenharia de software e qualidade de processo aplicados ao desenvolvimento, a tendência é que erros sejam introduzidos no software ao longo de seu ciclo de vida (Pressman, 2014). Portanto, novas vulnerabilidades são reportadas todos os dias, tornando a tarefa de mitigar ameaças um esforço constante.

2.2 Vulnerabilidades

Uma vulnerabilidade em um sistema computacional é um estado de fraqueza que permite que uma potencial ameaça ocorra com efetividade. Desta forma, a existência deste estado frágil compromete os princípios que constituem a segurança de um sistema computacional (Amoroso, 1994). Administradores de sistema costumam implantar varredores automatizados em suas máquinas para que possam identificar e corrigir as vulnerabilidades encontradas. Este processo é chamado de gestão das vulnerabilidades.

Estes varredores consultam vulnerabilidades previamente catalogadas, as *Common Vulnerabilities and Exposures* (CVE). Os seres humanos tendem a cometer os mesmos erros e a reutilizar soluções para problemas conhecidos. Ora, seria inviável desenvolver uma aplicação se fosse necessário criar uma nova linguagem de programação para cada projeto ou desenvolver do zero um servidor web para hospedar a aplicação. Portanto, os mesmos erros costumam ocorrer em diversos ambientes computacionais e são replicados por instituições através do reúso de ferramentas, de códigos e programas de software. Por isso, são catalogadas as brechas mais conhecidas.

As CVEs têm como objetivo identificar, definir e catalogar as vulnerabilidades de segurança computacional de forma pública. Cada vulnerabilidade ganha um registro CVE em um catálogo. Após identificadas, elas são atribuídas à organização que as descobriu e são publicadas em parceria com o programa CVE. Portanto, organizações ao redor do mundo podem coordenar seus esforços para corrigir vulnerabilidades de maior prioridade e proporcionar melhor qualidade da proteção de dados para todos (CVE, 2024).

A limitação que existe nesta técnica são as vulnerabilidades não cadastradas. Além disso, o processo de identificar as vulnerabilidades pode ser automatizado com scripts de varreduras periódicas, mas o processo de corrigir vulnerabilidades é uma tarefa exaustiva e de difícil automação.

2.3 Tipos de Ataques

Apesar do esforço contínuo para identificar e corrigir as vulnerabilidades, nem todas são eliminadas antes de serem exploradas por atacantes. Um *ataque* é uma ação no ecossistema computacional que explora uma vulnerabilidade e leva uma ameaça em potencial a ser concretizada (Amoroso, 1994). Os principais alvos dos atacantes de sistemas de informação são dados e informações de sites do governo, instituições financeiras, grandes mídias e redes militares.

As motivações por trás dos ataques são a obstrução de informações, retardamento do processo de tomada de decisões, negação de prestação de serviços públicos, diminuição da reputação de instituições e obtenção de recursos financeiros (Uma e Padmavathi, 2013). A subseção a seguir pauta alguns possíveis ataques que exploram vulnerabilidades não corrigidas:

2.3.1 Ataques de negação de serviço e de negação de serviço distribuído

Um ataque de negação de serviço (DoS) deixa os recursos indisponíveis para os usuários alvo através de uma sobrecarga nos recursos do sistema. Desta forma, os recursos ficam alocados para "atender"o atacante e seus serviços primários não são executados. O primeiro host afetado espalha o software malicioso que o infectou pela rede, expandindo o ataque de negação de serviço distribuído (DDoS) (Biju et al., 2019).

TCP SYN Flood attack, *teardrop attack*, *smurf attack*, *ping-of-death attack* e *botnets* são variações do ataque DoS e DDoS. Estes ataques são efetivos pois é difícil diferenciar uma solicitação de tráfego legítima de uma solicitação maliciosa (Biju et al., 2019) e muitas instituições não possuem mecanismos de defesa contra esse ataque ou não possuem uma infraestrutura de rede que consiga competir com a dos atacantes.

2.3.2 *Man-in-the-middle* (MitM) *attack*

Em um ataque MitM, o atacante age como intermediário entre o cliente e o servidor para interceptar e manipular as informações entre ele. Após interceptar mensagens em tempo real enviadas pela rede, ele personifica uma das partes envolvidas. Variações do ataque MitM são o sequestro de sessão, *IP spoofing and reply* (Biju et al., 2019).

2.3.3 *Phishing*

Envio de e-mails fraudulentos que tentam parecer de fontes legítimas e confiáveis. Esse é um ataque de engenharia social que explora o erro humano para conseguir informações que levem a acesso não autorizado. Os e-mails são forjados para parecerem autênticos, mas possuem algum hyperlink com um malware incorporado, site que leva a fazer o download de um malware ou formulário cujo destino das informações é uma base de dados do atacante. Diferentes formas em que desse ataque são *whaling*, *spear phishing*, *pharming* e *deceptive*.

2.3.4 *Drive-by-download attack*

Esse ataque ocorre quando um usuário visita um site alterado por atacantes para conter objetos maliciosos dentro das páginas web. A intenção é instalar programas maliciosos sem o consentimento do usuário. Esses ataques injetam código malicioso em sites legítimos, que acabam sendo usados para espalhar essa ameaça. Ao visitar a página, código malicioso é carregado no navegador web do usuário. Geralmente, as variações de código JavaScript, os redirecionamentos e as interações maliciosas são imperceptíveis. O objetivo é espalhar o malware para a maior quantidade de usuários e conseguir acesso não autorizado às suas informações.

2.3.5 Ataque de senhas

Esse ataque consiste em obter a senha do usuário para personificá-lo. Esse ataque é um ataque de engenharia social que pode ser combinado com recursos computacionais à disposição

do atacante. Você pode obter a senha de maneira direta, por exemplo, ao observar o usuário digitar a senha ou roubar um caderno físico de senhas do usuário. Ou pode tentar deduzir a senha a partir de informações como datas de aniversário, nome dos filhos e animais de estimação.

Ainda, há como observar pacotes da rede que não passem por um processo de criptografia ou utilizar dicionários de senha especializados para tentar quebrar a senha por força bruta (Biju et al., 2019). Na prática, o método de força bruta é o mais eficaz, ainda mais quando está aliado à engenharia social, pois pode reduzir o escopo do dicionário de senhas. Por exemplo, saber a localização do usuário permite usar um dicionário regional.

2.3.6 Injeção SQL (SQLi)

Linguagem de consulta estruturada (SQL) é uma linguagem desenvolvida para gerenciar dados relacionais em um sistema de gerenciamento de banco de dados, ou para processamento de fluxo de dados. Nesse ataque, o invasor consegue uma brecha para executar consultas SQL não validadas no backend de uma aplicação. São feitas consultas estratégicas com a intenção de manipular o banco para obter acesso a informações. Isso pode levar à visualização, exposição ou deleção de informações sensíveis (Biju et al., 2019).

Em casos mais comprometedores, em que o banco de dados possui privilégios administrativos, o invasor consegue acesso não somente aos dados da aplicação mas também ao sistema de informações inteiro. Algumas variações deste ataque são SQLi em banda, SQLi com base em erro, SQLi com base em Union, SQLi de inferência, SQLi com base em tempo, SQLi booleano, SQLi fora de banda (Kaspersky, 2024b).

2.4 Defesa em profundidade

Defesa em profundidade (DiD) é um termo de origem militar em que uma das linhas de defesa se sacrificava para deter forças adversárias. Em segurança computacional, este termo aparece com frequência mas é empregado de maneira completamente diferente, no sentido de que várias camadas de segurança cooperam para preservar as propriedades de segurança de um sistema. Isso ocorre porque não há uma técnica ou ferramenta que consiga barrar todos os tipos de ataque em um sistema (Cloudflare, 2024).

Os ataques DoS e DDoS são combatidos implantando um IDS e proporcionando uma infraestrutura com grande largura de banda. Ataques MitM também são combatidos com o emprego de um IDS, mas também podem ser evitados com redes virtuais privadas. Porém, ataques de engenharia social como ataques de senha e phishing são combatidos ao fornecer treinamentos de segurança para equipes e aplicando políticas rígidas de senhas. Ataques SQLi são combatidos através da implantação de um *firewall*, da implantação de um IDS ou através do gerenciamento de vulnerabilidades Biju et al. (2019).

Através destes exemplos, é possível observar que cada tipo de ataque possui uma abordagem diferente para ser evitado. Portanto, para proporcionar um sistema que cubra a maior quantidade de ameaças, é necessária a implementação de várias ferramentas e práticas de segurança que podem auxiliar na detecção, prevenção e mitigação de ataques.

2.5 *Firewall*

Apenas a gestão das vulnerabilidades não é suficiente para prevenir diferentes categorias de ataques. Além da gestão das vulnerabilidades, outra técnica de mitigação de ataques é o uso

de *firewall*. O *firewall* é a primeira linha de defesa da rede. Ele é um mecanismo de segurança intermediário entre a rede interna e externa. De acordo com (Cisco, 2024a), o *firewall* pode se basear em um conjunto de regras pré-definidas para executar as seguintes operações:

1. Permitir (*Allow*): O *firewall* permite que o tráfego passe livremente. Os pacotes são encaminhados até o seu destino final.
2. Negar (*Deny*): O *firewall* apenas bloqueia o tráfego. Os pacotes são descartados sem nenhuma sinalização para o remetente e o destino final não é alcançado.
3. Rejeitar (*Reject*): O *firewall* bloqueia o tráfego e notifica o remetente que o pacote foi descartado. O destino final não é alcançado.
4. Desviar (*Redirect*): O *firewall* redireciona o tráfego para outra porta ou endereço final.
5. Espelhar (*Mirror*): Ele permite que o tráfego passe livremente mas o copia e envia para um segundo destino, geralmente para análise ou monitoramento. O destino final é alcançado.

   A Figura 2.1 ilustra algumas operações feitas por um *firewall*.

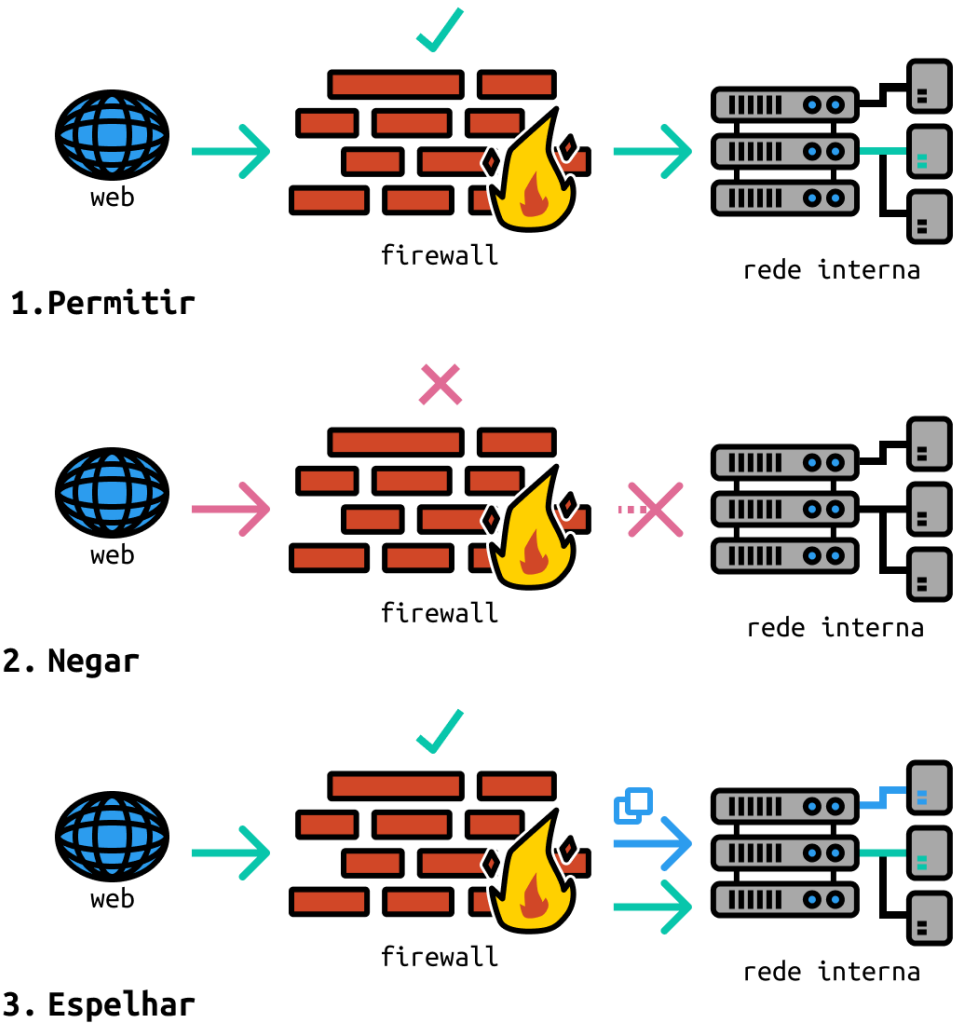


Figura 2.1: Operações comuns em um *firewall*

O *firewall* é o mecanismo mais primitivo de filtragem de pacotes de rede. Consequentemente, ele possui algumas limitações. De acordo com (Sulaman, 2011), algumas de suas limitações são:

- Ele não consegue impedir a espionagem ou tentativas de intrusão a partir da rede interna.
- Ele é limitado pela robustez de suas regras de configuração, pois ele não sabe o que é e o que não é permitido.
- O próprio *firewall* pode estar sujeito a ataques de negação de serviço (DoS), através de algumas técnicas de fragmentação .

2.6 Sistema de Detecção de Intrusão

Uma outra técnica para impedir que vulnerabilidades permitam com que uma ameaça potencial ocorra são os Sistemas de Detecção de Intrusão (IDS). Uma intrusão pode ser definida como qualquer acesso não autorizado que cause dano ao sistema de informação. Em outras palavras, qualquer tentativa de invasão que possa comprometer as primitivas ou propriedades diretamente associadas ao sistema de informação (Khraisat et al., 2019). Um Sistema de Detecção de Intrusão é um programa de software ou hardware que identifica ocorrências que indiquem mau uso ou violação das regras da rede ou do sistema, na intenção de preservar as propriedades essenciais de segurança.

O objetivo de um IDS é identificar diferentes padrões de intrusão que não poderiam ser identificados por um *firewall* tradicional (Khraisat et al., 2019). Diferente do *firewall*, um IDS não é necessariamente implantado como um intermediário entre a rede interna e a rede externa, podendo ser posicionado de maneira estratégica fora do tráfego principal. A maneira mais comum de operar um IDS é através do espelhamento de pacotes.

Um IDS coleta e armazena dados, analisa padrões de comportamento, horários e outros aspectos que considerar relevantes. Ele utiliza as informações coletadas em conjunto com um conhecimento prévio de padrões de ataque, e identifica se o evento é legítimo. A coleta pode ser feita em várias pontas, como mecanismos de entrada e saída, sistemas de arquivos, tabelas de regras ou análise do fluxo de rede. Então, um IDS executa uma série de algoritmos em busca de evidências que possam tirar a legitimidade de um evento, considerando-o malicioso por consequência (Rebello et al., 2024).

O termo IDS surgiu no começo da década de 80 com o trabalho "*Computer Security Threat Monitoring and Surveillance*", de de Anderson (1980). O trabalho dele introduziu o conceito de "detecção de uso indevido" e eventos específicos de usuários. Além de criar a noção de IDS em geral, seu trabalho contribuiu principalmente para o desenvolvimento de IDS baseados em host, ou *host-based IDS* (Ashoor e Gore, 2011).

Alguns anos após o trabalho de J. Anderson, a SRI International, em parceria com o Dr. Dorothy Denning, desenvolveu o primeiro IDS funcional, que ficou conhecido como *Intrusion Detection Expert System* (IDES). Os trabalhos publicados por Dr. Denning foram base para a maioria dos IDS que surgiram posteriormente (Ashoor e Gore, 2011). Após o aumento da popularidade dos IDS e cada vez mais novos trabalhos surgindo na área, começou-se a categorizar os IDS com base em certos critérios. Iremos explorar essa categorização na próxima subseção.

2.6.1 Classes de IDS

Os IDS podem ser classificados ou categorizados com base em dois principais critérios: a técnica utilizada para a detecção das intrusões e a forma como as entradas de dados são fornecidas (Khraisat et al., 2019). Ao classificar por fontes de entrada dados, temos três tipos principais de classes IDS: *Host based IDS* (HIDS), *Network based IDS* (NIDS) e *Hybrid based IDS* (Ashoor e Gore, 2011).

O HIDS é uma classe que inspeciona dados originados a partir de um dispositivo, como um servidor ou uma estação de trabalho. Os dados são analisados localmente na máquina e podem vir diferentes fontes da mesma máquina, como arquivos acessados, processos abertos ou registros de logs. Por ter essa natureza, o HIDS pode detectar ataques originados da rede interna que não envolvem tráfego de rede. O HIDS pode usar tanto o sistema de detecção de anomalias quanto o de uso indevido (Ashoor e Gore, 2011).

O NIDS é uma classe implantada em um ponto específico da infraestrutura de rede. Ele pode capturar e analisar o tráfego de rede, ou apenas analisar um tráfego previamente capturado na forma de Sflow, Netflow, pcap ou outros formatos conhecidos (Khraisat et al., 2019). Os NIDS são conhecidos também como "*packet-sniffers*" (Rebello et al., 2024). Um NIDS pode monitorar uma atividade maliciosa numa fase ainda inicial, impedindo que ela se espalhe para outro computador. Sua limitação é a dificuldade de analisar todos os dados quando se tem uma alta largura de banda de dados (Khraisat et al., 2019).

E por fim, o *Hybrid based IDS* é uma classe que mescla aspectos das classes anteriores. É um arquétipo possível em que, embora não tenha uma arquitetura padrão, a ideia é fazer o gerenciamento dos alertas de dispositivos HIDS e NIDS em orquestrador central de detecção de intrusão (Ashoor e Gore, 2011). Algumas abordagens podem combinar o uso de HIDS, NIDS e *firewall*, com a ideia de fazer várias camadas de proteção contra ameaças.

2.6.2 Tipos de detecção

A subseção anterior categorizou os IDS pela tipo de fonte de dados. Uma outra abordagem é a classificação por técnica de detecção de ocorrências. Podemos classificar, novamente, em três grupos principais: Sistemas de Detecção Baseada em Assinatura (SIDS) e os Sistemas de Detecção de Intrusão Baseados em Anomalia (AIDS) (Ashoor e Gore, 2011).

Os SIDS são baseados em técnicas de casamento de padrões para detectar ataques conhecidos. Quando uma assinatura de intrusão coincide com uma assinatura de intrusão anterior que já existe no banco de dados de assinaturas, um alerta é gerado. Em um HIDS, o casamento por assinaturas poderia olhar logs do host para encontrar sequências de comandos ou ações que caracterizem comportamento malicioso. A ideia geral do SIDS é construir um banco de assinaturas de intrusão, comparar com o conjunto atual de atividades com as assinaturas existentes e gerar alertas (Khraisat et al., 2019). E é justamente na natureza do SIDS que surge sua limitação: ele não detecta intrusões não catalogadas em sua base de dados. Em outras palavras, ele não detecta novos ataques.

O AIDS surge para tentar superar as limitações do SIDS. A ideia gira em torno de monitorar ocorrências que saiam do padrão de uso, das regras convencionais do sistema ou da rede. O padrão de uso da rede pode ser determinados por modelos estatísticos, machine learning ou conhecimento prévio da rede. Qualquer ocorrência significante pode ser apontada como anomalia e classificada como comportamento não legítimo (Khraisat et al., 2019).

O AIDS vai argumentar que é justamente por ser um evento que destoa do comportamento normativo da rede que podemos assumir que é uma tentativa de intrusão não catalogada. E é nessa característica intrínseca do AIDS que notamos a limitação dessa técnica: há uma dependência de

dados estatísticos sobre o uso da rede ou um conhecimento prévio do comportamento padrão da rede, além do tempo e recursos para treinar os modelos de aprendizado de máquina (Khraisat et al., 2019).

Ainda, há construções híbridas que mesclam os tipos de detecção. Isso acontece pois os IDS tradicionais possuem limitações. Eles não podem ser facilmente modificados, não detectam novos ataques maliciosos, possuem precisão relativamente baixa e geram um grande volume de alarmes falsos. Portanto, algumas implementações geram combinações de detecção baseada em assinatura e detecções baseadas em anomalias, conseguindo uma melhor precisão de detecção de ocorrências maliciosas (Khraisat et al., 2019).

2.6.3 Sistemas de Prevenção de Intrusão

Os sistemas de prevenção de Intrusão (IPS) interceptam e analisam o tráfego malicioso. Eles operam preventivamente para mitigar intrusões antes que um pacote malicioso entre e se propague na rede interna. Isso reduz a carga de trabalho dos analistas e da equipe de segurança. Os IPS atuam de maneira mais ativa, agindo como um intermediário entre o tráfego interno e externo, interrompendo tentativas de exploração das vulnerabilidades de um sistema. Diferente de um IDS, um IPS age sempre bloqueando as ameaças em tempo real. Um IDS pode fazer a análise de um tráfego previamente coletado. Além disso, um IDS gera um alerta sobre as ocorrências de eventos ilegítimos, enquanto o IPS age imediatamente para barrar o ataque. É comum que eles sejam implantados após o *firewall* (pal, 2024).

2.7 Revisão da Literatura

(Khraisat et al., 2019) faz uma revisão da literatura, compilando pesquisas sobre IDS. Eles apresentam um histórico sobre a evolução dos malwares e intrusões. Após isso, eles descrevem o que é um sistema de detecção de intrusão e qual o seu objetivo. Eles seguem fazendo definições e taxonomias, classificando os IDS em dois grandes grupos: sistemas baseados em assinaturas (SIDS) e sistemas baseados em anomalias (AIDS). Khraisat et al., fazem uma revisão dos datasets mais comuns, como o NSL-KDD e DARPA, apontando suas limitações, como a falta de representatividade para cenários reais. Ainda, falam da aplicação de técnicas de aprendizado de máquina para superar as limitações dos métodos tradicionais e como a hibridização dos AIDS e SIDS podem contribuir para a redução de falsos positivos, melhorando a precisão dos IDS. Por fim, falam da dificuldade de lidar com técnicas de evasão por partes dos atacantes.

(Sulaman, 2011) Faz uma comparação entre *Firewall* de IDS. Ele começa caracterizando as funções de um Firewall, suas classificações e tipos. Ele também carateriza e classifica um IDS. Após falar sobres suas principais funcionalidades, objetivos e classificações, ele fala sobre as limitações de um *Firewall* e as limitações de um IDS, e como uma boa estratégia pode ser utilizar os dois (Defesa em Profundidade) para garantir maior segurança.

(Vaz et al., 2021) faz um estudo de caso sobre a implantação de um ambiente de prevenção de intrusões com a ferramenta Suricata. Os autores avaliam a eficácia de um IPS na detecção de bloqueios e ameaças em uma rede de computadores de uma instituição educacional. A arquitetura é feita com a combinação de um NIPS e um HIPS, utilizando Suricata em modo inline. Para a visualização dos registros, eles fazem a integração do Suricata com a pilha ELK (Elasticsearch, Logstash, Kibana). Essa pilha, é responsável por armazenar, compactar e proporcionar um ambiente capaz de criar e interagir com painéis. Após uma semana coletando alertas, eles foram afunilando as regras para aumentar a eficácia do sistema.

(Eriksson e Karavek, 2023) Compara duas soluções de gerenciamento de logs amplamente utilizadas, a pilha ELK ELK (Elasticsearch, Logstash, Kibana) e a pilha PLG (Promtail, Loki, Grafana). Os autores apresentam uma visão inicial das plataformas e seus compomentes, detalhando o que cada parte faz em relação aos registros. Seguem falando sobre vantagens e desvantagens de cada uma delas, suas limitações e adequações de uso. A metodologia foi feita através de experimentos em containers avaliando uso dos recursos (CPU, memória, armazenamento) e comparando suas funcionalidades (interfaces gráficas).

Capítulo 3

Prova-de-Conceito

No capítulo anterior, apresentamos a fundamentação que nos permitirá ter um melhor entendimento do trabalho proposto. Neste capítulo, descrevemos a prova-de-conceito de um sistema de detecção de intrusão implantado na rede do C3SL no contexto do projeto em parceria com o SUS. Vamos detalhar:

- Como o tráfego chega da rede externa até o IDS;
- O arcabouço computacional do DInf empregado no projeto SUS;
- A segmentação lógica da rede em que o projeto reside;
- Como os alertas são armazenados e visualizados.

3.1 Roteamento dos pacotes

As universidades brasileiras se conectam à internet através do Sistema RNP. A rede RNP está conectada às redes acadêmicas internacionais na América Latina, América do Norte, África, Europa, Ásia e Oceania. Atualmente, a RNP possui Pontos de Presença (PoP) nos 26 Estados, conectando universidades públicas, laboratórios de pesquisa e instituições educacionais (RNP, 2024). São estes pontos que permitem que os laboratórios de pesquisa e órgãos da universidade conversem com a internet.

O C3SL é um grupo de pesquisa vinculado ao DInf, localizado na UFPR. Portanto, para acessar o mundo, o C3SL se conecta ao sistema RNP através do PoP-PR. Os pacotes encaminhados ao DInf pelo Pop-PR chegam ao roteador de borda do C3SL, a Roble.

A Roble é uma máquina física que atua como *firewall* e roteador de toda a rede. A Roble também atua como servidor de anúncio de roteador IPv6, necessário para autoconfiguração do IPv6. E por fim, faz a retransmissão de DHCP simples para permitir a comunicação entre as sub-redes e o servidor DHCPv4. O protocolo DHCP é baseado no protocolo Bootstrap (BOOTP), que provê uma estrutura para transmitir informações de configuração aos hosts de uma rede TCP/IP (Cisco, 2024). A Tabela 3.1 lista as especificações técnicas da Roble.

Roble	
CPU	2x Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz
RAM	62 GiB
Threads lógicas	2 sockets x 2 SMT x 10 cores
Sistema operacional	Debian GNU/Linux Unstable 6.5.11 (2023-11-19)
Gerenciador de serviços	System-V-like Init
Sistema de arquivos	ext4 MD RAID1 111.8 GiB

Tabela 3.1: Especificações técnicas Roble.

Em suma, a Roble é a responsável por tratar o tráfego entre o PoP-PR e o C3SL, atuando como *firewall* e prestando serviços que auxiliam no funcionamento da rede. Uma vez que os pacotes chegam à Roble, ela decide se os pacotes serão encaminhados para Departamento de Física, para a Agência de Tecnologia da Informação e Comunicação (AGTIC), para a Sagres (máquina dos espelhos de software livre) ou para o Malbec (switch principal do DInf). A Figura 3.1 ilustra a posição da Roble no contexto de roteador da rede do C3SL:

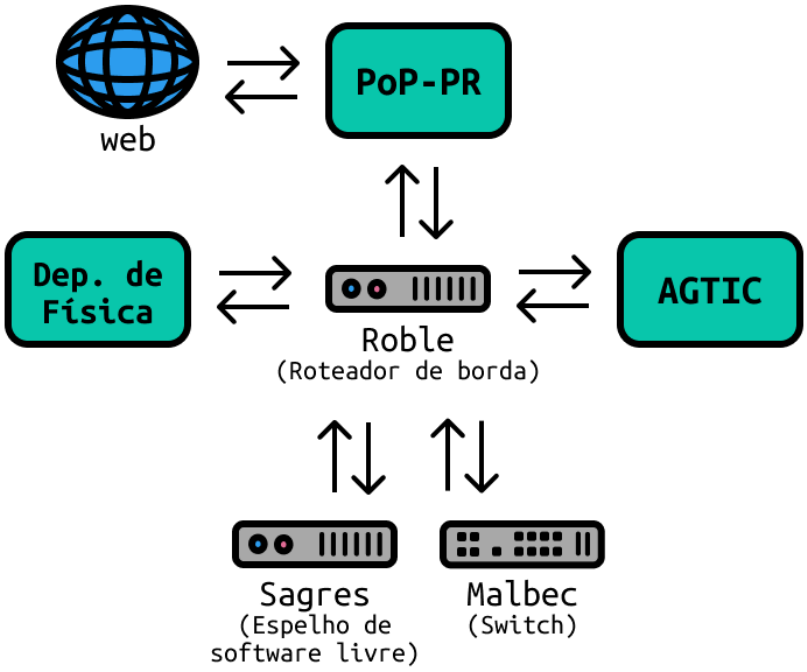


Figura 3.1: Roteamento de pacotes pela Roble

3.2 Espelhamento do tráfego de rede

Como visto anteriormente, uma das primitivas de um *firewall* é o espelhamento de pacotes. No espelhamento, o *firewall* permite que o pacote chegue ao seu destino final mas cria

uma cópia que é enviada para um outro dispositivo. A Roble faz com que os pacotes destinados a VLAN SUS sejam espelhados para o IDS. O *firewall* da Roble é implementado a nível de kernel do Linux, através do iptables.

O iptables é um utilitário de linha de comando para configurar o *firewall* do kernel do Linux. Já é comum que o termo iptables seja utilizado como metonímia, pois, usuários comuns usam o termo iptables para se referir ao *firewall* no nível do kernel do Linux. O iptables é usado para IPv4 e o ip6tables é usado para IPv6, mas ambos têm a mesma sintaxe, apenas algumas opções são específicas para cada versão (Wiki Archlinux, 2024a).

O iptables se organiza em uma coleção de tabelas. As tabelas são compostas por um conjunto de cadeias predefinidas, e as cadeias contêm regras que são percorridas em ordem. Cada regra consiste em um predicado de possíveis correspondências e uma ação que é executada caso a correspondência seja verdadeira. Se o pacote IP chegar ao final de uma cadeia incorporada, incluindo uma cadeia vazia, o destino da política da cadeia determinará o destino final do pacote (Wiki Archlinux, 2024a).

3.2.1 iptables: Tabelas, Cadeias e Regras

O iptables contém cinco tabelas:

1. raw: usada somente para configurar pacotes de modo que eles fiquem isentos do rastramento de conexão;
2. filter: tabela padrão, em que a maioria das ações associadas a um *firewall* ocorrem;
3. nat: usada para tradução de endereços de rede;
4. mangle: usada para fazer alterações especializadas em pacotes;
5. secutiry: usada para controle de acesso;

Majoritariamente, as tabelas utilizadas serão a filter e a nat. As outras configurações são mais complexas e envolvem vários roteadores e decisões de roteamento.

A tabela padrão, filter, contém três cadeias associadas: INPUT, OUTPUT e FORWARD, que são acionadas em pontos diferentes no processo de filtragem de pacotes. Por padrão, as cadeias são vazias, e regras surgem apenas conforme os administradores adicionam. A política padrão de uma cadeia é o ACCEPT, e é aplicada sempre que o pacote chega ao final de uma cadeia (Wiki Archlinux, 2024a).

São as regras que determinam os destinos dos pacotes. As regras exigem que um conjunto de condições sejam satisfeitas para que as ações possam ser aplicadas nos pacotes. As condições podem ser interfaces por onde o pacote chegou, portas destino, protocolo do pacote (Wiki Archlinux, 2024a). Os alvos são especificados com o parâmetro -j ou –jump. Possíveis alvos padrão são: ACCEPT, DROP, QUEUE e RETURN. Mas há uma gama de módulos de extensão como: REJECT, TEE, TOS, e outros (Wiki Archlinux, 2024a).

3.2.2 Regra de espelhamento para o IDS

Para gerenciar o *firewall* através do iptables, o C3SL implementou um script em bash que invoca várias regras do iptables em uma ordem que faça sentido para o funcionamento da rede. O fluxo do script do *firewall* é:

1. Limpar todas as regras escritas no iptables;
2. Fechar todas as portas de entrada e saída;
3. Adicionar novas regras escritas ao longo do script;

Levando em conta a ordem das regras, as seguintes invocações do iptables foram feitas no script para fazer o espelhamento do tráfego para o IDS:

```
$ipt -A FORWARD -p tcp -i POP -o SUS -j TEE --gateway ids.sus.
   c3sl.ufpr.br;
$ipt -A FORWARD -p tcp -i CCE -o SUS -j TEE --gateway ids.sus.
   c3sl.ufpr.br;
```

Quebrando os componentes, temos que:

- `$ipt` invoca o iptables;
- `-A FORWARD`, Anexa nova regra ao final da cadeia "FORWARD"da tabela filter.
- `-p tcp`, aplica a regra caso o pacote pertence ao protocolo TCP.
- `-i POP` ou `-i CCE`, são as interfaces de entrada pelas quais os pacotes estão chegando ao sistema.
- `-o SUS`, é a interface de saída, do SUS.
- `-j TEE`, o alvo TEE duplica o pacote e o direciona para outra máquina. O fluxo original não é interrompido.
- `-gateway ids.sus.c3sl.ufpr.br`, define o destino dos pacotes duplicados, a VM ids dentro da VLAN SUS.

A adição destas regras à cadeia "FORWARD", garante que os pacotes que vem da interface POP ou CCE, e tem como destino a interface do SUS, são copiados ao endereço do ids.

3.3 Arquitetura do projeto SUS

O projeto desenvolvido pelo C3SL em parceria com o SUS possui sua própria rede local virtual para separar o tráfego de rede destinado ao projeto. Esse isolamento, garante um nível maior de segurança. Nas seções anteriores, vimos que os pacotes chegam nessa VLAN através do roteamento feito pela Roble. Através do iptables, a Roble encaminha os pacotes destinados a interface SUS e faz o espelhamento para o IDS sem interromper o fluxo original.

Dentro da VLAN SUS, há duas máquinas físicas responsáveis pelo desenvolvimento do projeto: Arino e Paster. Ambas, hospedam máquinas virtuais responsáveis pelo desenvolvimento do projeto.

A Paster hospeda as seguintes máquinas virtuais:

- web: servidor web.
- listas: lista de e-mail.
- IDS: ponto em que o Suricata reside.

- log: pilha de armazenamento de alertas gerados pelo IDS.

E a Arino hospeda as seguintes máquinas virtuais:

- Email: servidor de e-mail.
- Httproxy: servidor proxy.

A Figura 3.2 ilustra essa arquitetura. Nela, podemos observar o fluxo de rede externo passando pela Roble e pelo switch malbec e chegando a sub rede do SUS. Dentro da sub-rede, as duas máquinas Arino e Paster orquestram suas máquinas virtuais. E por fim, a sub-rede SAPS, que possui um roteador interno, um banco de dados e um conjunto de máquinas kubernetes para a automação de deploy e aplicações containerizadas.

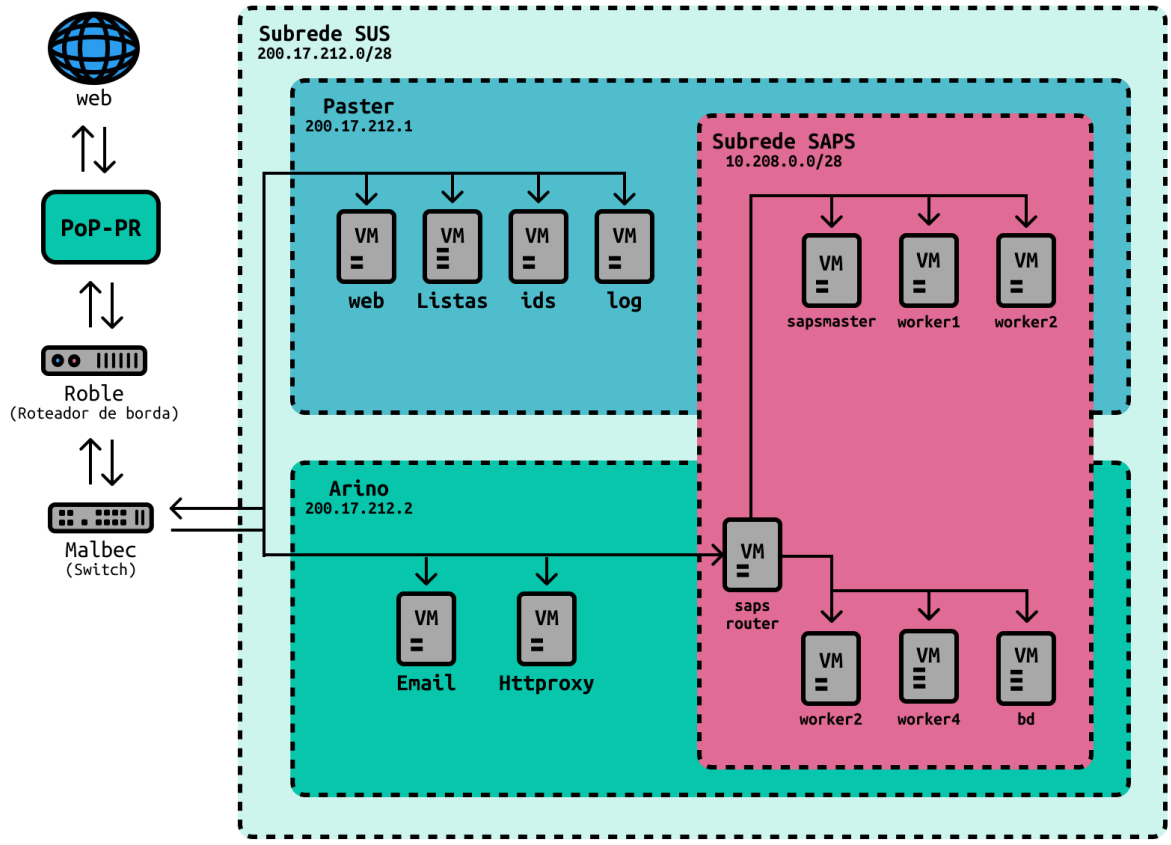


Figura 3.2: Máquinas SUS

O escopo deste trabalho irá focar nas máquinas virtuais ids e log, onde foram implantados o sistema de detecção de intrusão e a pilha de agregação de logs para armazenar a visualizar os alertas. Ambas as máquinas virtuais usam Debian GNU/Linux 12 (bookworm).

3.4 Software Livre

O C3SL desenvolve seus projetos de pesquisa empregando ferramentas de software livre. Os projetos são compartilhados para serem adaptados pela comunidade, a não ser que exista uma ressalva contratual (C3SL, 2024). O C3SL não apenas desenvolve código na filosofia

de software livre mas também opta por empregar ferramentas de software livre disponíveis na comunidade no desenvolvimento de deus projetos.

O uso de software livre permite que a comunidade audite o código-fonte. Assim, pessoas podem contribuir com a identificação de defeitos e brechas de segurança. A comunidade também pode identificar se os dados estão sendo tratados de forma segura e ética em conformidade (Perens, 1999). Softwares Livres aumentam a independência dos projetos de interessa públicos para que eles não fiquem dependentes de uma solução empresarial. Isso ocorre pois o projeto não fica refém repentinas pelo uso de suas soluções ou encerramento de suporte (Perens, 1999).

Portanto, levando em conta o contexto dos dados sensíveis do projeto SUS, escolhemos ferramentas de software livre para:

- Estar em conformidade com os princípios do C3SL;
- Tentar proporcionar um tratamento mais seguro dos dados como exigem as diretrizes da LGPD;
- Proporcionar independência de projeto de interesse público.

3.5 Justificando a escolha da ferramenta de IDS

Entre as várias ferramentas de IDS de software livre, as mais populares entre a comunidade, entre fóruns e trabalhos acadêmicos são o Bro-IDS, Snort e o Suricata (Lopez, 2014). O Bro-IDS é um IDS híbrido que possui características de SIDS e AIDS. Como o escopo do nosso trabalho foca em detecção por assinatura, não o consideramos como possível ferramenta. A seguir, apresentaremos os principais aspectos do Snort e do Suricata e iremos justificar a escolha da ferramenta.

3.5.1 Snort

O Snort é um dos SIDS de código aberto mais populares e difundidos pela comunidade. Ele foi feito em 1998 por Martin Roesch, e surgiu como um simples monitorador de pacotes de rede (*packet sniffer*) a nível de camada de aplicação. Baseado na biblioteca libcap, o Snort tornou-se uma aplicação modular que permite desenvolvedores criarem e acrescentarem novos recursos sem a necessidade de reescrever o motor principal de detecção (Albin, 2011).

De acordo com (Tenhunen, 2008), a arquitetura do Snort possui quatro componentes principais:

- Um monitorador de pacotes de rede;
- Um pre-processador;
- Um motor de detecção
- Plugins de saída

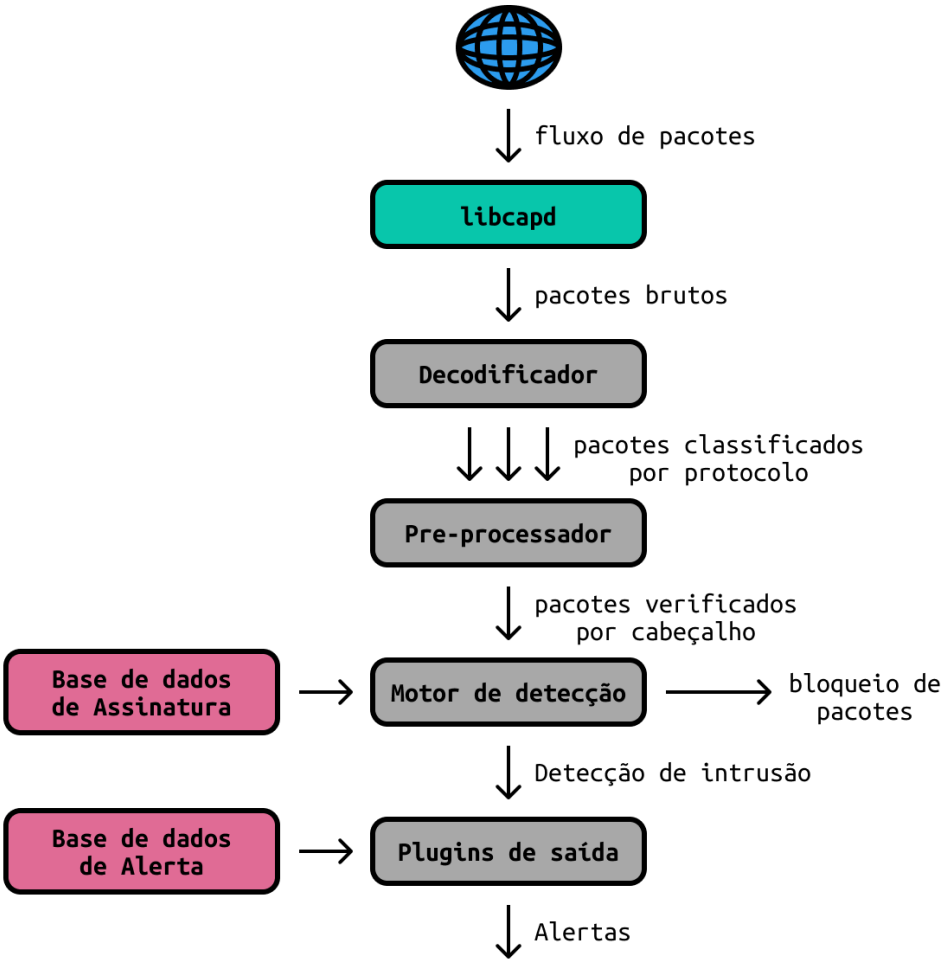


Figura 3.3: Arquitetura do Snort

A Figura 3.3 ilustra a arquitetura do sistema Snort. Os pacotes capturados da rede são ordenados e checados pelos decodificadores de pacote para tornar a análise mais eficiente. O motor usa os preprocessadores para normalizar o tráfego e o examinar para possíveis ataques ou comportamento anômalo. Quando os dados deixam o preprocessador, eles vão para o motor de detecção. Os dados são comparados com uma base de dados de assinatura. Os pacotes que casarem com as regras disparam alertas (Tenhunen, 2008).

O De acordo com o (Cisco, 2024b), o Snort pode ser usado de três formas:

1. como um *packet sniffer*, em que apenas lê os pacotes da rede e imprime o fluxo no terminal de saída.

2. como um logger de pacotes, em que não apenas lê mas armazena os pacotes de rede em formato pcap. Isso auxilia na identificação de problemas da rede.

3. como um IDS, que analisa os pacotes buscando casamento de assinaturas.

3.5.2 Suricata

Em 2009, a Open Information Security Foundation (OISF) lançou um novo SIDS de código aberto, o Suricata. Ele foi desenvolvido para ser o mecanismo de detecção de intrusão

mais moderno em relação ao Snort. Um avanço que o Suricata incorpora é um novo normalizador de Hyper-Text Transfer Protocol (HTTP). Ele é um analisador HTTP projetado para ser capaz de examinar o tráfego HTTP em busca de estratégias de ataque e técnicas de evasão usadas pelos invasores para contornar mecanismos comuns de IDS Albin (2011).

Outro avanço Suricata é a capacidade de operar multi-thread nativamente, algo que se torna cada vez mais necessário conforme a largura de banda da rede escala. Uma instalação padrão do Suricata pode processar o tráfego de rede a uma taxa de 100-200 megabits por segundo, e começa a descartar pacotes quando há sobrecarga dos recursos de rede. Para que o Snort consiga tirar proveito de uma arquitetura multiprocessadores, ele precisaria ser instanciado em cada CPU desejada, o que dificulta sua gerência. Isso porque o Snort foi um projeto de vários estágios, para thread único. Já o Suricata foi projetado para operar com vários CPUs (Albin, 2011).

O código do Suricata foi feito a partir do zero, mas os desenvolvedores não hesitam em dizer que a arquitetura foi inspirada no Snort (Albin, 2011). Portanto, a Figura 3.4 se assemelha muito a Figura 3.3, com a vantagem da arquitetura tirar proveito de multiprocessadores.

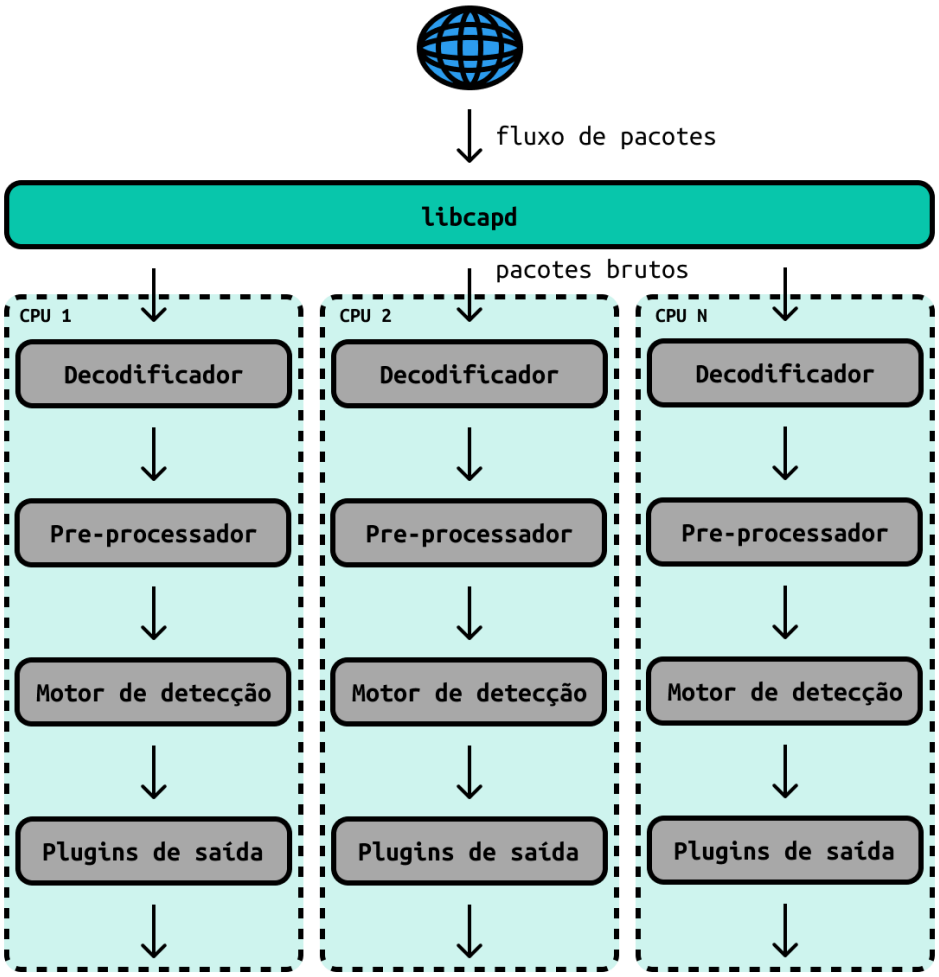


Figura 3.4: Arquitetura Multi-Thread do Suricata

3.5.3 Snort VS Suricata

Com o avançar do tempo, surgem métodos de intrusão mais sofisticados que usam tecnologia mais avançada, como por exemplo, máquinas com maior capacidade de processamento, maior largura de banda da rede. Portanto, surge também a necessidade de avanços nas técnicas de detecção de intrusão para acompanhar as ameaças à segurança em constante evolução e manter a segurança da rede crescente da organização.

Por conter vários threads no seu motor de detecção, o Suricata pode dividir de maneira inteligente o processamento e coordenar a detecção de assinaturas entre essas threads. De acordo com a Nielsen (2023), a tendência é que tenhamos um aumento de 50% na largura de banda de rede dos usuários a cada ano. Foi pensando em suprir essa demanda de rede que os desenvolvedores optaram pelo processamento multi-thread (Albin, 2011).

A Tabela 3.2 sumariza uma comparação entre o Snort e o Suricata.

Características	**Snort**	**Suricata**
Processamento Multi-Thread	Não	Sim
Detecção Automática do Protocolo	Não	Sim
Aceleração por GPU	Não	Sim
Consumo de Recursos	Maior uso de CPU e memória	Menor uso de CPU e Memória
Desempenho	Melhor em redes com tráfego intenso; suporta maiores volumes de dados.	Bom para redes menores ou com menos demanda.
Licença	GNU GPL v.2	GNU GPL v.2
Análise Offline	Sim, múltiplos arquivos simultaneamente	Sim, um único arquivo

Tabela 3.2: Comparação dos IDS de código aberto.

O Suricata consome mais CPU e Memória em relação ao Snort, o que torna seu uso não tão interessante em ambientes com recursos limitados. Porém, o Suricata proporciona um melhor aproveitamento de arquiteturas multi-thread, melhor processamento em grandes volumes de tráfego e possui funcionalidades modernas como normalização HTTP (Albin, 2011). Portanto, considerando o contexto do nosso projeto, escolhemos o Suricata como ferramenta IDS.

3.6 Operando o Suricata

3.6.1 Anatomia das Regras

As regras são a base do funcionamento do Suricata. Após analisar um conjunto de pacotes, o Suricata vai tentar fazer o casamento com alguma regra em sua base de dados. Por

tanto, precisamos entender como as regras são constituídas para que possamos entender bem o casamento de regras do Suricata.

Uma regra consiste nas seguintes partes:

- A ação, que determina o que acontece quando a regra corresponde.
- O cabeçalho, que define o protocolo, os endereços IP, as portas e a direção da regra.
- As opções da regra, que definem as especificidades da regra.

Cada regra é uma entrada em um que combina essas três partes sequencialmente:

```
ação cabeçalho (opções)
```

Expandindo o cabeçalho:

```
ação protocolo endereço_origem -> endereço_destino porta (opçõ
   es)
```

Um exemplo concreto:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"HTTP GET
   Request Containing Rule in URI"; flow:established,to_server;
    http.method; content:"GET"; http.uri; content:"rule";
   fast_pattern; classtype:bad-unknown; sid:123; rev:1;)
```

- `alert`: a ação diz que um alerta deve ser gerado.
- `http`: O cabeçalho diz que o alerta deve ser gerado em protocolos http.
- `$HOME_NET any ->` O cabeçalho também diz que a ação deve ser aplicada se os endereços de origem estiverem contidos na variável `$HOME_NET`, em quaisquer porta.
- `$EXTERNAL_NET any ->` Por fim, o cabeçalho diz que a ação deve ser aplicada se os endereços de destino estiverem contidos na variável `$EXTERNAL_NET`, em quaisquer porta.
- `(*)`: O conteúdo envolvido pelos parênteses são inúmeras opções adicionais sobre como casar a regra. Por exemplo, `content:"GET"`, significa que a regra procura especificamente pelo método `GET` no cabeçalho `HTTP`.

O Suricata suporta os seguintes tipos de ação:

- *alert* - gera um alerta.
- *pass* - interrompe a inspeção adicional do pacote.
- *drop* - descarta o pacote e gera um alerta.
- *reject* - envia erro RST/ICMP unreach para o remetente do pacote correspondente.
- *rejectsrc* - realiza a mesma função que o reject.
- *rejectdst* - envia pacote de erro RST/ICMP para o receptor do pacote correspondente.

- *rejectboth* - envia pacotes de erro RST/ICMP para ambos os lados da conversa.

  Ao usar o Suricata como IPS, qualquer ação também ativa o *drop*.
  O Suricata possui suporte para estes tipos básicos de protocolo:

- TCP
- UDP
- ICMP
- IP (ip significa “todos” ou “qualquer”)

  Há algumas opções adicionais de protocolo relacionadas ao TCP:

- tcp-pkt (para correspondência de conteúdo em pacotes tcp individuais)
- tcp-stream (para correspondência de conteúdo somente em um fluxo tcp remontado)

Há também alguns dos chamados protocolos de camada de aplicativo, ou protocolos de camada 7, que você pode escolher. Esses são: http (HTTP1 ou HTTP2), http1, http2, ftp, tls (isso inclui ssl), smb, dns, dhcp, ssh, smtp, imap, pop3, nfs, krb5, dhpc, snmp, websocket, tftp e outros.

A disponibilidade desses protocolos depende do fato de o protocolo estar ativado no arquivo de configuração, suricata.yaml. Se você tiver uma assinatura com o protocolo declarado como ’http’, o Suricata garante que a assinatura só corresponderá se o fluxo TCP contiver tráfego http.

3.6.2 Configurações e Instalação

Nesta subseção vamos apresentar mudanças pertinentes e resumir o processo de instalação adotado para esta prova de conceito. O objetivo não é reproduzir a documentação de instalação do Suricata, apenas apontar particularidades desta instância. Como dito anteriormente, a VM ids utiliza a versão 12 do Ubuntu. Após adicionar o repositório do OISF à lista de repositórios conhecidos e atualizar os pacotes no Ubuntu, instalamos os pacotes do Suricata através do gerenciador de pacotes do ubuntu, o apt. Não fizemos qualquer tipo de compilação do Suricata.

```
$ apt-get install software-properties-common
$ add-apt-repository ppa:oisf/suricata-stable
$ apt-get update
$ apt install suricata
```

A versão utilizada do Suricata 6.0.10. Este e outros detalhes da build do Suricata, como versão do compilador ou features disponíveis, podem ser conferidos ao invocar o Suricata com `-buildinfo`.

```
$ suricata --build-info
This is Suricata version 6.0.10 RELEASE
Features: NFQ PCAP_SET_BUFF AF_PACKET HAVE_PACKET_FANOUT
   LIBCAP_NG LIBNET1.1 HAVE_HTP_URI_NORMALIZE_HOOK PCRE_JIT
   HAVE_NSS HAVE_LUA HAVE_LUAJIT HAVE_LIBJANSSON TLS TLS_C11
   MAGIC RUST
```

```
SIMD support: none
Atomic intrinsics: 1 2 4 8 byte(s)
64-bits, Little-endian architecture
GCC version 12.2.0, C version 201112
compiled with _FORTIFY_SOURCE=2
L1 cache line size (CLS)=64
thread local storage method: _Thread_local
compiled with LibHTP v0.5.42, linked against LibHTP v0.5.42
...
```

Após isso, adicionamos o nome alternativo da interface de rede responsável pela captura no arquivo de configuração do Suricata `/etc/suricata/suricata.yaml`.

```
# Linux high speed capture support
 af-packet:
   - interface: enp0s18
```

Alteramos o caminho padrão da localização de regras do Suricata para facilitar na hora de baixar e explorar novas regras da comunidade.

```
default-rule-path: /var/lib/suricata/rules
rule-files:
  - suricata.rules
```

Após estes ajustes, reiniciamos o daemon do Suricata gerenciado pelo systemd e tornamos persistente a inicialização do Suricata após um eventual reboot do sistema.

```
$ systemctl enable suricata
$ systemctl restart suricata
```

Alteramos também o output padrão do Suricata para ser gerado em formato syslog. Isso é importante pois no futuro, os logs são enviados ao coletor através protocolo syslog, descrito pela RFC5424. O facility é um rótulo de identificação usado pelo syslog posteriormente.

```
# Extensible Event Format (nicknamed EVE) event log in JSON
   format
- eve-log:
   enabled: yes
   filetype: syslog
   identity: "suricata"
   facility: local5

...

# a line based alerts log similar to fast.log into syslog
- syslog:
   enabled: yes
   facility: local5
   level: info
```

Após feitas essas configurações, atualizamos as regras do Suricata com `suricata-update`. A operação padrão do `suricata-update` é usar o conjunto de regras Emerging Threats Open. Esse comando:

- Procura a versão do suricata.
- Procura por filtros a serem aplicados às regras baixadas.
- Faz download de um conjunto de regras Emerging Threats Open para a minha versão do Suricata
- Aplica filtros de ativação, desativação, eliminação e modificação conforme carregado acima.
- Escreve as regras em `/var/lib/suricata/rules/suricata.rules`.

3.7 Rsyslog

O termo Syslog é uma abreviação de *system logging protocol* (protocolo de registros do sistema). Ele é um padrão para coletar e armazenar informações de registros (logs). O Rsyslog é uma implementação do syslog. Ele pode ser configurado para receber entradas de registro do journal do systemd a fim de processá-las ou filtrá-las antes de serem escritas no disco ou mandadas pela rede (Wiki Archlinux, 2024b).

O arquivo `syslog.conf` é o principal arquivo de configuração do syslogd. Ele especifica as regras para os registros. Cada regra consiste em um campo seletor e um campo de ação. O campo seletor especifica padrões sobre a ação especificada.

Configuramos o Suricata para que os alertas fossem enviados ao syslog na facilidade 5. A partir disso, adicionei uma entrada no arquivo de configuração do Rsyslog para que enviasse seus logs da facilidade 5 para a VM de log, que foi usada para agregar e visualizar os dados. A nossa regra para enviar os alertas do Suricata é a seguinte:

```
local5.* action(type="omfwd" protocol="tcp" target="
   200.17.212.7" port="1514" Template="
   RSYSLOG_SyslogProtocol23Format" TCP_Framing="octet-counted"
   KeepAlive="on")
```

Quebrando a regra:

- `local5.*`: O campo seletor, representa a facilidade que será filtrada. que são todas as mensagens da facilidade 5.
- `action(...)`: O campo de ação. Ele define a ação que será executada sobre as mensagens que correspondem ao filtro. Ou seja, o que fazer com a mensagem filtrada da facilidade filtrada.
- `type="omfwd"`:, OMFWD é um acrônimo para syslog Forwarding Output Module. Este trecho significa que a ação vai utilizar este módulo responsável para enviar mensagens a um servidor remoto.
- `protocol="tcp"`: define que o envio vai utilizar o protocolo TCP.
- `target="200.17.212.7"`: endereço da VM log, que é o alvo do envio.
- `port="1514"`: porta destino onde a VM log está ouvindo.
- `Template="RSYSLOG_SyslogProtocol23Format"`: Define o formato da mensagem de log, protocolo syslog.

- `TCP_Framing="octet-counted"`: usa o tamanho da mensagem para delimitar os pacotes, de acordo com a RFC 5425, garantindo maior confiabilidade no envio.
- `KeepAlive="on"`: ajuda a detectar e gerenciar conexões inativas ou quebradas.

Essa configuração implementa um envio dos alertas do Suricata através do protocolo syslog, descrito pela RFC 5424, para a VM ids. Nessa VM, um coletor estará ouvindo na porta 1514, esperando por alertas.

3.8 Justificando a escolha da pilha de agregação e visualização de logs

Dada a natureza e relevância do projeto, podemos presumir que haverá um grande volume de tráfego de rede e um potencial número de alertas que podem ser gerados pelo Suricata. Portanto, surge a necessidade de gerenciar o volume de dados que pode ser gerado com o decorrer do tempo. Um dos objetivos deste trabalho é implantar um ambiente que garanta um bom nível de compressão dos dados, permitindo escalabilidade e flexibilidade.

Uma vez que os alertas são gerados pelo Suricata, é importante que o analista possua uma forma eficaz de visualizar os alertas. A visualização aprimora o processo de monitoramento ao permitir uma compreensão maior do comportamento da rede (Lokiny, 2023). A análise das métricas da rede auxilia no processo de investigação, solução de problemas e, posteriormente, ajuste fino das regras.

No passado, administradores de sistema utilizavam ferramentas simples como grep, awk ou perl para analisar logs. Com a popularização de aplicações em nuvem, crescimento de infraestruturas de sistemas de informação e aumento da largura de banda de rede, a análise de registros começou a se tornar mais complexa. Com a quantidade de logs sendo produzida, não é viável deixar a análise ser feita de forma manual. Soluções modernas propõem ferramentas que consigam armazenar os registros, coletá-los de múltiplas fontes, fazer uma análise sintática e propor visualizações personalizadas (Eriksson e Karavek, 2023).

Duas soluções de software livre comuns são a pilha Elasticsearch, Logstash e Kibana (ELK) e a pilha Promtail, Grafana e Loki (PLG). Ambas as soluções possuem três principais componentes e podem coletar dados de diferentes fontes e formatos, permitindo pesquisa, análise e visualização em tempo real.

3.8.1 A Pilha ELK

O centro da pilha ELK é o Elasticsearch, que é o mecanismo de análise e pesquisa de registros. Foi projetado para ser escalável horizontalmente, ou seja, pode lidar com maiores quantidades de dados ao adicionar novos nós para distribuir a carga de trabalho. Antes dos dados serem indexados no Elasticsearch, eles são analisados e normalizados pelo agente de acompanhamento, o Logstash, em um processo chamado ingestão de dados (Elastic, 2024c).

O processo de ingestão de dados envolve a extração de informações relevantes dos dados brutos, formatação e estruturação dos dados e adição de contexto para aprimorar os dados. O Logstash unifica dados de diferentes fontes, faz o processo de ingestão e os envia para diferentes nós. Após a indexação, usuários podem realizar consultas para recuperar resumos usando agregações com a linguagem de consulta Kibana (Elastic, 2024b).

O Kibana é o componente final da pilha ELK, que gera visualizações e painéis sobre os dados armazenados no Elasticsearch. Ele possui uma interface web que permite a interação com

os dados, gerando painéis dinâmicos, histogramas e tabelas que podem permitir uma melhor compreensão dos dados (Elastic, 2024a). A pilha ELK está ilustrada na Figura 3.5.

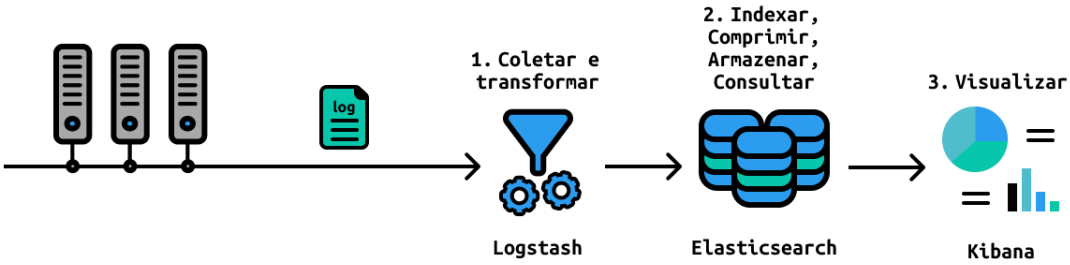


Figura 3.5: A Pilha ELK

3.8.2 A Pilha PLG

O Grafana Loki, ou simplesmente Loki, é uma ferramenta de código aberto de agregação de logs. Assim como o Elasticsearch, ele também é horizontavelmente escalável, porém, seu funcionamento é diferente. O Loki foi projetado com base na ideia de indexar apenas os metadados sobre rótulos dos registros. Ele também é horizontalmente escalável. Os dados de registro em si são compactados e armazenados em blocos na forma de armazenamento de objetos, como o serviço S3 da Amazon, Google Cloud Storage (GCS) ou localmente no sistema de arquivos (Labs, 2024b).

Os rótulos metadados que descrevem o fluxo de log na forma de chave-valor. O mecanismo de indexação do Loki se chama Single Store Loki. Ele simplifica as operações e reduz o custo de execução do Loki, exigindo apenas armazenamento para os índices e os blocos. A decisão de indexar apenas os metadados, ao invés do texto completo, poderia afetar a velocidade das consultas aos registros. Portanto, o Loki emprega paralelização para aumentar a velocidade de processamento (Eriksson e Karavek, 2023).

O Promtail é o agente responsável por coletar, transformar e filtrar os arquivos de log e encaminhá-los ao Loki, onde serão indexados e armazenados (Labs, 2024c). Análogo ao Logstash, ele também consegue coletar registros de várias fontes. De acordo com (Labs, 2024c), o Promtail usa pipelines para processar os registros. Essa pipeline se divide em quatro estágios:

1. Parsing: é feita uma análise sintática das linhas de logs, extraindo os dados em uma estrutura interna para que fiquem disponíveis para os próximos estágios.

2. Transformação: Os dados extraídos da análise são estruturados e padronizados.

3. Ação: diversas ações podem ser executadas mas o padrão é que os rótulos são definidos, dando-lhes nomes descritivos.

4. Filtragem: é um estágio opcional em que é possível filtrar dados com base em fatores configurados previamente. Por exemplo, realizar os estágios 1,2 e 3 com descarte dos dados para uma ou mais condições.

O Grafana é uma plataforma para visualização, análise e monitoramento de dados, análogo ao Kibana no ELK. O Grafana permite a criação de painéis e gráficos em tempo real para visualizar e analisar métricas. O Loki utiliza o LogQL, uma linguagem de consulta que filtra e recupera o conteúdo da linha de registro usando rótulos. Essas consultas podem ser executadas e visualizadas no Grafana (Labs, 2024a). A Figura 3.6 ilustra a arquitetura PLG.

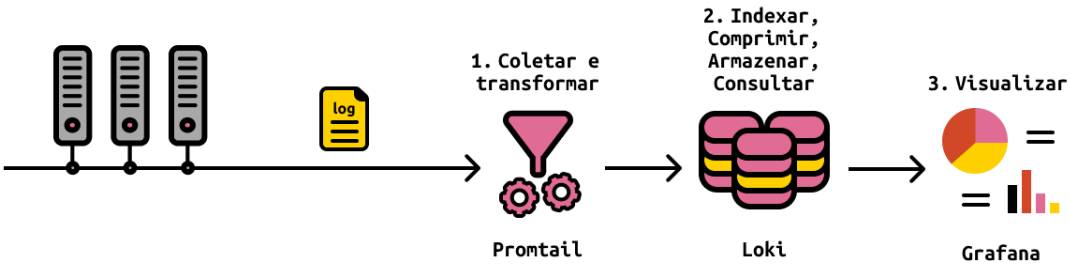


Figura 3.6: A Pilha PLG

3.8.3 A Pilha ELK VS A Pilha PLG

Em relação aos agentes que coletam os arquivos de log, Logstash e Promtail, ambos os agentes fornecem recursos muito semelhantes como coleta de diferentes fontes, análise sintática e envio para as soluções de armazenamento. O Logstash se destaca por oferecer mais opções de entrada de dados e por oferecer conversão de tipos de dados nos campos extraídos (Eriksson e Karavek, 2023).

Comparando a criação de visualizações e painéis, o Kibana permite que eles sejam criados mais facilmente. No Grafana, algumas consultas simples acabam se tornando complexas de serem realizadas, pois o Kibana oferece uma interface mais amigável que o Grafana e o Loki não apresenta um bom desempenho com um número grande de rótulos, exigindo com que algumas consultas sejam adaptadas. Apesar disso, o Grafana oferece mais opções de personalização e customização, pois há uma grande variedade de opções de formatação (Eriksson e Karavek, 2023).

Os painéis no Kibana são interativos por padrão, o que significa que os usuários podem clicar em um valor de campo/termo específico em uma visualização e o painél filtrará automaticamente todos os outros dados não relacionados a esse termo. Enquanto no Grafana, a criação de painéis interativos exige mais configurações e adaptações das consultas. Essa flexibilidade do Grafana pode ser uma vantagem ou desvantagem, depende de como o administrador pretende usar a plataforma (Eriksson e Karavek, 2023).

Tanto o Kibana quanto o Grafana são capazes de criar painéis compartilháveis por meio de importação ou exportação, o que pode ser útil na hora de atualizar ou reinstalar o software. Mas o Grafana leva a vantagem de fornecer código compartilhável aos usuários, o que permite o compartilhamento dos painéis sem precisar baixá-los (Eriksson e Karavek, 2023).

Apesar das vantagens de visualização de painéis e métricas, o trabalho de (Eriksson e Karavek, 2023) sugere que a Pilha PLG é uma solução mais eficiente em termos de uso de CPU e memória. As primeiras comparações foram em indexações de um fluxo de log com mil linhas, em que o Loki se mostrou mais rápido e eficiente. Em arquivos de 5GB, ele ainda performou

uma indexação mais rápida, com a compensação de maior uso de CPU e memória. Mas ainda, consegue ser mais eficiente em relação à pilha ELK. Por fim, em termos de armazenamento, a pilha PLG exigia menos espaço de disco para armazenar logs indexados em comparação ao ELK. A Figura 3.3 sumariza as principais vantagens e desvantagens entre as duas pilhas:

A Pilha ELK	**A Pilha PLG**
Vantagens (+)	
+ Consultas mais rápidas e suporte a consultas complexas.	**+** Mais eficiente em uso de CPU e memória.
+ Configuração mínima para funcionamento.	**+** Indexação mais rápida de arquivos.
+ Interface amigável ao usuário.	**+** Requer menos espaço em disco para armazenar logs.
+ Dashboards interativos por padrão.	**+** Visualizações altamente personalizáveis.
Desvantagens (-)	
- Uso elevado de CPU e memória ao indexar logs.	**-** Documentação limitada e suporte reduzido.
- Indexação mais lenta para arquivos de 5 GB.	**-** Consultas mais lentas e falhas ao executar algumas.
- Ocupa mais espaço em disco para a mesma quantidade de logs.	**-** Configuração complexa para habilitar funcionalidades.
- Menos opções de personalização visual nos dashboards.	**-** Maior propensão a erros em volume elevado de requisições.

Tabela 3.3: Comparação entre a Pilha ELK e a Pilha PLG

Apesar da complexidade inicial de configuração e do tempo de aprendizado da Pilha PLG, há vantagens de eficiência de recursos de CPU e memória na hora de indexar os logs, vantagens de espaço em disco e maior flexibilidade na criação de painéis. Considerando que a natureza do projeto, em que a tendência é a geração de um grande número de alertas, decidimos que as vantagens oferecidas pela pilha PLG poderiam ser mais adequadas.

3.8.4 Topologia do Trabalho

Nesta seção, vamos observar uma visão geral da topologia do nosso trabalho para entender aonde cada ferramenta atua e aonde os processos ocorrem. A Figura 3.7 ilustra o processo desde a chegada dos pacotes pela rede externa até a visualização dos alertas.

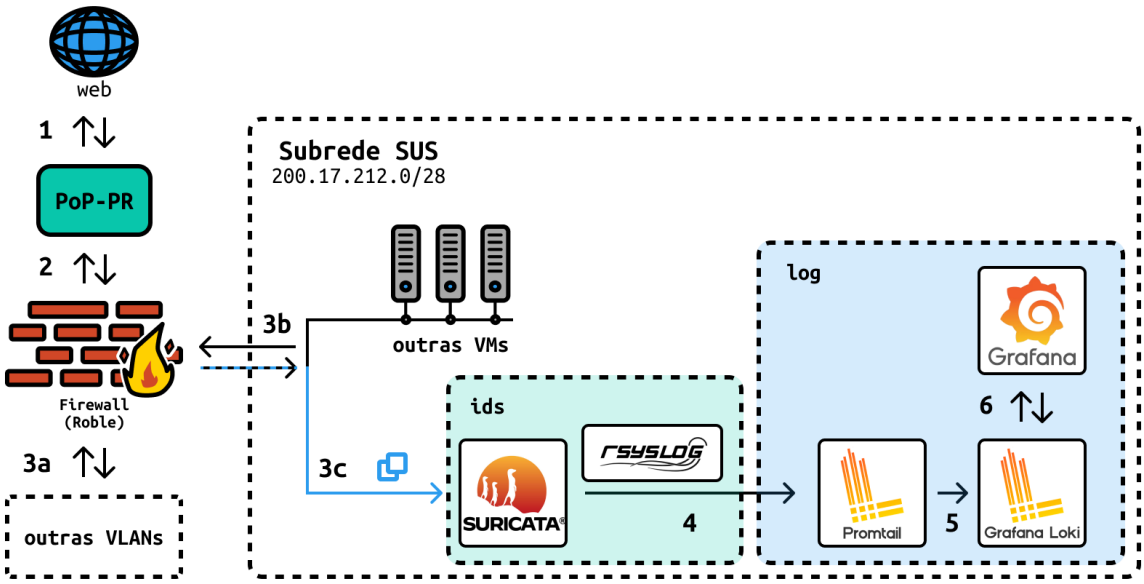


Figura 3.7: Arquitetura

1. O tráfego externo, considerado não confiável, chega até o PoP-PR.
2. O PoP-PR encaminha os pacotes destinados ao DInf para a Roble, o roteador de borda e firewall.
3. Os pacotes são filtrados pelas tabelas, cadeias e regras do iptables. Após o iptables definir o destino final de um pacote, este pode:
   a. ser destinado para outras VLANs não relacionadas ao projeto. Não estamos interessados nestes pacotes.
   b. Ser destinado às VMs do SUS.
   c. Ser espelhado à VM ids caso o destino original seja uma VM do SUS.
4. O Suricata analisa os pacotes e gera os alertas. Os alertas são enviados para a pilha de agregação de logs através do protocolo syslog, descrito pela RFC5424. Eles serão coletados pelo Promtail.
5. Uma vez que o Promtail coletou os alertas gerados pelo Suricata, ele faz o processo de ingestão de logs e os envia para o Promtail.
6. O Grafana consulta o Promtail para gerar visualizações e painéis sobre os dados coletados.

Capítulo 4

Resultados

Nas seções anteriores, justificamos nossas escolhas de tecnologia, mostramos todo o caminho feito pelos pacotes desde rede externa até a visualização e detalhamos como as tecnologias escolhidas funcionam. Agora, iremos falar sobre os dados coletados e a visualização das métricas. A estrutura geral do projeto está implantada desde o dia dez de outubro de 2024. O período dos dados analisados nesta seção começa em 10/out/24 e se estende até 11/dez/24.

Neste período, foram gerados um total de 115.038.773 alertas, uma média de 14.379.847 por semana, como pode ser visto na Figura 4.1.

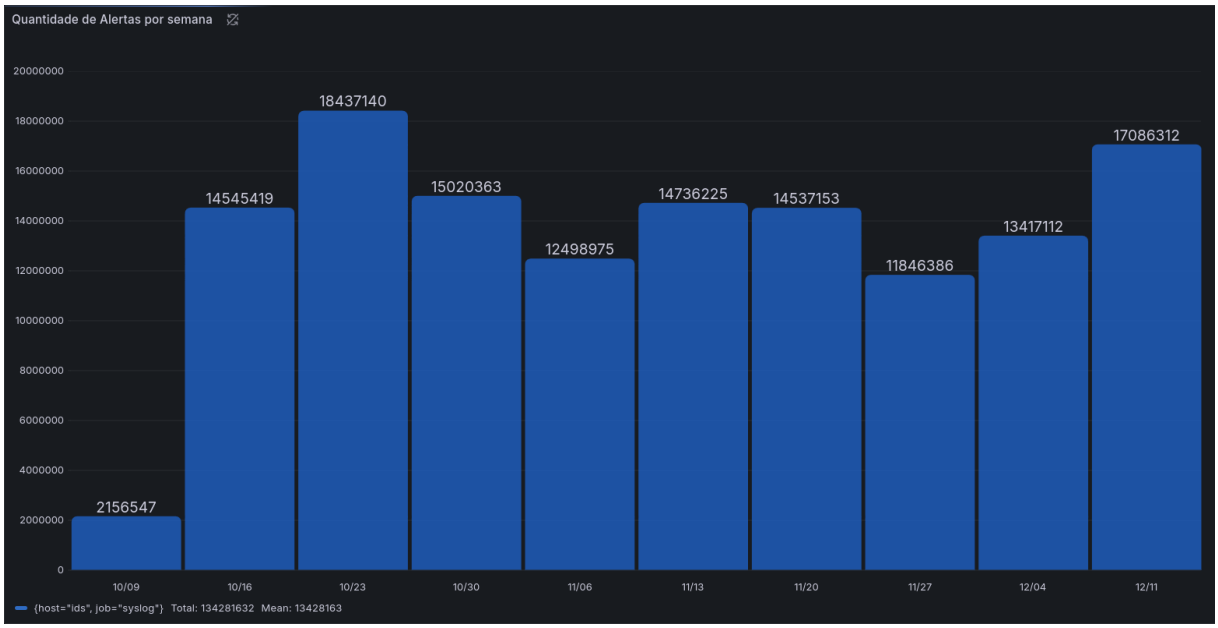


Figura 4.1: Distribuição dos Logs ao longo das semanas

Cada registro no Suricata é um JSON que possui uma "estrutura comum". Essa estrutura comum possui uma série de campos, como timestamp, dest_ip, src_ip, que podem ser usados para analisar e classificar o tráfego de rede. O campo `event_type` ajuda a categorizar o tipo de registro gerados durante a coleta do tráfego de rede. Cada evento corresponde a uma funcionalidade específica que o Suricata está registrando. Em nossa amostragem, os principais tipos de registro foram:

1. `flow`: Representa apenas um fluxo de rede. Uma comunicação bidirecional entre dois dispositivos finais na rede. Ele mantém apenas os metadados como endereços IP, portas, protocolos, etc.

2. `alert`: É um registro gerado quando há o casamento de uma regra na base de dados. No nosso caso, as regras da base de dados do Emerging Threats. Além dos metadados do fluxo de rede, contém uma mensagem do alerta, identificador da regra e contexto da violação. Um alerta indica um **possível** ataque, atividade suspeita ou violação das regras de rede.

3. `stats`: Métricas de desempenho do Suricata. São metadados sobre as estatísticas do motor de detecção como pacotes processados, taxa de perda dos pacotes, uso de memória e CPU.

4. `ssh`: Eventos específicos relacionados ao protocolo SSH. Metadados sobre as sessões como versão do protocolo, algoritmo de criptografia usado.

Os registros do tipo `flow` compõem a grande maioria dos registros de nossa amostragem, uma marca de 99%. Porém, 801.128 registros são eventos do tipo alerta. O que mostra o grande volume registrado pelo Suricata nessa janela de tempo. Essa estatística está representada pela Figura 4.2

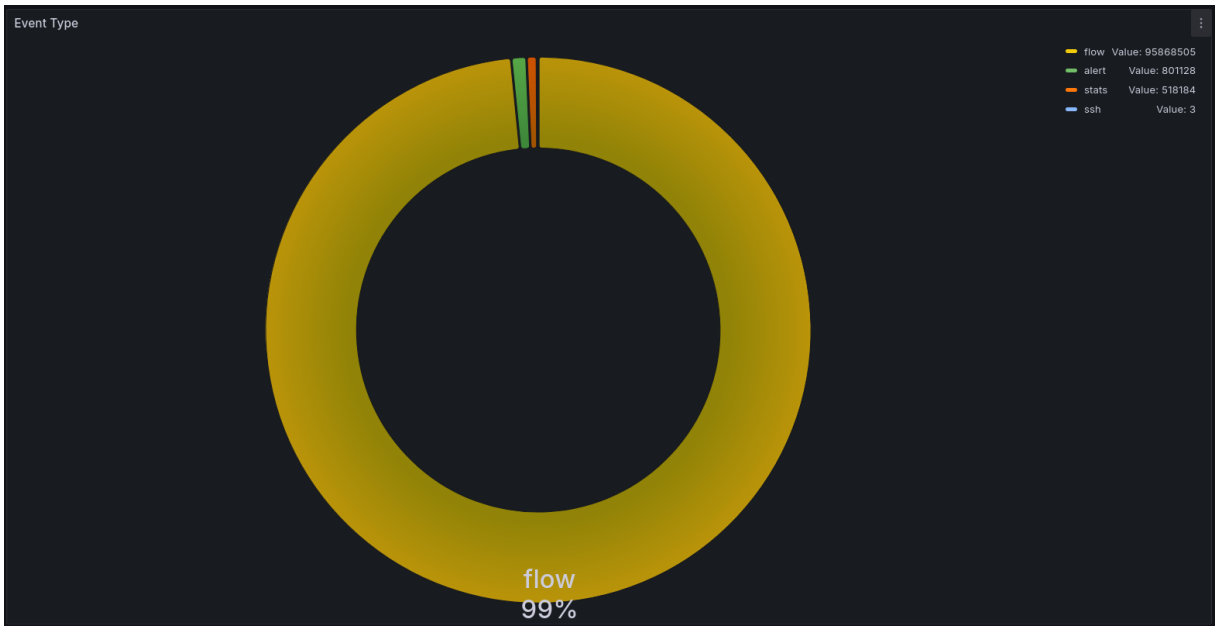


Figura 4.2: Distribuição dos Eventos de Log

No escopo deste trabalho, iremos analisar somente os registros do tipo `alert`.

4.1 Registros do tipo Alerta

Os registros do tipo `alert` possuem inúmeros campos com dados sobre uma regra que foi acionada. Os campos são:

- `action`: Indica a ação tomada sobre o pacote que acionou esse registro, sendo que `drop` ou `reject` cabem apenas quando o Suricata está no modo IPS. No estudo de caso, esta ação sempre estará como `allowed`, ou seja, o pacote foi permitido.

- `gid`: Origem da regra ou evento que gerou o alerta, ajudam a rastrear a fonte do alerta, podendo ajudar na hora de debbuging ou personalização das regras.

- `rev`: Versão da regra, ajuda a garantir que estamos analisando a versão correta da regra.
- `signature_id`: Identificador único da regra.
- `signature`: Texto ou descrição daquela regra.
- `category`: Categoria à qual a regra pertence. Isso permite agrupar alertas relacionados, facilitando a priorização e análise.
- `severity`: nível de gravidade associado ao alerta, em que `1` significa alta gravidade, `2` moderada e `3` baixa prioridade.
- `metadata`: Informações adicionais e contexto sobre a ameaça detectada.

4.1.1 Filtrando por categoria de ataque

Podemos fazer consultas filtrando por campos específicos para definir diferentes estratégias de defesa. Por exemplo, podemos fazer uma consulta em nossa pilha de agregações, separando os alertas pelo campo `category`, como ilustrado na figura 4.3.

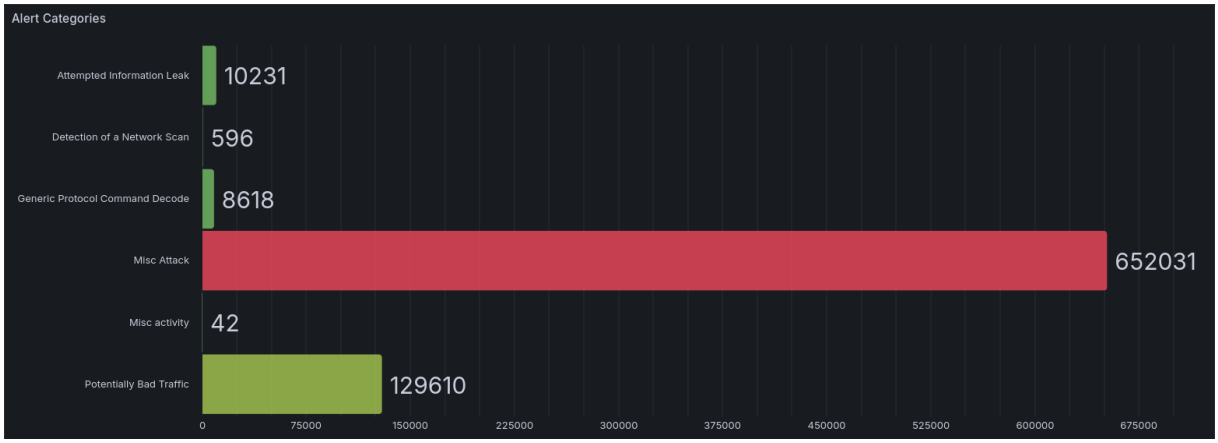


Figura 4.3: Categorias de alerta

Ao filtrar por categoria, podemos identificar tendências e focos de análise. Uma abordagem possível é a priorização de categorias de alto volume, pois indicam onde estão ocorrendo atividades mais frequentes em nossa rede. Em nossa amostragem, podemos observar a categoria *Misc Attack* (652.031) sendo a responsável pelo maior número de alertas e a *Potentially Bad Traffic* (129.610) como a segunda maior.

Misc Attack inclui alertas que indicam ataques gerais ou diversos que não se enquadram em categorias mais específicas. Podem incluir padrões de ataque incomuns ou menos definidos ou técnicas que visam vulnerabilidades em protocolos de rede, aplicativos ou dispositivos (Gonzalez, 2018). *Potentially Bad Traffic* incluem tráfego incomum ou minimamente suspeito, mas não viola de forma clara as regras ou políticas. São pacotes malformados, uso incomum de alguns protocolos, ou tráfego que geralmente está associado a comportamento malicioso.

Por exemplo, podemos pegar os alertas de Misc Attack, dividir por endereços de IP origem e destino. Se a maior parte dos endereços de IP forem de instituições confiáveis ou da rede interna, podemos acrescentar exceções para estes casos.

Nem sempre concentrar esforços nas categorias de alto volume é uma estratégia ideal. Isto porque, muitas vezes, estas categorias contêm falsos positivos – tráfego legítimo mal

classificado ou eventos genéricos que não representam ameaça crítica. Portanto, outra abordagem possível é justamente ir nas categorias de baixo volume. Essa abordagem procura eliminar atividades maliciosas mais específicas, minando os ruídos dessas categorias antes de lidar com o volume principal e mais genérico.

Em nossa amostragem são as *Detection of a Network Scan* ou *Attempted Information Leak*. O principal objetivo de um *Detection of a Network Scan* é coletar informações sobre as portas de um endereço de rede. Esse ataque tenta descobrir quais estão abertas, fechadas e filtradas, para depois tentar explorá-las (Gonzalez, 2018). A regra de tentativa de vazamento de informações trata de assinaturas de tentativas de coleta de informações potencialmente prejudiciais. Ataques de reconhecimento classificados como tentativa de vazamento de informações não são prova positiva de que uma tentativa de coleta de informações foi bem-sucedida. Mas há a possibilidade de sucesso do atacante (O'Reilly, 2024).

4.1.2 Filtrando por assinatura

Outro filtro pode ser feito no campo `signature`, como ilustrado na figura 4.4. Ao saber as assinaturas mais frequentes, podemos adicionar regras que derrubem ou rejeitem no *Firewall*, analisar se estes mesmos IPs ou padrões aparecem em outras atividades dentro da rede ou aferir se não é um falso positivo, para evitar a repetição de alertas irrelevantes.

Alert Signatures

Time	alert_signature	Value #A ↓
2024-12-03 23:59:59.000	ET DROP Dshield Block Listed Source group 1	370663
2024-12-03 23:59:59.000	ET SCAN Suspicious inbound to MSSQL port 1433	61864
2024-12-03 23:59:59.000	ET SCAN Suspicious inbound to mySQL port 3306	35801
2024-12-03 23:59:59.000	ET SCAN Suspicious inbound to PostgreSQL port 5432	19731
2024-12-03 23:59:59.000	ET DROP Spamhaus DROP Listed Traffic Inbound group 24	17114
2024-12-03 23:59:59.000	ET SCAN Suspicious inbound to Oracle SQL port 1521	12214
2024-12-03 23:59:59.000	ET CINS Active Threat Intelligence Poor Reputation IP group 34	11909
2024-12-03 23:59:59.000	ET CINS Active Threat Intelligence Poor Reputation IP group 29	11881
2024-12-03 23:59:59.000	ET CINS Active Threat Intelligence Poor Reputation IP group 35	11878
2024-12-03 23:59:59.000	ET CINS Active Threat Intelligence Poor Reputation IP group 30	11873
2024-12-03 23:59:59.000	ET CINS Active Threat Intelligence Poor Reputation IP group 31	11846
2024-12-03 23:59:59.000	ET CINS Active Threat Intelligence Poor Reputation IP group 32	11821
2024-12-03 23:59:59.000	ET CINS Active Threat Intelligence Poor Reputation IP group 33	11809
2024-12-03 23:59:59.000	ET CINS Active Threat Intelligence Poor Reputation IP group 36	11799
2024-12-03 23:59:59.000	ET CINS Active Threat Intelligence Poor Reputation IP group 28	9355
2024-12-03 23:59:59.000	ET SCAN Potential SSH Scan	7816
2024-12-03 23:59:59.000	ET 3CORESec Poor Reputation IP group 2	6568
2024-12-03 23:59:59.000	ET CINS Active Threat Intelligence Poor Reputation IP group 76	6451

Figura 4.4: Principais assinaturas casadas

Falando um pouco de nossa amostragem, em primeiro lugar, com 370.663 alertas gerados, *ET DROP Dshield Block Listed Source group 1*. Esse alerta indica que um endereço IP de origem foi identificado na lista de bloqueio do Dshield, uma fonte que rastreia endereços IP maliciosos. O tráfego deste IP está geralmente associado a ataques de força bruta, varreduras ou tentativas de exploração de vulnerabilidades. Para este caso, poderíamos criar uma regra no roteador de borda, a Roble. Se este endereço for de origem legítima, podemos melhorar nossas regras para evitar alertas desnecessários.

Em segundo lugar, *ET SCAN Suspicious inbound to MSSQL port 1433* (61,864). Indicam conexões mal formadas ou suspeitas à porta 1433, comumente utilizada pelo Microsoft SQL Server. Podemos, por exemplo, limitar a taxa de acessos a esta porta para diminuir tentativas

de força bruta ou varredura. Logo em seguida, *ET SCAN Suspicious inbound to MySQL port 3306* (35,801), similar a anterior, mas desta vez com a porta alvo 3306, utilizada pelo MySQL. *ET SCAN Suspicious inbound to PostgreSQL port 5432* (19,731), novamente, o mesmo dos anteriores, mas desta vez a porta 5432 é o alvo. A partir destes três casos, que estão entre os mais frequentes, é possível notar um padrão de ataques. Podemos utilizar essa informação no afunilamento de nossas regras ou resolver no *Firewall*. Por fim, *ET DROP Spamhaus DROP Listed Traffic Inbound group 24* (17.114), atividades maliciosas ligadas a um IP mapeado por Spamhaus. Comportamento muito parecido ao alerta que mais aparece.

Este filtro pode ser somado ao filtro de severidade. Assim, podemos consultar as assinaturas que mais aparecem mas com severidades mais altas, portanto mais preocupantes.

4.2 Discussão

Com base nas métricas analisadas, foi possível observar que os pacotes foram espelhados com sucesso para o IDS, dado o imenso volume de tráfego analisado e de alertas gerados. Apesar do enorme volume de alertas gerados em 2 meses e meio, o uso do sistema de arquivos por parte da Pilha PLG corresponde a cerca de 1% do total disponibilizado para a VM. Isso indica uma eficiência de compressão de dados satisfatória para os requisitos do projeto. A Pilha PLG também conseguiu proporcionar visualização e criação de painéis que podem auxiliar na definição de estratégias de defesa e mitigação de ataques.

Portanto, apesar da dificuldade inicial de configuração, documentação limitada e curva lenta de aprendizado na criação de painéis, a solução proposta pode ser considerada adequada à natureza do projeto. Isto porque, uma vez aprendida, a ferramenta apresenta um bom uso dos recursos de hardware e consegue gerar os painéis necessários para visualização das métricas.

Como trabalhos futuros, podemos aprimorar a solução implementada por meio da redução de ruídos, afunilamento e otimização das regras. Há um grande volume de alertas que podem ser ruídos, ou falso positivos, pois as regras utilizadas foram genéricas do *Emerging Threats Open*. É necessário afunilar e otimizar as regras para facilitar a tomada de decisões de priorização de defesa por parte dos analistas.

Outro passo no futuro, seria a geração de estatísticas da rede. As estatísticas da rede podem ser utilizadas para identificar comportamento anômalo, permitindo criar uma solução híbrida ao combinar aspectos SIDS com AIDS. Há trabalhos que mostram redução significativa de falsos positivos, cerca de 90% (Khraisat et al., 2019).

Ainda, há a necessidade de salvar os pacotes junto aos alertas. Isso permitirá analisar os pacotes junto aos alertas e tirar conclusões mais precisas, e pode criar uma base de dados que pode ser utilizadas em análises pela comunidade acadêmica no aprendizado de segurança computacional.

Apesar dessa gama de trabalhos a fazer, devemos notar que, estes trabalhos de aprimoramento poderão ser feitos a partir desta monografia. Em outras palavras, esse trabalho de coleta de dados e geração de métricas foi necessário ser implantado como solução inicial antes que refinamentos da solução possam surgir. Assim, melhorando a defesa do projeto em parceria com o SUS.

Sumarizando, podemos concluir que:

1. O espelhamento de tráfego funciona adequadamente.

2. As ferramentas propostas podem ser consideradas satisfatórias para a natureza do projeto.

3. Há desafios para o futuro, mas, este trabalho foi o primeiro passo para permitir que os desafios possam ser executados no futuro.

Referências Bibliográficas

(2024). *IPS. vs. IDS vs. Firewall: What Are the Differences?* Palo Alto Networks. Acessado em 14/11/2024.

Albin, E. (2011). A comparative analysis of the snort and suricata intrusion-detection systems. Dissertação de Mestrado, Naval Postgraduate School, Monterey - California.

Amoroso, E. (1994). *Fundamentals of Computer Security Technology*. Prentice Hall PTR.

Anderson, J. P. (1980). Computer security threat monitoring and surveillance. Relatório técnico, James P. Anderson Co.

Ashoor, A. S. e Gore, S. (2011). Importance of intrusion detection system (ids). *International Journal of Scientific Engineering Research*, 2(1).

Biju, J. M., Gopal, N. e Prakash, A. J. (2019). Cyber attacks and its different types. *International Research Journal of Engineering and Technology*, 6(3).

BRASIL (2024). Esther dweck destaca importância do plano de ia do governo e da carteira de identidade nacional para a integração e proteção de dados. `https://www.gov.br/servidor/pt-br/assuntos/noticias/2024/agosto/esther-dweck-destaca-importancia-do-plano-de-ia-do-governo-e-da-carteira-de-identidade-nacional-para-a-integracao-e-protecao-de-dados`. Acessado em 11/11/2024.

BRASIL (2024). Registro de incidentes com dados pessoais. `https://www.gov.br/saude/pt-br/acesso-a-informacao/lgpd/registro-de-incidentes-com-dados-pessoais`. Acessado em 20/11/2024.

C3SL (2024). Software livre. https://www.c3sl.ufpr.br/2021-o-c3/2021-software-livre/. Acessado em 11/11/2024.

Cisco (2024). Dhcp overview. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr$_dhcp/configuration/xe-3s/asr903/dhcp-xe-3s-asr903-book/dhcp-overview.pdf. Acessadoem26/11/2024.

Cisco (2024a). What is a firewall? https://www.cisco.com/site/us/en/learn/topics/security/what-is-a-firewall.html. Acessado em 12/11/2024.

Cisco (2024b). What is snort? https://www.snort.org/. Acessado em 12/11/2024.

Cloudflare (2024). What is defense in depth? | layered security. https://www.cloudflare.com/learning/security/glossary/what-is-defense-in-depth/. Acessado em 16/11/2024.

CVE (2024). About the cve program. https://www.cve.org/About/Overview. Acessado em 26/11/2024.

Elastic (2024a). Kibana: Explore, visualize, discover data | elastic. https://www.elastic.co/kibana. Acessado em 23/11/2024.

Elastic (2024b). Logstash introduction. https://www.elastic.co/guide/en/logstash/8.16/introduction.html. Acessado em 23/11/2024.

Elastic (2024c). What is elasticsearch? https://www.elastic.co/guide/en/elasticsearch/reference/8.16/elasticsearch intro-what-is-es.html. Acessado em 23/11/2024.

Eriksson, J. e Karavek, A. (2023). *A comparative analysis of log management solutions: ELK stack versus PLG stack*. Tese de doutorado, School of Innovation, Design and Engineering, Västerås, Sweden.

Gonzalez, R. (2018). Suricata 5, 6, 7 rule categories. https://community.emergingthreats.net/t/suricata-5-6-7-rule-categories/94. Acessado em 10/12/2024.

Kaspersky (2024a). O que é engenharia social? https://www.kaspersky.com.br/resource-center/definitions/what-is-social-engineering. Acessado em 19/11/2024.

Kaspersky (2024b). O que é injeção de sql? definição e explicação. https://www.kaspersky.com.br/resource-center/definitions/sql-injection. Acessado em 15/11/2024.

Khraisat, A., Gondal, I., Vamplew, P. e Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(20):801–823.

Labs, G. (2024a). Grafana | query, visualize, alerting observability platform. https://grafana.com/grafana/. Acessado em 10/12/2024.

Labs, G. (2024b). Loki overview | grafana loki documentation. https://grafana.com/docs/loki/latest/get-started/overview/. Acessado em 10/12/2024.

Labs, G. (2024c). Promtail agent | grafana loki documentation. https://grafana.com/docs/loki/latest/send-data/promtail/. Acessado em 10/12/2024.

Lokiny, N. (2023). Monitoring and observability solutions for cloud-native applications. *European Journal of Advances in Engineering and Technology*, 10(3):66–69.

Lopez, M. E. A. (2014). Uma arquitetura de detecção e prevenção de intrusão para redes definidas por software. Dissertação de Mestrado, Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia - UFRJ, Rio de Janeiro, RJ – Brasil.

Maziero, C. (2019). Segurança computacional. `https://wiki.inf.ufpr.br/maziero/lib/exe/fetch.php?media=sc:seg-texto-01.pdf`. Acessado em 11/11/2024.

Nielsen, J. (2023). Nielsen's law of internet bandwidth. https://www.nngroup.com/articles/law-of-bandwidth/. Acessado em 10/12/2024.

O'Reilly (2024). Attempted information leak. https://www.oreilly.com/library/view/intrusion-detection-with/157870281X/157870281X$_{a}pp02lev1sec4.html.Acessadoem10/12/2024$.

Perens, B. (1999). *The open source definition*. Open sources: voices from the open source revolution 1.

Pressman, R. (2014). *Software Engineering: A Practitioner's Approach*. McGraw Hill; 8th edition.

Rebello, G. A. F., d. Reis, M. L., Damasceno, R. G., d. Souza, R. O. S. e d. Jesus, R. C. R. (2024). Sistemas de detecção de intrusão, conceituação. https://www.gta.ufrj.br/grad/16$_2$/2016$IDS/conceituacao.html. Acessadoem$12/11/2024.

Ribacionka, T. M. S. M., Machado, I. L. O., Souza, A. V. A. P., Labarrere, D. S. e Chagas, M. C. (2022). LGPD na saúde. `https://oabdf.org.br/wp-content/uploads/2022/01/Cartilha-LGPD-na-Saude.pdf`. Acessado em 11/11/2024.

RNP (2024). Sistema rnp. `https://www.rnp.br/sistema-rnp/`. Acessado em 18/11/2024.

Significados (2024). Segurança. https://www.significados.com.br/seguranca/. Acessado em 12/11/2024.

Sulaman, S. M. (2011). An analysis and comparison of the security features of firewalls and idss. Dissertação de Mestrado, Institutionen för Systemteknik - Department of Electrical Engineering, Linköping, Sweden.

Tenhunen, T. (2008). Implementing an intrusion detection system in the mysea architecture. Dissertação de Mestrado, Naval Postgraduate School, Monterey - California.

Uma, M. e Padmavathi, G. (2013). A survey on various cyber attacks and their classification. *International Journal of Network Securit*, 15(5):390–396.

Vaz, G. M., Rizzetti, T. A. e Filho, W. P. (2021). Um estudo de caso sobre a implantação de um ambiente de prevenção de intrusões com a ferramenta suricata. *SBC*.

Wiki Archlinux (2024a). iptables. `https://wiki.archlinux.org/title/Iptables`. Acessado em 23/11/2024.

Wiki Archlinux (2024b). iptables. `https://wiki.archlinux.org/title/Rsyslog`. Acessado em 23/11/2024.