MONOGRAFIA DE PROJETO FINAL DE GRADUAÇÃO

IMPLEMENTAÇÃO E ANÁLISE DE RESULTADOS DE FERRAMENTA DE DETECÇÃO E RESPOSTA PARA PROTEÇÃO DE ENDPOINTS EM AMBIENTE CONTROLADO

Ana Paula de Aguiar Alarcão

Curso Superior de Engenharia de Redes de Comunicação

DEPARTAMENTO DE ENGENHARIA ELÉTRICA
FACULDADE DE TECNOLOGIA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

MONOGRAFIA DE PROJETO FINAL DE GRADUAÇÃO

IMPLEMENTAÇÃO E ANÁLISE DE RESULTADOS DE FERRAMENTA DE DETECÇÃO E RESPOSTA PARA PROTEÇÃO DE ENDPOINTS EM AMBIENTE CONTROLADO

Ana Paula de Aguiar Alarcão

Monografia de Projeto Final de Graduação submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Bacharel em Engenharia de Redes de Comunicação

Banca Examinadora

Prof. Dr. Georges Daniel Amvame Nze, FT/UnB _____
Orientador

Prof. Dr. Robson de Oliveira Albuquerque, FT/UnB _____
Co-Orientador

Prof. Dr. Fábio Lúcio Lopes de Mendonça, FT/UnB _____
Examinador Interno

FICHA CATALOGRÁFICA

ALARCÃO, A.P.A.
IMPLEMENTAÇÃO E ANÁLISE DE RESULTADOS DE FERRAMENTA DE DETECÇÃO E RESPOSTA PARA PROTEÇÃO DE ENDPOINTS EM AMBIENTE CONTROLADO [Distrito Federal] 2021.
xvi, 63 p., 210 x 297 mm (ENE/FT/UnB, Bacharel, Engenharia de Redes de Comunicação, 2021).
Monografia de Projeto Final de Graduação - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Proteção de Endpoints	2. Detecção de intrusão
3. Ataques cibernéticos	4. Gestão de incidentes
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

ALARCÃO, A.P.A. (2021). *IMPLEMENTAÇÃO E ANÁLISE DE RESULTADOS DE FERRAMENTA DE DETECÇÃO E RESPOSTA PARA PROTEÇÃO DE ENDPOINTS EM AMBIENTE CONTROLADO*. Monografia de Projeto Final de Graduação, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 63 p.

CESSÃO DE DIREITOS

AUTORES: Ana Paula de Aguiar Alarcão e TÍTULO: IMPLEMENTAÇÃO E ANÁLISE DE RESULTADOS DE FERRAMENTA DE DETECÇÃO E RESPOSTA PARA PROTEÇÃO DE ENDPOINTS EM AMBIENTE CONTROLADO.
GRAU: Bacharel em Engenharia de Redes de Comunicação ANO: 2021

Ana Paula de Aguiar Alarcão
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP: 70919-970 - Brasília-DF - Brasil

AGRADECIMENTOS

Agradeço primeiramente à minha mãe, Raquel, por todo o amor, apoio, excelente educação e, sobretudo, por ter sido meu primeiro grande exemplo de mulher forte. Agradeço ao meu avô, Eneas, pelo amor, incentivo e incontáveis bons conselhos. À minha avó, Brigida e ao meu irmão Rafael, por tanta ternura, segurança e por sempre vibrarem tanto com as minhas conquistas. Obrigada por tudo, família!

Agradeço aos meus orientadores, professor Georges Daniel Amvame Nze e professor Robson de Oliveira Albuquerque, pelas ideias, ensinamentos e toda a paciência ao longo deste projeto.

Agradeço aos meus colegas de curso, colegas de profissão, e amigos queridos, que me proporcionaram ótimas lembranças desta jornada, me ajudaram em momentos difíceis e sem os quais eu não teria chegado até aqui.

RESUMO

Nos últimos anos, pode-se observar um aumento no número de ataques cibernéticos, e por consequência, a quantidade de empresas que tiveram seus ambientes e sistemas comprometidos. Isso se deve ao número extenso de vulnerabilidades presentes nas aplicações atuais, mas também na complexidade crescente que os ataques tem ganhado ultimamente. Assim, muitas vezes, uma exploração inicial passa indetectável para administradores de TI por falta de ferramentas adequadas e bem configuradas de detecção de intrusão. Portanto, é crucial ter um ambiente preparado que proporcione a identificação dessas ameaças o mais rápido possível e com análises que sejam úteis para mitigação e remediação de falhas de segurança identificadas. Então, neste projeto, propõe-se analisar resultados obtidos a partir de ataques externos direcionados a um ambiente controlado implementado com o auxílio uma solução de detecção e reposta para *endpoints*. Utilizou-se como metodologia a de experimento-teste para desenvolvimento de testes de intrusão e, como solução do ambiente de detecção, foi implementado o EDR Wazuh e a pilha Elastic para visualizar eventos de segurança. Os resultados expõem como a ferramenta reagiu a comportamentos maliciosos e exploram a correlação entre logs e o modelo consolidado de defesa proposto pelo Mitre ATT&CK.

Palavras-chave: Proteção de Endpoints, Ataques cibernéticos, Detecção de intrusão, Gestão de incidentes, Mitigação e remediação de ataques.

ABSTRACT

From the last years, it can be observed an increased number of cyber attacks, and consequently, a gorwth in the number of companies that had their environments and systems compromised. This is due to the extensive number of vulnerabilities present in current applications, but also to the increasing complexity that attacks have gained lately. Thus, an initial exploit is often undetectable to IT administrators for lack of proper and well-configured intrusion detection tools. Therefore, it is crucial to have a prepared environment that provides the identification of these threats as quickly as possible and with analysis that are useful for mitigation and remediation of identified security flaws. So, for this project, we propose to analyze results obtained from external attacks targeting a controlled environment implemented with the aid of an EDR, a detection and response solution for *endpoint* protection. A test-experiment methodology was used to develop intrusion tests and, as a solution for the detection environment, Wazuh EDR and Elastic stack were implemented to visualize the security events. The results present how the tool reacted to malicious behavior and explore the correlation between logs and the consolidated defense model proposed by Mitre ATT&CK.

Key words: Endpoint Protection, Cyberattacks, Intrusion Detection, Incident Management, Attack mitigation and remediation.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SÍMBOLOS

Siglas

API	*Application Programming Interface*
APT	*Adavanced Persistant Threat*
AWS	*Amazon Web Services*
CPU	*Central Processing Unit*
CVE	*Common Vulnerabilities and Exposure*
CIA	*Confidentiality, Integrity and Availability*
CIDR	*Classless Inter-Domain Routing*
DNS	*Domain Name System*
DoS	*Denial of Service*
EBS	*Elastic Block Storage*
EC2	*Elastic Cloud Computing*
EDR	*Endpoint Detection and Response*
ELK	Elasticsearch, Logstash e Kibana
GDPR	*General Data Protection Regulation*
GPG	*GNU Privacy Guard*
HIDS	*Host-based Intrusion Detection System*
HIPAA	*Health Insurance Portability and Accountability Act*
HTTP	*Hypertext transfer protocol*
HTTPS	*Hypertext transfer protocol* sobre TLS
IaaS	*Infrastructure as a Service*
IDS	*Intrusion Detection System*
IoT	*Internet of Things*
IPS	*Intrusion Protection System*
IP	*Internet Protocol*
JSON	*JavaScript Object Notation*
MAC	*Medium Access Control*
NAT	*Network Address Translation*
NIDS	*Network Intrusion Detection System*
PaaS	*Plataform as a Service*
PCI DSS	*Payment Card Industry Data Security Standard*
PEM	*Privacy Enhanced Mail*
PKCS	*Public Key Cryptography Standards*
PKI	*Public Key Infrastructure*
RAM	*Random Access Memory*
RAT	*Remote Access Tool*
RDP	*Remote Desktop Protocol*

RFC	*Request For Comment*
SaaS	*Software as a Service*
SIEM	*Security Information and Event Manager*
SO	Sistema Operacional
SSH	*Secure Shell*
SSL	*Secure Socket Layer*
TCP	*Transport Control Protocol*
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
VM	*Virtual machine*
vCPU	*Virtual Central Processing Unit*
VPC	*Virtual Private Network*
VPN	*Virtual Private Network*

1 INTRODUÇÃO

De acordo com estatísticas do CERT.br [5] (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil), desde 2014 há um crescente quantidade de notificações de incidentes de segurança no Brasil. Em 2014 ocorreu o maior número de notificações até hoje, cerca de 1 milhão de incidentes de segurança foram reportados. Apenas em 2020, foram notificados cerca de 665.000 incidentes de segurança. Dentre os tipos de ataques estão atividades maliciosas de *worms*, DoS (*Denial of Service*), invasão, ataques web, scans, fraudes e outros.

O avanço e facilidade de conexões realizadas através das redes e da Internet tornou este ambiente propício para agentes maliciosos, intrusões e aplicativos vulneráveis disponíveis aos consumidores. Assim, o que se tem hoje em dia é uma corrida entre as novas vulnerabilidades e as ferramentas de segurança tentando proteger ativos e ambientes para garantir as características fundamentais de integridade, confidencialidade e disponibilidade.

Além disso, existe o problema de ameaças persistentes, ou APTs (*Advanced Persistent Threats*), que são aquelas que ficam indetectáveis em um sistema durante longos períodos até atingir o objetivo final. É extremamente difícil implementar proteções contra essas ameaças, apenas ferramentas que fazem correlação de logs de origens diversas tem alguma chance em detectar esse tipo de ataque.

Tipicamente hoje em dia se utiliza ferramentas de segurança em profundidade, como por exemplo *firewalls* de rede, antivírus e *firewalls* de aplicação. Esse tipo de ferramenta de segurança da informação é também chamado de proteção em camadas, pois são várias as que necessitariam ser violadas para se ter um ataque bem sucedido. Porém, uma vez que o ataque ocorre e já há um agente malicioso dentro do ambiente, essas ferramentas não rastreiam o movimento desse atacante e nem previnem o deslocamento lateral do intruso ou *malware* pela rede.

O movimento lateral é a tática que diferencia as ameaças avançadas dos dias atuais de ataques cibernéticos simples do passado, pois possibilita a um atacante que ele mantenha o acesso ao ambiente comprometido e vá aumentando os privilégios que possui naquele sistema. Assim, mesmo que a máquina inicialmente comprometida seja descoberta, o atacante já estará imerso na rede à procura de dados sensíveis ou outro ativo que seja valioso para exploração.

Portanto, o movimento lateral malicioso é de difícil detecção pois se confunde com o trafego normal de usuário, ainda mais quando são utilizadas ferramentas que já estão instaladas no ambiente. O essencial é investigar e remover estes invasores o mais rápido possível para minimizar maiores perdas.

O período médio de *breakout*, demonstrado na figura 1.1, que é o tempo que um invasor leva para se movimentar lateralmente dentro da rede assim que compromete uma máquina, é de 1 hora e 58 minutos, segundo relatório de 2020 da fabricante CrowdStrike [1]. Assim, é necessário diminuir ou até mesmo eliminar esta janela em que o acante consegue se mover lateralmente e, para isso, é essencial utilizar soluções de segurança de *endpoints* modernos e possuam funcionalidades para análise de comportamento e correlação de eventos para atividades suspeitas.

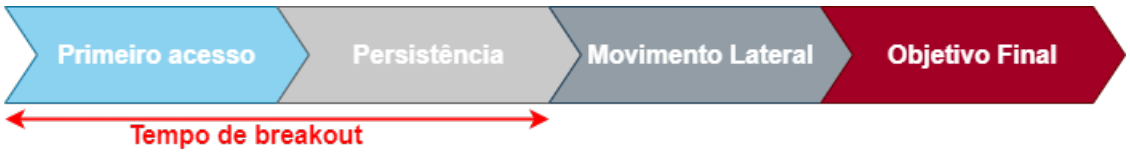

Figura 1.1: Tempo de *breakout* de 1h e 58 minutos, período entre o primeiro acesso e o início do movimento lateral que pode ser executado pelo atacante. Traduzido de CrowdStrike [1].

Ferramentas de EDR (*Endpoint Detection and Response*) que realizam a segurança da rede correlacionando informações de máquinas de todo o perímetro, previnem tentativas de intrusão e detectam comportamentos suspeitos são soluções de segurança que dão essa visibilidade em relação a ataques complexos. Com EDR, eventos que ocorrem em apenas um *host* são coletados, processados e correlacionados a outros eventos.

1.1 DEFINIÇÃO DO PROBLEMA

Hoje em dia, qualquer infraestrutura que possui comunicação com a internet e com o mundo exterior está suscetível a um ataque. Ameaças e ataques estão cada vez mais silenciosos e, consequentemente, aumenta-se a complexidade de detecção de algum desses eventos. Daí parte uma preocupação de profissionais de segurança da informação se caso ocorra um incidente de segurança, eles serão capazes de detectar a ocorrência do incidente e de forma rápida.

Logo, é preciso garantir a segurança desse ambiente com ferramentas que agreguem valor ao trabalho do analista de segurança que irá monitorar o ambiente. Quanto mais baixo nível essa ferramenta puder monitorar os sistemas de uma empresa e mais correlacionado estejam os dados obtidos na forma de logs, mais chances um incidente de segurança tem de ser detectado rapidamente.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é criar um ambiente controlado para detecção de ataques e intrusões em *endpoints* com base nos resultados providos por uma ferramenta de segurança da informação.

1.2.2 Objetivos Específicos

Alguns objetivos detalhados são propostos afim de se atingir o objetivo principal de implementar e detectar ataques com a ferramenta de EDR Wazuh:

- Arquitetar um ambiente de teste em nuvem com todas as ferramentas de segurança à que se aplicará a prova de conceito. Princípios de boas práticas de segurança devem ser aplicados à todas às fases do projeto.

- Realizar tentativas de penetração e intrusão no ambiente segundo a *Cyberkill chain*, mencionada na seção 3.2.
- Configurar de forma apurada o software de EDR para detecção dos ataques propostos
- Defender o ambiente controlado segundo principais técnicas Mitre ATT&CK, mencionado na seção 3.1.
- Analisar o desempenho do EDR em detectar ataques no ambiente controlado.

1.3 ESTRUTURA DO DOCUMENTO

Este trabalho está dividido entre 6 capítulos. O primeiro deles é a introdução, cujo principal objetivo é descrever as principais motivações e objetivos do trabalho. O capítulo 2 tratá sobre conceitos e base teórica para os assuntos que o projeto explorará. O capítulo 3 descreve as ferramentas que foram utilizadas e características de cada serviço bem como suas funcionalidades principais e como foram configuradas. Já o capítulo 4 apresenta como o trabalho foi construído através da metodologia utilizada e das topologias lógicas e físicas da solução proposta. Ainda, no capítulo 5, os testes e resultados serão apresentados com análise e discussão sobre cada um. Por fim, o capítulo 6 trará uma parecer final sobre o problema abordado, a solução, os resultados e aspectos que podem ser explorados ainda em um desenvolvimento futuro.

2 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo visa explicar a teoria por trás de outras tecnologias abordadas neste projeto de forma a favorecer seu entendimento. Sendo assim, realizaremos uma revisão de conceitos e fundamentos teóricos necessários durante a execução da solução proposta.

2.1 ELEMENTOS DE SEGURANÇA DA INFORMAÇÃO

Conforme Kemmerer [6], a capacidade de se garantir uma segurança cibernética eficaz, consiste, em grande parte, na aplicação de métodos defensivos utilizados para detectar e frustrar os possíveis intrusos. E ainda mais, segundo Blackley et al. [7], o propósito de se proteger informações é proteger os recurso mais valiosos de uma organização, como hardware, software, dados.

Portanto, de maneira mais ampla, a segurança da informação age desde à criação de uma cultura de segurança guiada por documentos e políticas que contenham diretrizes, até desenvolver um código com boas práticas de segurança.

A segurança da informação caminha junto a objetivos de negócios, pois atua diretamente na prevenção de ataques que poderiam negativar a reputação do negócio. Ou então com práticas de continuidade de negócios, com criação de backups, e dessa forma o negócio teria menos chances de ser comprometido completamente em caso de desastre. Enfim, o papel da segurança da informação é proteger as informações e recursos de um ambiente através de ações contínuas, ferramentas e, principalmente, conscientização.

Portanto, a segurança da informação é uma área composta por princípios, tal qual os da tríade CIA, quanto de protocolos seguros, tanto de soluções e sistemas que são utilizados como ferramentas de infra-estrutura. Estes todos são elementos de segurança da informação.

2.1.1 CIA: tripé da segurança da informação

A tríade CIA de segurança da informação consiste nos princípios de confidencialidade, integridade e disponibilidade (em inglês *availability*, por isso o A em CIA) e é um modelo globalmente reconhecido e designado a guiar diretrizes e trabalhos de segurança da informação em vários âmbitos. Estes três são os fundamentos mínimos de segurança sobre o qual sistemas de informação devem ser arquitetados, segundo Yeboah-Boateng [8].

Em cada ataque, vulnerabilidade ou ameaça que surge, é sempre possível identificar qual ou quais desses três princípios foram violados. Além disso, a confidencialidade, integridade de disponibilidade são comumente utilizados por equipes de segurança da informação para implementar controles no ambiente em busca de minimizar riscos associados a ele.

Cabe aqui apresentar uma breve noção de cada conceito e como podem ser violados:

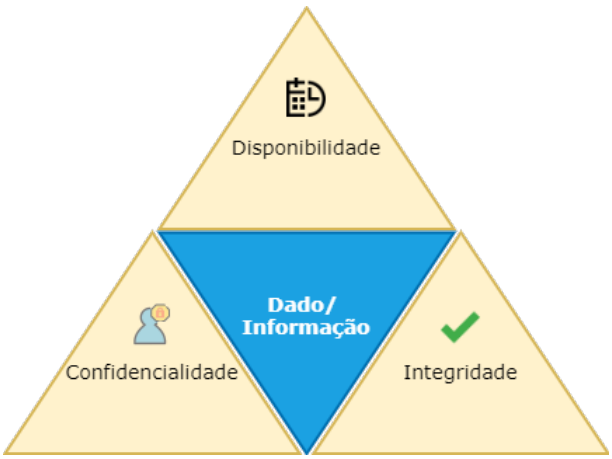

Figura 2.1: Tríade de fundamentos de segurança da informação. Fonte: o autor.

- Confidencialidade: o ato de se manter qualquer informação ou dado protegido de acesso não autorizado. Exemplos de violações incluem o compartilhamento de credenciais de acesso a sistemas diversos, ataques que realizam reconhecimento da rede através de escaneamento de portas e serviços, ataques *man-in-the-middle*. Medidas para manter a confidencialidade incluem classificar dados e documentos, criptografar o tráfego ou mesmo educar indivíduos em relação à proteção de credenciais.

- Integridade: consiste em se manter um objeto inalterado e autêntico de forma que ele permaneça confiável. Exemplos de violações podem ser a modificação de logs para mascarar uma ação suspeita, modificar arquivos de configuração de sistemas ou erros e alterações no código de uma aplicação. Algumas medidas para proteger a integridade podem ser utilizando certificados digitais, utilizar sistemas de detecção de intrusão, auditorias ou versionamento de código.

- Disponibilidade: o ato de manter recursos acessíveis de forma confiável, consistente e estável para usuários autorizados. Exemplos de violação à esse princípio incluem falhas de software e hardware, falhas naturais, ataques de negação de serviço. Para manter a disponibilidade, algumas medidas são possuir redundância em todos os recursos, possuir backups e sistemas de detecção e proteção contra negação de serviço.

2.1.2 Segurança de *Endpoints*

Pode-se dizer que um *endpoint* é o ponto de extremidade da rede, segundo Wikipedia [9], e são utilizados como ponto de acesso à uma aplicação, à Internet como um todo ou até a outro *endpoint*, podendo ser um notebook, um *smartphone*, um computador e mais recentemente, dispositivos de Internet das Coisas (*Internet of Things* - IoT).

Até recentemente, o conceito de segurança para esses pontos de extremidade estavam limitados a instalação de *softwares* de antivírus, segundo Yoo [10]. Porém, com o aumento da complexidade dos ataques e com a difusão de dispositivos IoT, é preciso utilizar serviços que trocam dados e se integram em vez dos que funcionam individualmente, ainda conforme Yoo [10].

Assim, a segurança de *endpoint* atualmente propõe ferramentas e soluções que concentrem funcionalidades de controle de aplicações, controle de portas, inteligência de ameaças, monitoramento de arquivos, monitoramento do sistema, controle de prevenção de dados, tudo isso em um mesmo agente instalado no dispositivo. Dessa forma, o *endpoint* funciona como uma rica fonte de medidas e eventos de segurança dispersos na rede como uma estratégia de se obter visibilidade sobre o tráfego do ambiente em que o *endpoint* se encontra.

Os EDRs combinam elementos de soluções de gerenciamento de antivírus e *endpoint* para detectar, investigar e remover qualquer software malicioso que penetre os dispositivos de uma rede. As ferramentas EDR dão maior visibilidade à saúde geral de um sistema, incluindo o estado de cada dispositivo específico.

2.1.3 SIEM (*Security and Information Event Management*)

Conforme Bhatt et al. [11], os sistemas SIEM foram projetados para coletar eventos de diversas fontes onde cada uma delas pode representar um esquema específico de fornecedor. O sistema normaliza esses esquema diferentes em uma representação comum e os armazena. Seu banco de dados regulatório (motor de regras) dispara alertas dos eventos armazenados e permite a correlação desses eventos a diferentes sensores.

Ainda conforme Bhatt et al. [11], a principal força dos sistemas SIEM é sua capacidade de cruzar registros de diversas fontes usando atributos comuns para definir padrões e cenários de ataque significativos, que quando ocorrem, podem alertar os analistas de segurança, sendo como um radar que detecta objetos em tempo hábil.

O log é a peça principal em sistemas de SIEM. Anastov e Davcev [12] definem log como um registro de eventos que ocorrem em sistemas e redes, sendo composto por entradas que proveem informações acerca de um evento. Sendo assim logs de segurança podem ser gerados pelas mais variadas fontes como softwares de antivírus, firewalls, soluções de IDSs/IPSs, logs de sistema operacional e de equipamentos de redes.

A capacidade de retenção de logs em sistemas de SIEM de longo prazo é útil para análise forense pós-hoc, bem como investigar e detectar ataques lentos e furtivos, incluindo ameaças persistentes avançadas (APTs).

Neste projeto, mostraremos como os logs apoiam na investigação de incidentes.

2.1.4 Sistema de Detecção de Intrusão (IDS-*Intrusion Detection System*)

Um sistema de detecção de intrusão (IDS) é uma solução de rede que inspeciona o tráfego buscando por atividades maliciosas, podendo ser por monitoramento do tráfego individual de uma máquina, que é chamado de IDS baseado em *host* (HIDS- *Intrusion Detection System*), ou monitoramento do tráfego de rede, que é conhecido como IDS baseado na rede (NIDS - *Intrusion Detection System*).

O IDS automatiza a tarefa de analisar dados de auditoria na identificação de atividades suspeitas, pois podem ser usados para estabelecer a culpabilidade do atacante e na maioria das vezes é o único modo de descobrir uma atividade sem autorização, detectar a extensão dos danos e prevenir tal ataque no futuro, tor-

nando desta forma o IDS uma ferramenta valiosa para análises em tempo real e também após a ocorrência de um ataque, segundo Laureano et al. [13].

Há muitas soluções de IDS disponíveis no mercado, muitas vezes já embutidas em soluções de *firewall*, sendo a maioria baseada em assinaturas. Logo, o que os IDSs fazem é monitorar pacotes que entram e saem de uma interface comparando com padrões de ataques conhecidos. Quando uma nova ameaça é descoberta, esse padrão é associado à uma nova assinatura dentro de uma biblioteca. A biblioteca geralmente está dividida por protocolo ou por serviço e possui várias assinaturas referentes à quais campos ou quais comportamentos devem ser inspecionados pois já foram considerados suspeitos no passado.

Usos comuns de sistemas de detecção de intrusão são em conjunto com soluções de análise de logs para implementação de ferramentas de SIEM, conforme mencionou-se na subseção 2.1.3 para combinar entradas de múltiplas entradas e distinguir atividade maliciosa e de falsos positivos mais facilmente. Outro uso específico é juntamente com *honeypots*, que são sistemas propositalmente vulneráveis do ponto de vista do invasor e funcionam como armadilha para ele, justamente para atrair a ameaça e caracterizar nossas assinaturas que representarão esse tráfego malicioso.

Algumas soluções *open source* conhecidas no mercado que fazem função de IDS são Snort, Suricata e Zeek. Cada um possui seu princípio de inspeção de pacotes e funcionamento próprio para processamento de regras, porém os três são soluções que se integram facilmente à outras, por exemplo à pilha Elastic Stack.

2.1.5 SSL/TLS (*Secure Socket Layer/Transport Layer Security*)

O protocolo SSL surgiu da preocupação em se proteger o protocolo HTTP, porém na realidade o SSL protege conexões TCP. Este é um protocolo da camada de transporte que aprimora o TCP com serviços de segurança tal como criptografia, integridade e autenticação do cliente final, segundo Kurose e Ross [14]. Porém, a versão 3.0 do protocolo foi denominada *Transport Layer Security* 1.0, portanto iremos nos referir daqui pra frente ao protocolo da forma correta, isto é, como TLS.

O TLS trouxe implementações de segurança que são usadas para proteger a conexão entre o cliente e o servidor web até hoje, são elas:

- Autenticação com assinatura digital e certificado X.509, tanto do lado do servidor como o cliente de forma opcional.
- Criptografia da sessão estabelecida entre cliente e servidor.
- Mantendo a integridade dos dados trocados através da inserção de uma chave MAC durante a troca de chaves na apresentação inicial.

Neste projeto, fez-se extenso uso de certificados digitais para comunicação segura entre os servidores envolvidos para garantir criptografia e autenticação. Dessa forma, um agente malicioso não poderia fazer parte do cluster ElasticStack implementado e encaminhar pacotes infectados. Utilizando TLS mutuamente nos servidores, garantimos que os nós usarão os certificados para identificação ao se comunicarem com outros nós, tudo isso utilizando o pacote elasticsearch-certutil para geração de certificados.

Usando como base os fundamentos descritos por Leavitt [15], os passos envolvidos da geração e utilização de certificados a partir da infraestrutura de chaves públicas, PKI (*public key infrastructure)* são:

1. A entidade interessada solicita à autoridade certificadora (CA) um certificado digital. Com a identidade verificada, a CA emite o certificado, que geralmente possui algumas informações sobre quem o solicitou, a chave privada fornecida pela CA e sua assinatura digital em forma de arquivos .key e .crt, respectivamente.
2. O solicitante instala os arquivos referentes ao certificado nos servidores desejados.
3. A partir daí, um cliente (geralmente, um *browser* ou outro servidor) solicita iniciar uma comunicação e também o certificado digital do servidor.

2.1.6 Vulnerabilidade e ameaça

Sherman et. al [16] definem vulnerabilidade como uma fraqueza que pode levar a prejudicar ou comprometer um sistema. Já uma ameaça, ainda segundo Sherman et. al, seria uma potencial ação ou condição que pode causar prejuízo, podendo estar relacionada a uma ou mais vulnerabilidades.

Sendo assim, podemos estabelecer que qualquer componente de um sistema, o sistema em si, ou seus processos, possuem vulnerabilidades. Ou seja, nenhum sistema está isento de vulnerabilidade. A diferença é que algumas vulnerabilidades são conhecidas e outras ainda não foram descobertas. Ademais, algumas podem ser de baixa complexidade e fácil exploração, e podem ser exploradas por agentes pouco qualificados. Já outras podem ser de difícil exploração e alta complexidade, apenas podendo ser exploradas por um agente altamente capacitado.

Outro aspecto que interfere na exploração de uma falha de segurança é a motivação. Atacantes podem ser agrupados pelos seus objetivos, motivação e capacidades, sendo que 4 grupos são de grande importância: os ciberterrorista, hacktivistas, agentes patrocinados pelo Estado e cibercriminosos, segundo Ablon [17]. Portanto, diferentes atacantes possuem diferentes motivações para atacar uma entidade, podemos ver na figura 2.2.

2.2 COMPUTAÇÃO EM NUVEM

Jadeja e Modi [18] definem "nuvem"a partir da utilização de redes privadas virtuais (VPN - *Virtual Private Network*) de provedores para serviços de comunicação de dados. Ainda citando Jadeja e Modi [18], a computação em nuvem lida com serviços de computação, armazenamento, processamento e serviços que não requerem que o cliente final conheça a localização física dos dispositivos que os implementam e nem a configuração dos sistemas que entregam os serviços.

Como o objetivo principal da nuvem é otimizar o uso dos recursos distribuídos os usuários finais podem usufruir de algumas características chave da computação em nuvem, são elas:

- Acesso via qualquer dispositivo que suporte o uso de navegadores e tenha conexão com a internet.

Figura 2.2: Agentes maliciosos e suas respectivas motivações em praticar ataques. Traduzido de Canadian Cyber threat and Cyber Threat Actors. [2]

- Como o nível de abstração de serviços e suas configurações é muito alta, usuário com pouco conhecimento de tecnologia da informação são capazes de fazer o uso das plataformas.
- Os recursos contratados possuem alta confiabilidade, pois geralmente o próprio provedor empregam redundância dos serviços como premissa básica de implementação.
- Compartilhamento dos custos entre usuários através do compartilhamento da infraestrutura que pertencente ao provedor.
- Escalabilidade dos serviços através do pagamento sob demanda e alta disponibilidade dos recursos.

Alguns fabricantes do meio comercial investiram no passado e passaram a implementar e oferecer a computação em nuvem, como por exemplo a Amazon em 2006 que lançou a *Amazon Web Services* (AWS), a Google com a Google Cloud Platform lançada em 2008 e o lançamento da Microsoft da nuvem Azure, em 2010.

Existe também uma definição comercial de computação na nuvem, criada em 2011 pelo Instituto de padronizações e tecnologias dos Estados Unidos (*National Institute of Standarts and Technology* - NIST) na intenção de estabelecer uma base conceitual na qual a computação em nuvem pudesse ser regulamentada, comercializada, fiscalizada, entre outros. Além de características essenciais, Mell e Grance [19] expõe modelos de serviço e modelos de implantação de computação em nuvem, sendo eles:

- **Infraestrutura como serviço (*Infrastructure as a Service* - IaaS)**: o modelo de infraestrutura é o que permite um maior controle por parte do usuário, como observa-se na comparação da Figura 2.3. Com esse modelo de serviço, é garantido ao usuário o privilégio de provisionar recursos virtualizados

de computação, armazenamento e de rede. Por conta do alto nível de controle da infraestrutura, esse modelo de serviço é mais adequado a administradores de TI, que serão responsáveis por implantar os mais diversos ambientes para o restante de sua organização

- **Plataforma como serviço (*Plataform as a Service* - PaaS)**: nesse modelo de serviço, é fornecido ao usuário o poder de implantar na nuvem aplicações previamente disponibilizadas pelo provedor. O usuário, portanto, não controla a infraestrutura por trás das aplicações disponibilizadas, e sim as configurações da aplicação que servirá como plataforma para ele. Assim, modelo PaaS pode oferecer ambientes de desenvolvimento (*Integrated Development Environment* - IDE), com suporte a bibliotecas e linguagens de programação para o desenvolvimento das aplicações.

- **Software como serviço (*Software as a Service* - SaaS**: : No mais alto nível do diagrama da figura 2.3 encontra-se o SaaS. Nesse modelo de serviço, é provido ao usuário a capacidade de utilização de *softwares* rodando em uma infraestrutura de nuvem. A utilização desse software pode ser feita através de uma interface web em um *browser*, através da interface de outra aplicação via chamadas API (*Application programming Interface*). O usuário não possui controle da infraestrutura por trás da aplicação, como acontece, em diferentes graus, nos modelos de serviço IaaS e PaaS.

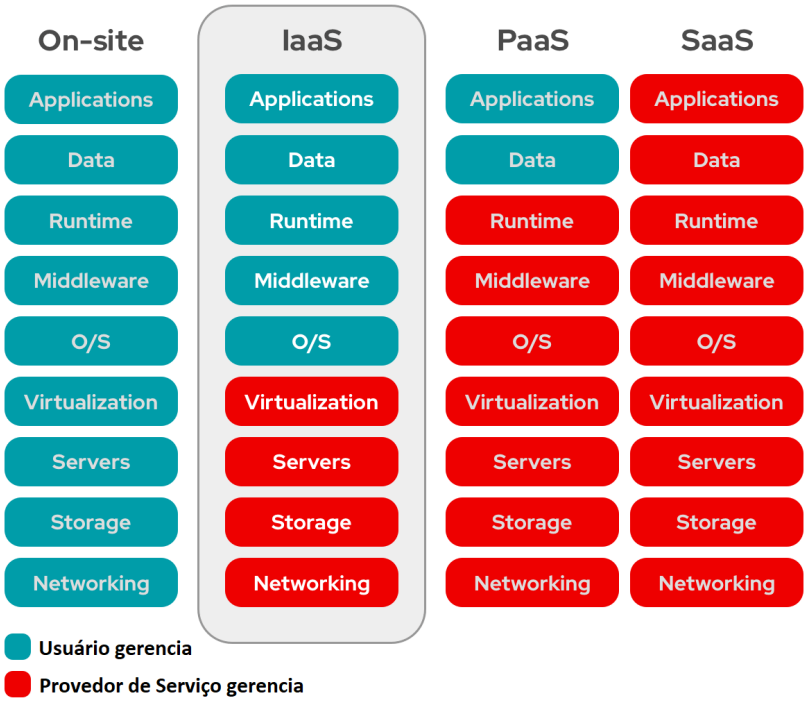

Figura 2.3: Modelos de Serviço providos pela computação em nuvem em comparação com o modelo tradicional de Datacenters *on-premises*. Em verde estão os serviços gerenciado pelo usuário e em vermelho, os serviços gerenciados pelo provedor de nuvem. Fonte: Red Hat [3].

2.2.1 Nuvem AWS

A Amazon Web Services - AWS é o provedor de serviços de nuvem criado pela Amazon e combina os modelos de serviço IaaS, PaaS e Saas desde 2006, de acordo com Wikipedia [20], com diversos *datacenters*

espalhados por regiões em todos os continentes, onde cada região possui algumas zonas de disponibilidade que representam os *datacenters* físicos da AWS. Ela possui atualmente mais de 200 serviços, entre eles, banco de dados, gerenciamento de infraestrutura, desenvolvimento de aplicações, segurança da informação, armazenamento, computação e outros.

A AWS disponibiliza serviços a nível global independente de onde se encontra o usuário, basta que ele possua um navegador com conexão à internet para acessar sua conta, que é basicamente um contêiner para todos os recursos que serão utilizados. É também através da conta que o usuário tem a gerência e acesso ao faturamento de serviços, conforme cita Ali [21]. Alguns serviços providos pela AWS serão utilizados nesse projeto e estão contidos em uma conta dedicada à ele.

A Amazon *Virtual Private Cloud* - VPC (Rede Privada em Nuvem) é o serviço que define uma versão virtual de redes físicas mas dentro da infraestrutura da AWS. A VPC é logicamente isolada na nuvem AWS e portanto é possível associar um intervalo de endereço IP a ela e às sub redes contidas nela, configurar as tabelas de roteamento, tal qual um *datacenter* tradicional.

O Amazon EC2 (*Elastic Compute Cloud* - Computação Elástica em Nuvem) é o serviço que provê, através da virtualização, servidores virtuais que são hospedados pela AWS. O usuário pode usufruir da capacidade de computação e processamento de máquinas virtuais por demanda, que são chamadas de instâncias. As instâncias possuem configurações configuráveis de quantidade de CPU, memória, armazenamento, recursos de rede e até mesmo do sistema operacional. Assim, este é um exemplo do modelo de serviço de *IaaS* da seção 2.2, onde o usuário pode requisitar um servidor virtual em poucos minutos sem ter que adquirir um *hardware* de servidor físico robusto e de alto custo inicial.

O Amazon EBS (*Elastic Block Store* - Armazenamento Elástico em Bloco) é o serviço de armazenamento em blocos para o serviço de EC2 mencionando no último parágrafo e é utilizado para armazenar dados persistentes das máquinas virtuais. O EBS funciona como servidores tradicionais de armazenamento que oferece discos virtuais para máquinas virtuais utilizadas em datacenters tradicionais.

3 FERRAMENTAS UTILIZADAS

O capítulo a seguir visa descrever em detalhes as ferramentas selecionadas para esse projeto de código aberto disponíveis na comunidade de segurança da informação, e que podem ser empregas a nível empresarial. Portanto, traremos uma visão de como elas se encaixam no desenvolvimento da solução proposta com suas principais funcionalidades.

3.1 FRAMEWORK MITRE ATT&CK

Segundo Georgiadou et. al [22] projeto MITRE ATT&CK (*Adversarial Tactics, Techniques, and Common Knowledge*) surgiu em 2013 com foco em ambientes executivos formados predominantemente por Windows e a partir de 2017, começaram a focar em sistemas operacionais Linux e MacOS. Em 2019, foi publicado e incluído no modelo *enterprise* do MITRE ATT&CK a matriz ATT&CK for Cloud que descreve o comportamento de ataques em ambientes e serviços de nuvem.

A estrutura do MITRE ATT&CK descreve táticas e técnicas sobre como atacantes penetram redes, se movimentam dentro dela, escalam privilégios e violam ferramentas de segurança da informação. O MITRE ATT&CK busca pelos métodos utilizados pelos atacantes, as táticas e os objetivos a serem atingidos.

Algumas táticas conhecidas desse *framework* são Evasão de Defesas (*Defense Evasion*), Movimento Lateral (*Lateral Movement*) e Reconhecimento (*Reconnaissance*). Cada tática possui um conjunto de técnicas utilizadas para atingir um objetivo definido pelo atacante. Há muitas técnicas que podem ser utilizadas em cada tática, pois há inúmeras formas de se explorar um ambiente, só depende das habilidades do hacker, das ferramentas a que ele possui acesso e do ambiente que ele está lidando.

Para cada técnica, há também o método utilizado pelo atacante, o ambiente ou sistema operacional alvo e quais grupos conhecidos utilizam ou já utilizaram essa técnica para explorar alguma falha de segurança. Além disso, as técnicas também descrevem como mitigar aquele tipo de ataque.

O *framework* do MITRE ATT&CK também são utilizados por times de segurança da informação e ferramentas para auxiliar na detecção e identificação de comportamentos maliciosos, como é o caso do Wazuh utilizado nesse projeto. Um exemplo de análise seria levantar quais as táticas mais exploradas e daí implementar soluções e ferramentas visando mitigá-las.

A matriz de táticas e técnicas pode ser encontrada no site oficial da Organização Mitre ATT&CK [23].

3.2 MODELO *CYBER KILL CHAIN*

Segundo Yadav e Rao [4], o modelo *Cyber kill chain* é utilizado pela investigação forense digital, análises de *malware* e reposta a incidentes pois ele molda as etapas ofensivas tomadas por um agente

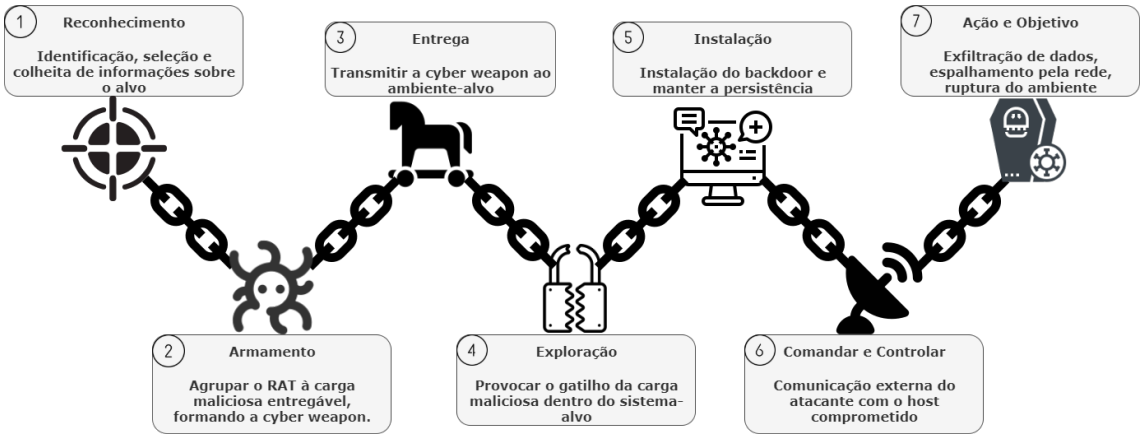

Figura 3.1: Etapas da *Cyber kill chain*. Traduzido para o português a partir de Yadav e Rao [4].

malicioso antes e durante um ataque.

O *framework Cyber kill chain* divide em etapas ataques que podem ser complexos se vistos como um todo para facilitar o trabalho da segurança da informação a olhar problemas menores e mitigar ou defender contra o ataque fase a fase.

Assim, o modelo *Cyber kill chain* completo engloba as seguintes fases: Reconhecimento, Armamento, Entrega, Exploração, Instalação, Comandar e Controlar, Ação e Objetivo, conforme figura 3.1. Contudo, não necessariamente a cadeia inteira precisa ser cumprida para atingir o objetivo final e algumas etapas podem ser puladas, dependendo somente de qual o objetivo final do acante. Então para frustrar o ataque, deve-se interromper algum dos passos envolvidos para que a cadeia falhe completamente.

Aqui será apresentado apenas uma breve explicação do que consiste cada fase. Começado pelo Reconhecimento, onde o atacante busca informações sobre o alvo, podendo ser de maneira passiva, quando é feito consultando fontes terceiras como o *Shodan* e o *Whois*, ou de forma ativa, quando o atacante sonda o ambiente-alvo por meio de *scans*, por exemplo. A busca ativa causa muito mais ruído na rede e por isso tende a ser mais facilmente detectado pelo alvo. Porém, o objetivo da fase de reconhecimento é encontrar uma brecha de segurança no ambiente-alvo.

A segunda fase é a de Armamento, onde se arquiteta o ataque para explorar a falha de segurança ou vulnerabilidade encontrada através de algumas ferramentas. Há muitas disponíveis no mercado, várias são bem conhecidas nessa fase, como *Metasploit* que inclusive foi utilizado nesse projeto, o Burpsuite, Exploit-DB e outras.

Depois, o atacante prepara a parte da Entrega do arquivo malicioso (*payload*) através de um ou múltiplos meios, tipicamente de forma a camuflar ou enganar o usuário fazendo-o crer que aquele é um conteúdo legítimo. Entre as formas de infectar um usuário estão sites comprometidos, redes sociais, e-mails falsos (phishing), USBs infectados. Apenas usuários bem treinados em relação às boas práticas de segurança da informação não caem em tentativas de intrusão que tentam enganá-los.

A quarta fase é a de Exploração. Se o ataque avança até essa fase, significa que o *payload* foi entregue à vítima com sucesso e o ataque foi executado. O ataque pode vir em forma de uma injeção SQL, onde comandos do interesse do atacante servem como entradas maliciosas para base de dados SQL, ou na forma

de *malwares*, *buffer overflow*, onde o atacante excede o uso de memória de um dispositivo fazendo-o falhar. Qualquer que seja a forma de exploração, o objetivo é obter acesso ao ambiente. Poucas ferramentas de segurança da informação são capazes de proteger sistemas nessa fase da *Cyber kill chain*.

Depois, o atacante tentar melhorar os níveis de acesso e permissão que ele possui ao sistema-alvo através da etapa de Instalação. Dessa forma, mesmo que o sistema seja atualizado ou reiniciado, o atacante ainda será capaz de controlá-lo. Nessa fase, também há poucas ferramentas de segurança para proteger o sistema, porém é possível utilizando soluções de *Endpoint Detection and Response - EDR* para monitorar o ambiente buscando mudanças de registros e arquivos, execução de comandos de PowerShell, por *Remote Access Tools - RATs* rodando no ambiente.

Já no fim da cadeia, temos a fase de Comando e Controle, onde o ambiente já está completamento comprometido e o atacante já tem o controle remoto sobre o sistema, basta um comando para dar início à última fase. Nessa fase, proteger o ambiente é uma tarefa complexa, porém há algumas alternativas: por exemplo, quanto mais segmentada uma rede, menor o estrago do ataque; além disso, alguns *firewalls* possuem funcionalidade de bloqueio de origens conhecidas que praticam Comando e Controle (C&C).

A última fase de Ação e Objetivo é onde o atacante executa ação buscando o objetivo primordial, que tipicamente envolvem motivos financeiros extorquindo a vítima, espionagem, ou simplesmente se movimentar lateralmente se alastrando pela rede.

Concluindo, a *Cyber kill chain* não é apenas uma sequência de passos para um atacante tomar a rede, é também um modelo para times de segurança da informação implementarem medidas de proteção, detecção e resposta, conforme implementaremos nesse projeto.

3.3 PILHA ELASTIC

A pilha Elastic é a antiga pilha ELK que implementa os serviços de três projetos *opensource* diferentes Elasticsearch, Logstash e Kibana. Anteriormente, cada letra do acrônimo implementava uma camada do projeto conforme a figura 3.2, mas agora em sua nova versão conta com o componente Beats. A pilha Elastic provê funcionalidades de agregação de logs, análise e visualizações para monitoramento de segurança e eventos de sistemas.

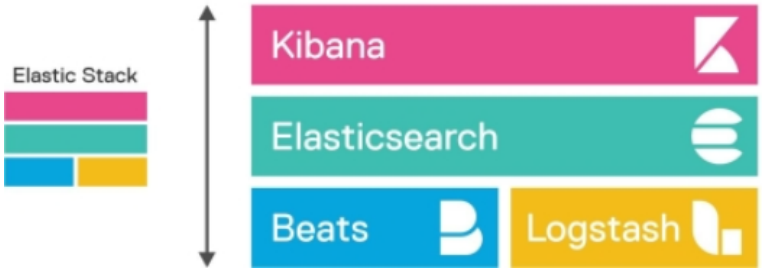

Figura 3.2: Estrutura da solução Elastic com os 3 produtos principais oferecidos Elasticsearch, Logstash e Kibana. Fonte: elastic.co

O primeiro nível da pilha é o Kibana, cuja principal função é implementar uma ferramenta de visuali-

zação e exploração de dados para análise de logs e eventos. Com ele é possível construir painéis, gráficos e filtros em uma interface web de usuários que possibilitam o monitoramento de eventos.

No segundo nível, tem-se o Elasticsearch, feito para ser um mecanismo de busca baseado em arquivos JSON para requisições e análises em tempo real. Todos os campos do JSON são rapidamente identificados e indexados para tornar todo e qualquer dado buscável na ferramenta. Dependendo do tamanho da infraestrutura, o Elasticsearch é facilmente escalável funcionando com uma arquitetura distribuída e vários nós comunicando entre si.

No terceiro nível dos serviços tem-se o Logstash e o Beats. O Logstash é o serviço do antigo ELK que pode ser utilizado opcionalmente para processar e enriquecer os dados coletados. Já o Beats realiza o encaminhamento de dados para o Logstash, caso necessite de enriquecimento, ou para o Elasticsearch diretamente para centralização.

O Beats possui três tipos principais de dados que encaminha. O primeiro é o Packetbeat, analizador de pacotes de rede e encaminhador de dados do tráfego de rede de dispositivos. O Metricbeat coleta métricas de sistema e de serviços, como uso de CPU e memória, estatísticas de leitura e escrita de disco, entrada e saída de interfaces de rede, e etc. Por último, tem-se o Filebeat, tipicamente usando para centralizar arquivos de logs das instâncias existentes em um ambiente, com sensibilidade para prever o sobrecarregamento do Elasticsearch no recebido de altos volumes de dados. Faremos o uso do Filebeat nesse projeto para encaminhar arquivos de logs das instâncias para correlação e análise no EDR Wazuh, que mencionaremos na seção 3.4.

3.4 EDR WAZUH

O Wazuh é uma solução de segurança da informação de *Enpoint Detection and Response* para monitoramento em tempo real, prevenção, detecção e resposta a ameaças a nível de *host* baseado em assinaturas e vulnerabilidades conhecidas e análise de comportamento anômalo, de acordo com a documentação da própria ferramenta Wazuh [24]. O Wazuh é uma plataforma para proteção de ambientes e são muitos os casos de uso à que ele se aplica e inclusos nos serviços gratuitos e *open source* providos. Os casos de uso em detalhes estão descritos nas sub subseções a seguir.

3.4.1 Detecção de Intrusão

Este é o módulo que realiza o monitoramento em tempo real de detecção de ameaças de tipos variados como infecção por *malware*, *rootkits*, ferramentas de acesso remoto, execução de comandos suspeitos, arquivos malicioso escondidos, integração com assinaturas de IDS. A visualização disponível é a de "Security Events"e mostra um compilado de todas essas atividades principais. Um exemplo destas atividades é o da figura 3.3, podemos ver várias regras sendo alertadas em cada *host*, por exemplo para tentativa de login com usuário não existente.

Por meio desses eventos de segurança é que são reportados os *malwares* que potencialmente podem ser detectados e também a exposição de *rootkits*, conforme veremos na seção 5.4 sobre o experimento

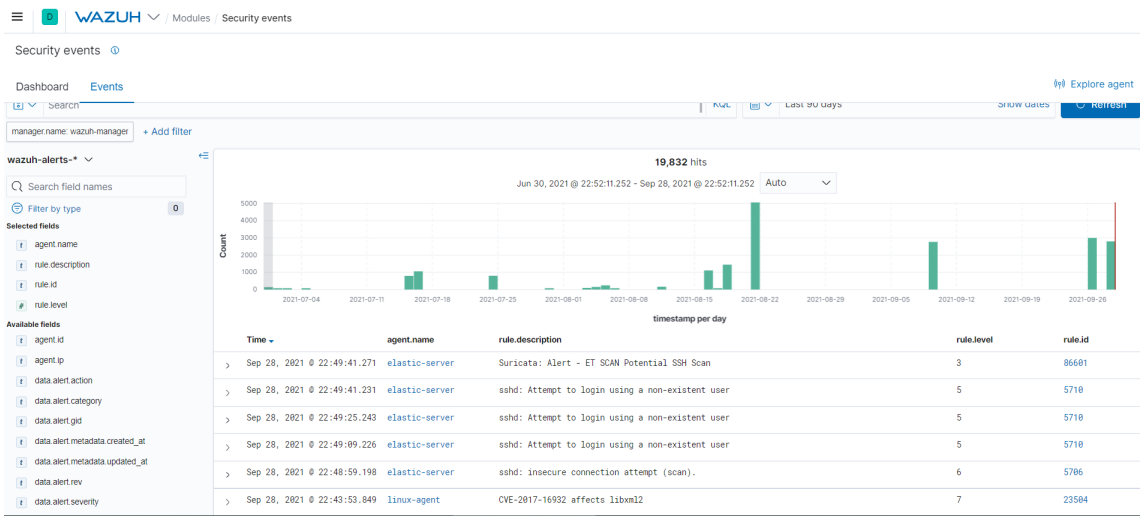

Figura 3.3: Tela de eventos de detecção de intrusão.

realizado no trabalho.

3.4.2 Logs de dados

É o módulo do Wazuh que registra eventos do sistema operacional e aplicações do *host*, monitorando por exemplo, versão do sistema, aumento do uso da memória, atividades de login etc. Um exemplo de atividade do sistema é o da imagem 3.4, que mostra uma das máquinas com a memória sobreutilizada.

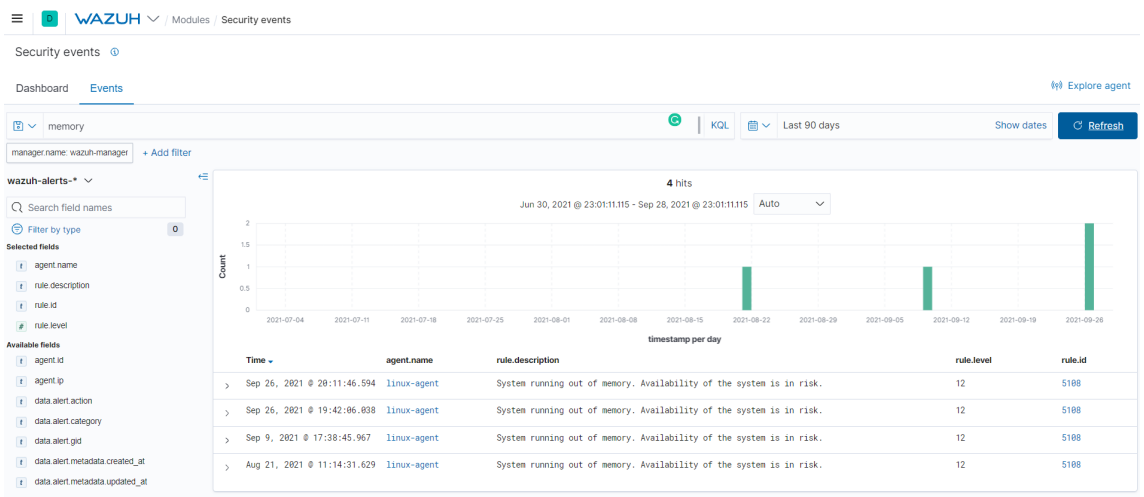

Figura 3.4: Exemplo tela de logs de sistema.

O funcionamento deste componente consiste na leitura e encaminhamento das mensagens de logs originadas no sistema operacional e outra aplicações para o servidor Wazuh, que, por sua vez, realiza as análises identificando por exemplo qual aplicação originou o log e aplica as regras com base em campos contidos no log bruto.

Daí dependendo do padrão de regra em que o log se encaixar, o Wazuh os enrique incluindo por exemplo a técnica e tática do modelo Mitre ATT&CK ou em relação à controles regulatórios de conformidade.

3.4.3 Monitoramento da Integridade de Arquivos

O *File Integrity Monitoring* (FMI - monitoramente da Integridade de Arquivos) é o componente responsável pelo monitoramento em tempo real do sistema de arquivos procurando por alterações de conteúdo, permissões e outros parâmetros.

Essa solução geralmente é utilizada quando é preciso ter controle sobre acesso ou mudanças de dados ou arquivos sensíveis. Os metadados de arquivos monitorados incluem: as somas de verificação MD5, SHA1, SHA2, tamanho, permissões, que usuário performou a alteração.

Assim, o componente responsável por monitorar esses arquivos é o `syscheck`, na qual ele armazena as somas de verificação criptográficas e os outros parâmetros e compara regularmente estes valores com os do objeto sendo utilizado no momento pelo sistema.

3.4.4 Gerenciamento de Vulnerabilidades

O módulo de detecção de vulnerabilidades do Wazuh monitora as versões dos *hosts* em relação às vulnerabilidades associadas de acordo com base de dados de terceiros de CVEs (*Common Vulnerabilities and Exposure*). Nesse caso, é possível utilizar múltiplas fontes para coletar informações acerca de vulnerabilidades. Por padrão, o Wazuh utiliza as bases de dados do Microsoft Security Response Center [25], base de dados online oficial do National Vulnerability Database [26], base de dados online oficial de CVEs para Linux Debian [27], base de dados de CVEs para RedHat e CentOS [28] e base de dados online oficial para Ubuntu [29].

Um exemplo é a tela da imagem 3.5, que mostra diferentes CVEs que foram encontrados nos *hosts* monitorados. Além disso, temos informações sobre as remediações necessárias para correção da vulnerabilidade e referências técnicas sobre o assunto. Mais sobre isso será abordado na sessão de resultados.

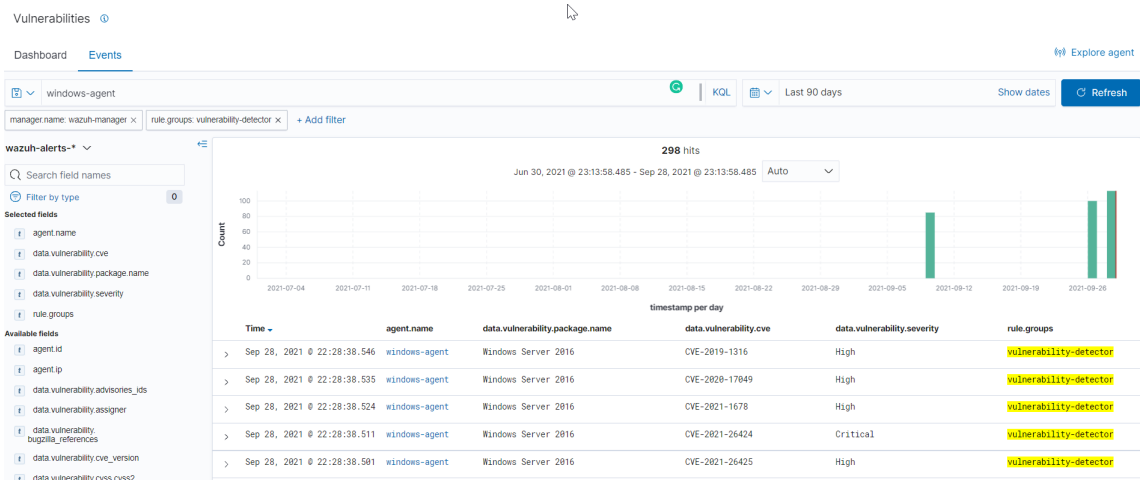

Figura 3.5: Exemplo tela com alertas de vulnerabilidades referentes à um servidores Windows.

3.4.5 Avaliação de Configurações de Segurança

A central de *Security Configuration Assessment* - SCA, ou simplesmente, de Avaliação de Configurações de Segurança do Wazuh estabelece algumas verificações de segurança para os *hosts* baseadas na versão do sistema, em aplicações que rodam nos *hosts*, de forma a notificar aplicações vulneráveis e sem atualização. O Wazuh também relaciona essas avaliações de segurança com controles regulatórios de conformidade.

A Avaliação de Configurações de Segurança é realizada de forma periódica e reporta erros de configuração e boas práticas de segurança que devem ser seguidas de acordo com a versão do sistema para aumentar a postura de segurança da informação do ambiente e reduzir a superfície de ataque. Para isso, o Wazuh coleta informações e organiza um inventário do ativo, com todas as aplicações que executam nele, reconhece quantas interfaces de rede o dispositivo possui, quais serviços permitem conexão com o mundo ou não.

Podemos ver na figura 3.6 para uma máquina Windows do ambiente do projeto exemplos de recomendações feitas e a porcentagem de quantas delas são acatadas na configuração do ambiente

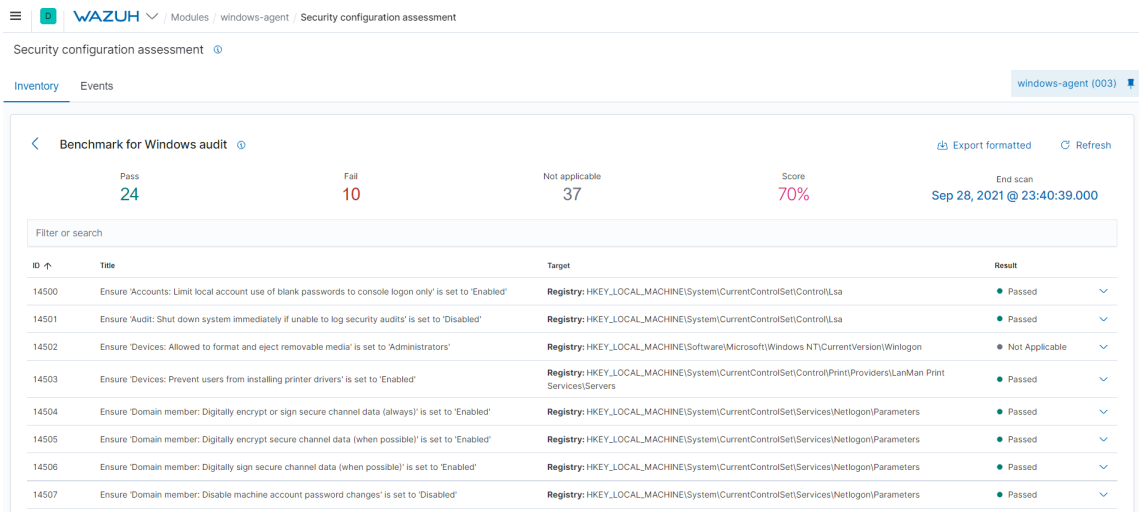

Figura 3.6: Exemplo tela com recomendações de melhores práticas para sistemas Windows.

3.4.6 Controles Regulatórios de *Compliance*

Essa função é utilizada quando se deseja estar em conformidade com aspectos técnicos de padrões regulatórios. Dessa forma forma, é possível escolher entre os padrões Payment Card Industry Data Security Standard (PCI DSS), General Data Protection Regulation (GDPR), NIST Special Publication 800-53 (NIST 800-53), Good Practice Guide 13 (GPG13), Trust Services Criteria (TSC SOC2), Health Insurance Portability and Accountability Act (HIPAA) e cada um possui o conjunto de regras para os controles que são obrigatórios. Dessa forma, um administrador é capaz de verificar o nível de *compliance* e quais controles precisam de atenção em seu ambiente.

3.4.7 Wazuh para Segurança de Nuvem

No caso do monitoramento de segurança de nuvem, é possível fazer a nível de cliente final ou a nível de infraestrutura. No primeiro caso, é possível fazer a implantação de instâncias na nuvem (como será abordado nesse projeto). No monitoramento de infraestrutura, coleta-se atividades dos serviços da nuvem concentrando os alertas e eventos de todos eles em um lugar só, a console do Wazuh. Essa funcionalidade é compatível com diversos serviços dos três provedores de nuvem principais AWS, Azure e Google Cloud Platform, de confome menciona a documentação da própria ferramenta Wazuh [24], então essa funcionalidade também funcionaria com um único painel centralizador de segurança da informação em um ambiente *multi-cloud*.

3.4.8 Wazuh para Segurança de Contêineres

Essa função monitora eventos entre as operações de contêineres em tempo real, podendo ser integrado ao Docker ou Kubernetes como um assinante ingerindo os eventos e alertando acerca de eventos suspeitos. Outra possibilidade de monitoramento seria instalando os agentes Wazuh nos nós de Docker e Kubernetes. A última possibilidade seria integrar à infraestrutura de serviços de nuvem para contêineres, por exemplo o Google GKE ou o Amazon EKS, daí nesse caso teríamos o Wazuh realizado análises em cima dos logs de auditoria providos pelos serviços.

3.4.9 Integração do Wazuh com Suricata

O Wazuh de forma individual é um HIDS (*Host-based Intrusion Detection System*), de acordo com a documentação da própria ferramenta Wazuh [24]. No caso de ser necessário inspecionar o tráfego de rede de uma máquina, que geralmente não é monitorado, um NIDS (*Network Intrusion Detection System*) é um excelente complemento ao HIDS.

Para esse projeto, implementaremos a solução Suricata, que é um NIDS utilizado para monitorar tráfego em equipamentos de rede ou o tráfego de rede em hosts Linux. O Suricata é uma solução de código aberto para monitoramento de segurança de rede, baseado em assinatura e é agnóstico de porta, então ele é capaz de identificar alguns protocolos como por exemplo HTTP, DNS, TLS mesmo que estejam usando portas não-padrão.Os logs gerados pelo Suricata são um arquivo JSON, então o resultado da integração com o Wazuh é bem suave.

Nas seções adiante descreveremos com detalhes essa integração e quais foram os resultados obtidos.

Resumindo, a figura 3.7 mostra como todos os componentes relatados nas seções de 3.4.1 a 3.4.9 integram-se entre si.

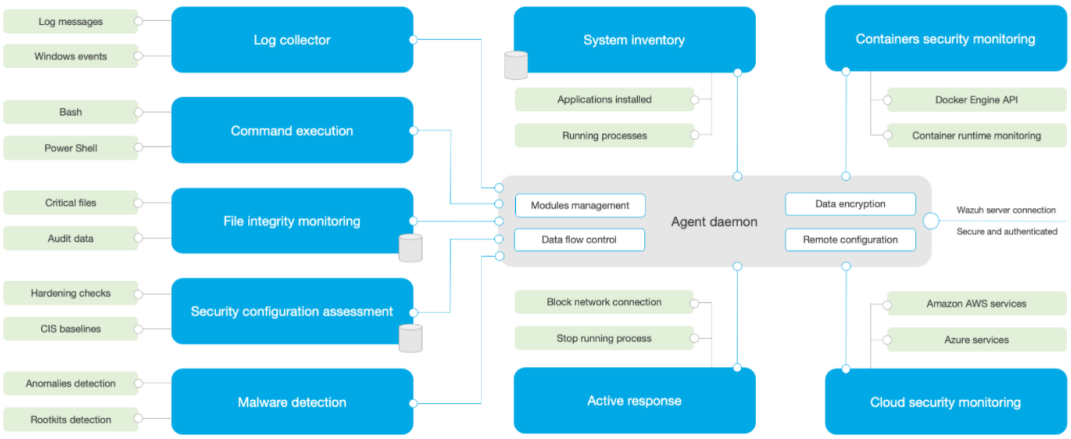

Figura 3.7: Integração entre os módulos no agente Wazuh. Fonte: Documentação Wazuh, acessado em setembro de 2021.

4 ARQUITETURA PROPOSTA

Este capítulo descreve o método de pesquisa e processos utilizados para conduzir o experimentos realizados nesse trabalho. Ainda, descreve-se as topologias lógicas e físicas propostas para construção do ambiente controlado onde os testes serão realizados e como se deu a construção da infraestrutura do projeto cujo objetivo principal é implantar conceitos de segurança da informação durante os testes de penetração em ambiente controlado na nuvem de forma que seja possível avaliar a performance de uma ferramenta de EDR na identificação e contenção de incidentes, conforme explicitado na seção 1.2.2.

Ao final deste capítulo, espera-se que seja possível entender como pode ser executada a implementação das tecnologias envolvidas na solução, assim como o papel de cada etapa na construção do ambiente controlado.

4.1 METODOLOGIA

Este trabalho visa avaliar a efetividade de detecção de eventos de segurança por uma ferramenta de EDR gerados por um atacante externo em relação à um ambiente controlado. Então, pode-se classificar esse trabalho como um experimento-teste, onde propriedades de um sistema são avaliados ou testados com o fim de determinar o quão efetivo ele se equipara às suas especificações ou outros critérios, de acordo com a classificação de Tedre [30]. No contexto deste projeto, o sistema é a ferramenta de EDR instalada em uma máquina virtual em uma infraestrutura controlada e a propriedade avaliada foi a sua capacidade de detecção de eventos de segurança, que foram avaliados simulando ataques individualmente ao ambiente controlado e posteriormente analisando os resultados de detecção.

Esse projeto utiliza a técnica de emulação de pesquisa na qual *softwares* reais são executados em um ambiente virtualizado, de acordo com a classificação de Gustedt e Quinson [31]. No caso a ferramenta real executada era o EDR Wazuh em uma máquina virtual hospedada pela AWS. As vantagens em se desenvolver a pesquisa baseada em emulação são:

- Segurança: como a máquina virtual está em uma rede isolada da máquina que o hospeda e consequentemente da máquina do pesquisador em virtude da virtualização, consegue-se prover a segurança no momento da realização dos ataques.
- Reiteratividade: permite que os testes de intrusão e de detecção sejam facilmente repetidos para validação.

Na arquitetura desse projeto, considera-se para todos os cenários de teste do capítulo 5 que o atacante estará posicionado do lado de fora da rede com o tráfego via internet com destino à máquinas específicas do ambiente controlado (a depender do cenário) cujas interfaces estão associadas a IPs públicos acessíveis da internet, conforme ilustrado na figura 4.1.

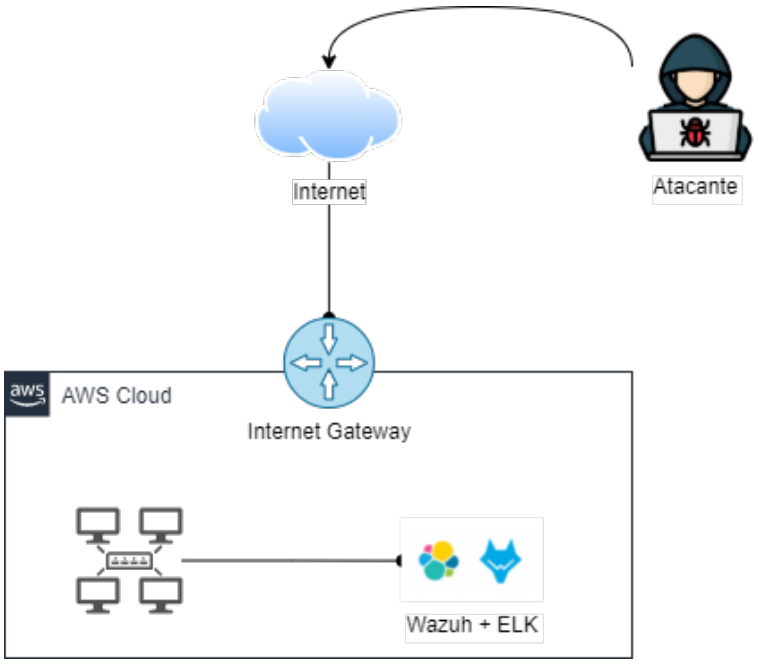

Figura 4.1: Diagrama representativo da metodologia empregada. Fonte: autora.

Assim, como metodologia adotada, a implementação e execução do projeto deu-se em 5 passos descritos abaixo:

- **Passo 1 - Configuração do ambiente em nuvem**: realização de toda a configuração lógica das redes e componentes que são necessário para desenvolvimento do projeto, bem detalhado na seção 4.3.
- **Passo 2 - Configuração do EDR Wazuh**: instalação e configuração do serviço de EDR nos moldes que bem atendem ao projeto, detalhando esse desenvolvimento na seção 4.4.
- **Passo 3 - Configuração da pilha Elastic**: instalação e configuração da pilha Elastic, integrando-o ao EDR de forma a visualizar os eventos que se encaixam em cada caso de uso. Esta configuração é explorada na seção 4.5.
- **Passo 4 - Estruturação de coleta e visualizações dos logs**: esse passo consiste na estruturação segundo templates e encaminhamento dos logs realizado pelos agentes, e por tanto está diluído nas seções 4.6, 4.4.
- **Passo 5 - Realização de ataques e testes de detecção**: esse passo consiste em elaborar e executar ataques direcionados ao ambiente controlado e validar a detecção pela ferramenta de EDR. Este passo está detalhado no capítulo 5.

4.2 TOPOLOGIA PROPOSTA

Para solucionar o problema de se prover visibilidade de eventos de segurança da informação à administradores de infraestrutura, é necessário primeiramente construir um ambiente que permita o desenvol-

vimento da solução. Ainda, é necessário escolher as ferramentas levando em conta os fatores de custo, performance, confiabilidade e disponibilidade de cada uma.

O ambiente base para implementação da arquitetura escolhido foi a AWS, conforme já detalhado na seção 2.2.1, provedor de nuvem consolidado no mercado. Os recurso de computação são compartilhados entre os clientes do provedor e isso faz com que a AWS seja um ambiente altamente disponível de alta escalabilidade, características imprescindíveis para este projeto.

Essa infraestrutura está localizada em um *datacenter* pertencente à AWS em Virgínia do Norte, nos Estados Unidos, na zona de disponibilidade representada pelo código `us-east-1d`. Essa região foi escolhida pois tipicamente tem os elementos de cobrança mais baratos em relação à outros datacenters localizados nos Estados Unidos ou no Brasil.

A AWS não informa explicitamente qual o tipo de hardware ou hipervisor utilizado na virtualização está sendo empregado no momento em que se reserva os recursos ou em que se executa as instâncias, porém informa o processador e algumas características dele de acordo com a classe da instância que foi contratada.

Foram usados então 4 instâncias no total na mesma rede, ou seja, podemos considerar que todas possuem comunicação com o roteador virtual da AWS dentro da VPC. Como está representado na figura 4.2, foram usados 4 Intel Xeon(R), 3 com configuração de 1 CPU virtual e 1 GB de memória RAM e um deles com 2 CPUs virtuais e 8GB de memória RAM. Em relação ao armazenamento, todas utilizam o EBS disponibilizado pela AWS.

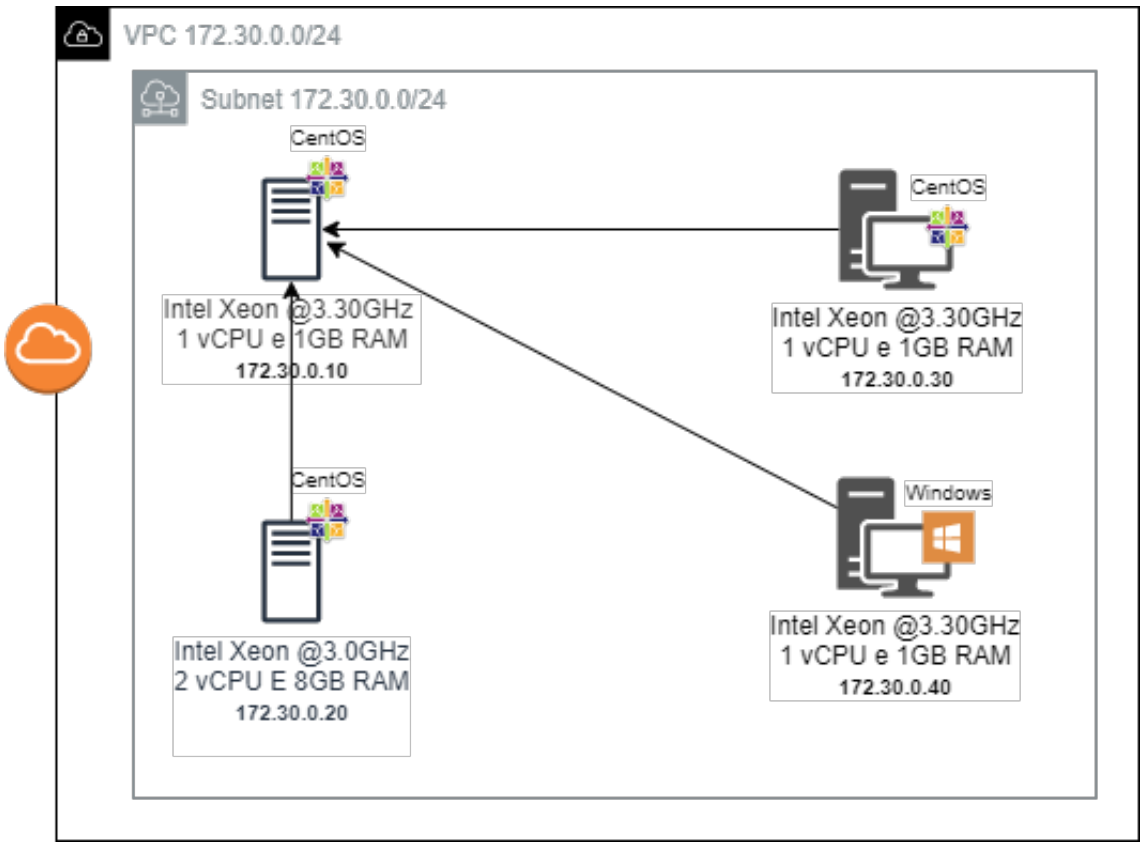

Figura 4.2: Diagrama da topologia que representa o cenário físico da solução. Fonte: autora do projeto.

A quantidade de armazenamento dedicada à cada instância foi estimada com base nos requisitos de espaço do sistema operacional e dos componentes que seriam instalados em cada um, utilizando o serviço de armazenamento em blocos EBS. Então, o tamanho do volume de armazenamento e o restante das configurações utilizadas nas máquinas do projeto estão na tabela 4.1 de forma resumida.

Tabela 4.1: Configuração das máquinas virtuais.

Nome	IP	Sistema operacional	vCPU	Memória	Armazenamento
Elastic Server	172.30.0.10	CentOS 7	2	8 GB	20 GB
Wazuh Server	172.30.0.20	CentOS 7	1	1 GB	20 GB
Linux Agent	172.30.0.30	CentOS 7	1	1 GB	8 GB
Windows Agent	172.30.0.40	Windows Server 2016	1	1 GB	30 GB

Nos servidores e em uma das estações de trabalho, está instalado o sistema operacional CentOS 7. Na estação de trabalho remanescente, está instalado o sistema operacional Windows Server 2016. As imagens disponibilizadas pela AWS não incluem o Windows 10, que seria o correto no cenário real de se ter em uma estação de trabalho, logo, utilizou-se o Windows Server 2016. Para esse trabalho, não haverá perdas em seguir com essa troca pois as funções do Windows Server não interferirão no projeto e nem farão falta as funcionalidades do Windows 10. Adicionalmente, o agente da ferramenta Wazuh é compatível com ambas versões de SO.

Tendo o ambiente físico estabelecido e também a metodologia de pesquisa definida, para atingir o objetivo final de realizar testes e ataques cibernéticos em um ambiente controlado, propõe-se a utilização de uma solução aberta e sem nenhum custo para monitoramento de eventos de segurança da informação sobre uma plataforma de análise de logs para em conjunto formar um SIEM.

Para se aproximar o máximo possível de um cenário real onde a alta disponibilidade e escalabilidade dos serviços são importantes, propõe-se a utilização da implantação distribuída da solução de EDR, o servidor Wazuh, e da solução de análise e gerenciamento de logs, o servidor Elastic, com o tráfego criptografado entre os serviços utilizando TLS. Além disso, de acordo com a metodologia, propõe-se que o *endpoint* do atacante está inicialmente localizado fora da rede. A figura 4.3 é uma representação da topologia lógica empregada no projeto.

Assim, na figura 4.2, vemos que o servidor denominado Wazuh Server possui logicamente os componentes: Wazuh Manager, que realizará o gerenciamento de todos os agentes monitorados, o Filebeat, que encaminhará os logs recebidos dos agentes para o Elasticsearch, e, por fim, a Wazuh API, que se integrará com o Kibana para prover uma interface de busca para o usuário. Esses componentes se comunicam sobre TLS de forma segura e autenticada. Na seção 4.4, detalha-se essas integrações e configurações.

O Wazuh será a solução base de segurança da informação, enquanto utilizaremos conjuntamente a pilha Elastic no servidor Elastic Server da figura 4.3, incluindo o Elasticsearch, conforme já explorado na seção 3.3, pois ela provê componentes para enriquecimento dos logs com o Logstash, visualização por *dashboards* em uma interface de usuário com o Kibana e uma base de dados não-relacional para os eventos com o Elastic. Aconfiguração do Elastic Server está detalhada na seção 4.5.

Dessa forma, juntaremos as capacidades de EDR do Wazuh com as de análise e gerenciamento de logs do Elasticsearch para formar uma ferramenta de SIEM e monitorarmos o ambiente completamente.

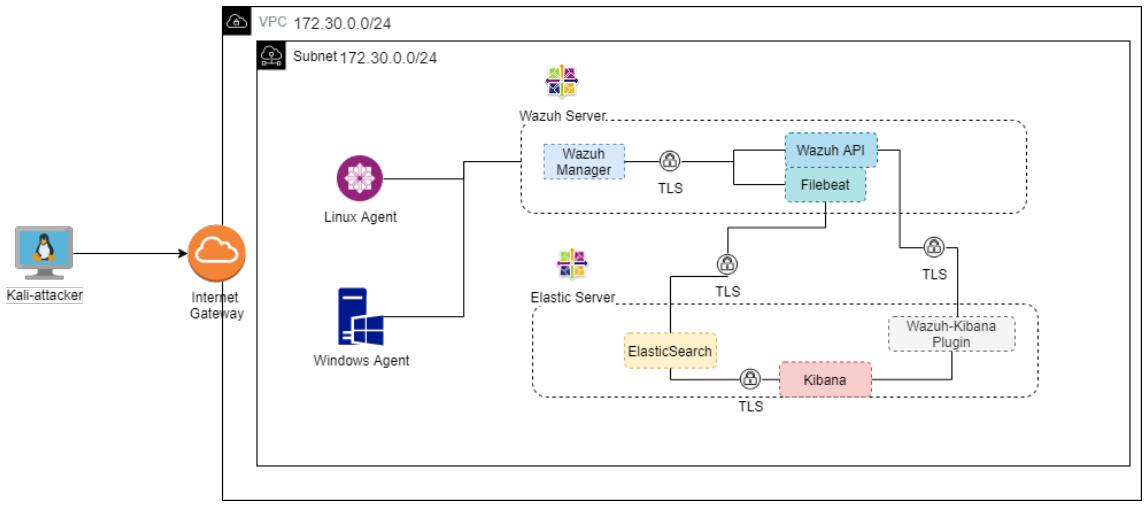

Figura 4.3: Topologia lógica proposta para solução do projeto e como é a integraçãoe ntre os componentes de cada servidor. Fonte: autora.

4.3 CONFIGURAÇÃO DO AMBIENTE EM NUVEM

Esse capítulo abordará como foi o desenvolvimento e configuração do ambiente de testes a ser atacado e onde estará localizado a ferramenta de monitoramento Wazuh. Esse ambiente será propositalmente comprometido de forma controlada.

O ambiente base para implementação da arquitetura escolhido foi a AWS, provedor de nuvem pública explorado na seção 2.2.1. Isso significa que os recursos de computação são compartilhados entre os clientes do provedor, fazendo com que a AWS seja um ambiente altamente disponível de alta escalabilidade, características imprescindíveis para este projeto.

Essa infraestrutura está localizada fisicamente em um datacenter pertencente à AWS em Virgínia do Norte, nos Estados Unidos, na zona de disponibilidade representada pelo código `us-east-1d`. Essa região foi escolhida pois tipicamente tem os elementos de cobrança mais baratos em relação a outros datacenters oferecidos para hospedagem de serviços.

A partir de uma conta na AWS, para conter os recursos do projeto conforme explicitado na seção 2.2.1, foi criada uma Virtual Private Connection (VPC), um objeto da nuvem AWS que representa uma rede e possui um roteador implícito. Esta VPC engloba uma sub-rede, denominada Wazuh Lab Subnet, de endereçamento 172.30.0.0/24 reservado para uso de redes privadas, segundo a RFC-1918 [32], onde estão alocados todos os recursos do projeto.

Essa sub-rede possui atrelada uma tabela de rotas, objeto da AWS utilizado para direcionar o tráfego. As rotas estão configuradas como na figura 4.4. A coluna *Target* indica por onde deve ser encaminhado o tráfego quando os destinos especificados forem os da coluna *Destination*, que contém intervalos de endereços IP na notação CIDR, ainda segundo padronizado na RFC-1918 [32].

Portanto, a tabela de rotas da figura 4.4 indica que todo tráfego destinado ao intervalo 172.30.0.0/24 deve ser roteado dentro da VPC. A segunda linda indica que qualquer outro destino de IPv4, representado por 0.0.0.0/0 (diferente de 172.30.0.0/24, pois essa já é a primeira rota) deve ser encaminhado ao objeto

Internet Gateway. O Internet Gateway é responsável por realizar a conexão entre a VPC e a Internet e é o alvo de todo o tráfego roteável destinado para fora da sub-rede.

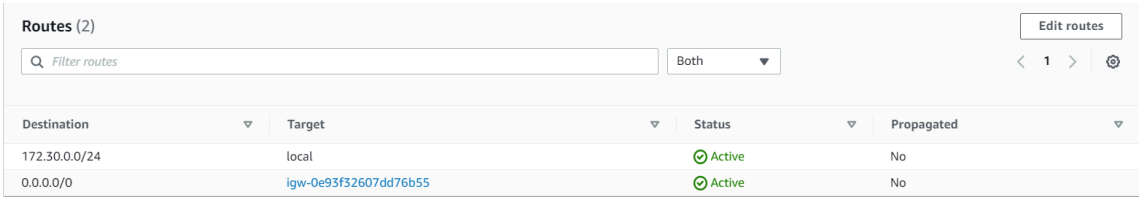

Figura 4.4: Tabela de roteamento da sub-rede Wazuh Lab Subnet. Fonte: autora.

O controle de tráfego a nível de instância é realizado pelos grupos de segurança, que funcionam como *firewall* virtual com regras para tráfego de entrada e para o tráfego de saída baseado em protocolos e portas. Como todo tráfego é negado por padrão, as regras sempre são de liberação, mantendo o estado do fluxo, então quando um fluxo de entrada é liberado, as respostas estão automaticamente liberadas para saída, independente das regras de saída que estão definidas.

Dois grupos de segurança foram configurados: um (Wazuh Linux) para instâncias com distribuição Linux e outra para a instância Windows (Wazuh Windows). O grupo de segurança Wazuh Linux da figura 4.5 permite tráfego SSH e HTTP/HTTPs originados da internet para as instância, permite tráfego irrestrito dentro da VPC. Este grupo de segurança está associado aos servidores Elastic Server, Wazuh Server e Linux-agent mencionados na tabela 4.1.

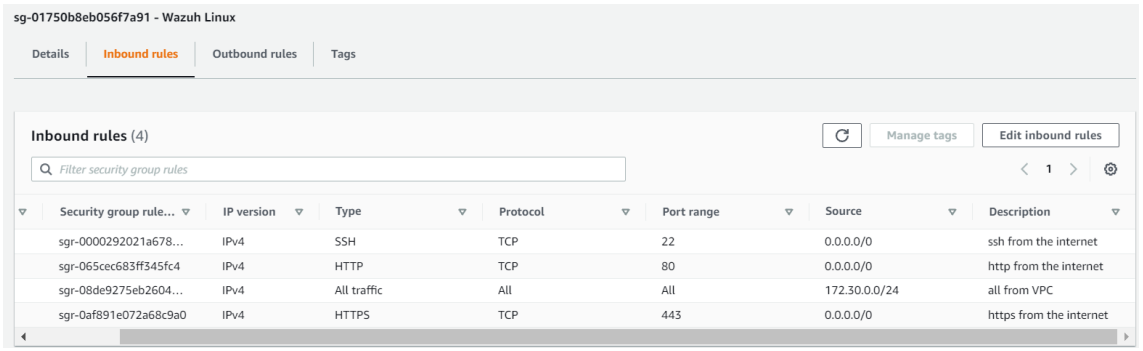

Figura 4.5: Regras de entrada do grupo de segurança Wazuh Linux. Fonte: autora.

O grupo de segurança Wazuh Windows permite acesso RDP originado da internet, acessos dentro da VPC, conforme a figura 4.6. A este grupo de segurança está associado o servidor Wazuh-agent mencionando na tabela 4.1.

Ambos grupos de segurança permitem todo o tráfego de saída com origem nas instâncias, conforme figura 4.7.

Dentro da sub-rede, estão os servidores mencionados na topologia física proposta 4.2. Cada um deles possui uma interface de rede com um endereço privado e um IP público utilizando NAT para acesso a Internet.

Quando estão ligadas, as instâncias recebem um endereço de IP público do intervalo de endereços pertencente à AWS, um nome de *host* DNS externo baseado nesse endereço IP (exemplo: <ec2-ip-address.compute-1.amazonaws.com>) e um endereço de IP privado dentro da VPN. O DNS é resolvido como

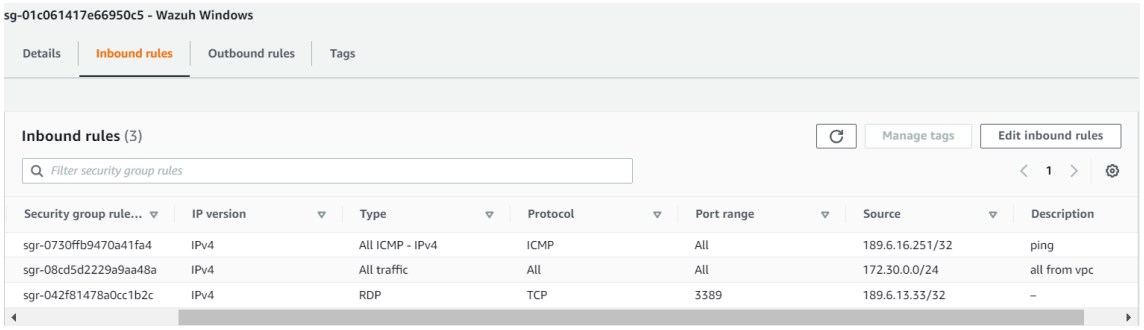

Figura 4.6: Regras de entrada do grupo de segurança Wazuh Windows. Fonte: autora.

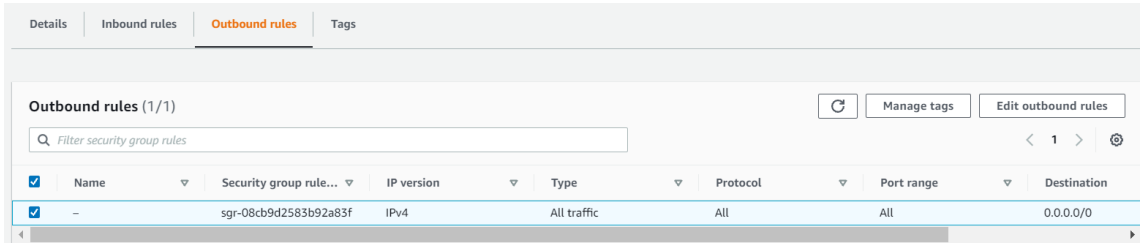

Figura 4.7: Regras de saída do grupos de segurança Wazuh Windows e Wazuh Linux. Fonte: autora.

endereço IP público da instância fora da VPC e como endereço de IP privado quando o tráfego é de dentro da VPC. Além disso, a AWS informa que o endereço público é mapeado para o endereço privado por meio de tradução de endereços de rede, NAT padronizado na RFC-2663 [33]. Através dos IPs públicos, realiza-se a comunicação e acesso remoto com os servidores.

Conforme foi proposto da topologia lógica da figura 4.3, a forma de configuração do ambiente escolhida foi a distribuída. Nesse tipo de desenvolvimento, os componentes são instalados em servidores separados. Os componentes são um servidor Wazuh, que inclui um nó de gerenciamento Wazuh contendo sua API e o Filebeat. O outro componente contém a pilha Elastic de código aberto, contando com a instalação do Kibana e o *plugin* do Wazuh para Kibana. A documentação dá a opção de o Kibana e o Elasticsearch estarem ou não no mesmo servidor. Para manter os custos do projeto viáveis, tomou-se a decisão de manter as duas aplicações na mesma máquina.

A VM Elastic Server hospeda os serviços referentes ao Elasticsearch e Kibana, portanto é através dela que se acessa a interface web do Kibana. Por último, há duas máquinas virtuais que simulam máquinas clientes comuns no ambiente. Ambas possuem o agente do Wazuh instalado e são os alvos dos ataques e tráfego malicioso a serem testadas.

No ambiente distribuído, a comunicação entre os hosts é criptografada utilizando certificados SSL gerados pela ferramenta `elasticsearch-certutil`, mais detalhes sobre essa instalação serão providos no anexo III.

4.4 CONFIGURAÇÃO DO WAZUH SERVER

A VM Wazuh server hospeda os serviços de API do Wazuh e o Filebeat. Todas as regras são implementadas nos arquivos de configuração do Wazuh desse servidor, que por sua vez se comunica com o servidor do ElasticSearch.

O servidor Wazuh é o responsável por coletar e analisar dados dos agentes, desencadear alertas para anomalias detectadas e monitorar os status dos agentes. Conforme a visão geral da ferramenta realizada na seção 3.4, as análises do Wazuh utilizam *threat intelligence*, base de dados de terceiros para CVEs, base de dados do framework Mitre ATT&CK e bases de requisitos de compliance, tal como PCI DSS, GDPR [34], NIST 800-53 e outros.

A máquina virtual que hospeda a aplicação do Wazuh tem as seguintes responsabilidades:

- Serviço de registro de agentes: realiza o registro de novos agentes via autenticação com senha.
- Serviço de conexão de agentes: realiza a comunicação de dados dos agentes, validando a identidade de cada um.
- Componente de análise: serviço que realiza a análise dos dados recebidos a partir de mensagens de log brutas.
- API Wazuh: interface que produz a integração entre a interface de usuário do Kibana e infraestrtura do serviço Wazuh, fazendo o gerenciamento de regras, monitorando os agentes e as configurações.
- Filebeat: realiza a leitura dos eventos de log após a análise do Wazuh e entrega ao Elasticsearch, estruturando os dados em tempo real.

A API Wazuh é uma API RESTful de código aberto que possibilita a interação com o servidor Wazuh e uma página Web ou qualquer outra ferramenta que realiza requests. Dessa forma, a API serve de interface de usuário e permite executar ações de gerenciamento de agentes e outras funções de gestão remota da infraestrutura do Wazuh.

Sendo assim, o pacote do Wazuh manager está disponível no repositório do fabricante [35]. A instalação da API Wazuh está contida neste mesmo pacote do wazuh manager. A instalação detalhada está no anexo II e após todos os passos, é possível verificar que o servidor está ouvindo eventos enviados pelos agentes na porta 1514 e também o serviço de auto-registro de agentes na porta 1515, conforme a figura .

Figura 4.8: Servidor Wazuh com status "Listen"nas portas utilizadas para serviços do gerenciador. Fonte: autora.

O Filebeat também é um dos serviços executado no Wazuh server. Ele é responsável por carregar e centralizar dados de qualquer pasta de log ou outra localização monitorada. Assim, para o cenário implementado, o filebeat encaminha os alertas e eventos ao servidor Elastic.

Por sua vez, o pacote do filebeat foi baixado a partir do repositório do Wazuh [35] e também os templates que servem de base para configurar os alertas moldados e enviados pelo Filebeat. Essa pré-configuração está contida no arquivo de configuração do filebeat (filebeat.yml). Após realização de todos os passos de instalação do anexo II, tem-se o arquivo configurado da figura 4.9.Esse arquivo também contém o IP da interface LAN do Elasticsearch (172.0.0.20), então o Filebeat encaminha os eventos para esse endereço na porta 9200. Além disso, podemos ver que o serviço está utilizando comunicação segura via HTTPs e utiliza certificados TLS configurados de acordo com o anexo III.

```
[centos@wazuh-manager /]$ cat /etc/filebeat/filebeat.yml
# Wazuh - Filebeat configuration file
filebeat.modules:
  - module: wazuh
    alerts:
      enabled: true
    archives:
      enabled: false

setup.template.json.enabled: true
setup.template.json.path: '/etc/filebeat/wazuh-template.json'
setup.template.json.name: 'wazuh'
setup.template.overwrite: true
setup.ilm.enabled: false

output.elasticsearch.hosts: ['172.30.0.20:9200']
output.elasticsearch.protocol: https
output.elasticsearch.ssl.certificate: "/etc/filebeat/certs/wazuh-manager.crt"
output.elasticsearch.ssl.key: "/etc/filebeat/certs/wazuh-manager.key"
output.elasticsearch.ssl.certificate_authorities: ["/etc/filebeat/certs/ca/ca.crt"]

output.elasticsearch.username: "elastic"
output.elasticsearch.password: "[redacted]"
[centos@wazuh-manager /]$
```

Figura 4.9: Arquivo de configuração do serviço Filebeat. Fonte: autora.

4.5 CONFIGURAÇÃO DO ELASTIC SERVER

A arquitetura da figura 4 conta com um servidor rodando os serviços do Elastic Stack, o Elasticsearch e Kibana. O Elasticsearch é um serviço de busca de texto de código aberto e altamente escalável, descrito na seção 3.3. Já o Kibana é uma ferramenta de visualização dos dados do elasticsearch, permitindo que o usuário crie diagramas, gráficos, mapas e outras opções para melhor percepção dos dados.

Sendo assim, a versão 7.10.2 do Elasticsearch e do Kibana foram baixadas a partir do repositório do Wazuh [35] e também o plugin Wazuh para Kibana. O procedimento de instalação de todos os componentes do servidor estãod etalhados no anexo I.

Por padrão, o servidor do Kibana apenas acessa a interface de loopback, ou seja, o host local, então o arquivo de configuração do kibana (kibana.yml) está configurado para ser acessado a partir de qualquer *host* ("0.0.0.0") na porta 443, porta padrão do HTTPS.

Após realizar todos os passos do anexo I, arquivo de configuração da API do Wazuh (wazuh.yml) possuirá as configurações de credencial de acesso à interface do Elasticsearch, a URL de acesso (https://172.30.0.10), porta padrão 5500, nome de usuário e senha, conforme a figura 4.10.

```
[centos@elastic-server ~]$ cat /usr/share/kibana/data/wazuh/config/wazuh.yml
hosts:
  - wazuhapi:
      url: https://172.30.0.10
      port: 55000
      username: wazuh-wui
      password: wazuh-wui
      run_as: false
```

Figura 4.10: Arquivo de configuração do *plugin* da API Wazuh. Fonte: autora.

Podemos verificar que a instalação funcionou e está operacional acessando o serviço através do DNS público associado ao servidor Elastic, além de possuir os certificados válidos, representado na figura 4.11.

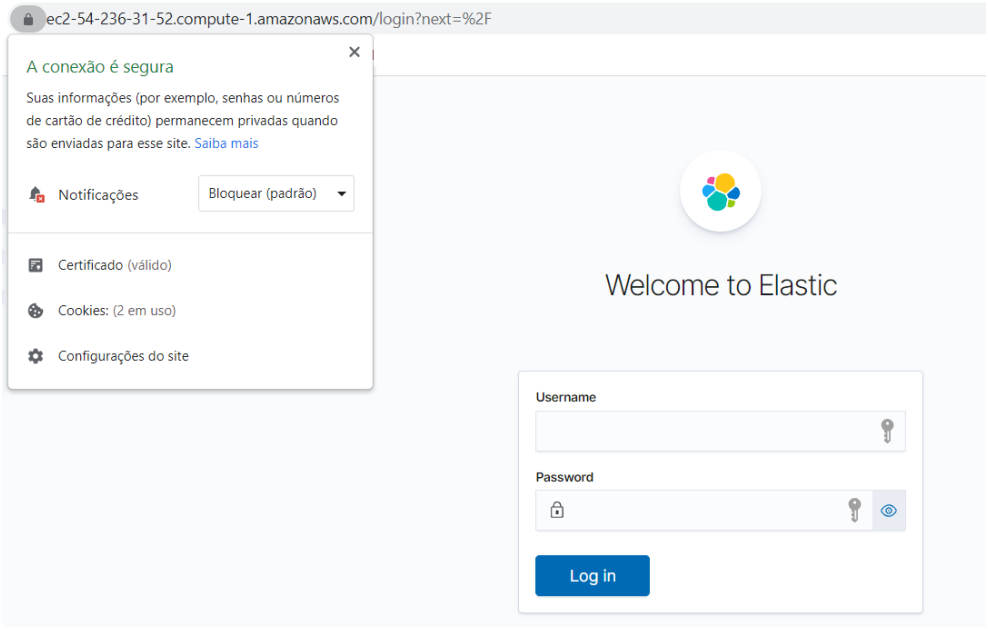

Figura 4.11: Servidor Elastic completamente operacional após instalação. Fonte: autora.

4.6 CONFIGURAÇÃO DE AGENTES DO WAZUH

Os agentes foram instalados nas máquinas em que se desejava fazer o monitoramento. Os agentes de comunicam com o servidor Wazuh de forma segura e autenticada utilizando as configurações do pacote X-Pack Security disponibilizados pelo fabricante Elastic e configurado no anexo III.

O pacote dos agentes para sistema operacional Linux foi retirado a partir do repositório do Wazuh [35] e sua instalação em detalhes está no anexo IV. Os agentes se comunicam com o servidor Wazuh (wazuh manager) pela porta 1514 para envio de dados e pela porta 1515 para registro.

Para a máquina Windows do ambiente, obteve-se a versão 4.2.1 do agente Wazuh para essa plataforma a partir do repositório e para instalação, seguiu-se os passos detalhados no anexo V.

Sendo assim, as máquinas que possuem o agente instalado são as instâncias Linux Agent, Elastic Server e Windows Agent. Após todos os procedimento de instalação e ocnfiguração, podemos ver no

servidor manager Wazuh que os 3 agentes foram registrados com sucesso na figura 4.12.

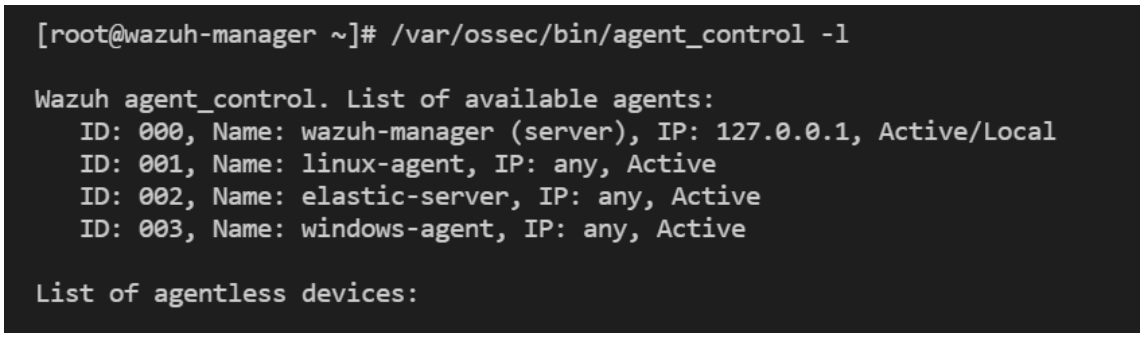

Figura 4.12: Confirmação de comunicação no servidor wazuh manager do registro de todos os agentes instalados.

4.7 CONFIGURAÇÃO DE CERTIFICADOS PARA COMUNICAÇÃO SEGURA

Para manter padrões mínimos de segurança entre as entidades do projeto, implementou-se comunicação segura e autenticação entre os serviços utilizando o X-Pack Security. Como o Elasticsearch e o Filebeat encontram-se em instâncias separadas, uma no Elastic Server e a outra no Wazuh Manager, respectivamente, estabeleceu-se um canal de comunicação TLS (*Transport Layer Security*) criptografado entre essas duas identidades. Além disso, a aplicação Kibana possui acesso liberado para a Internet e por isso o tráfego HTTP também será de forma segura com TLS, utilizando certificados de autenticação.

O certificado do wazuh-manager será utilizado para o Filebeat, que conforme explicado anteriormente, é a aplicação que se comunica enviando eventos ao Elasticsearch.

Para cada host, criou-se utilizando a ferramenta nativa do Elastisearch "elasticsearch-certutil"um arquivo .crt, que é o certificado auto-assinado contendo a chave pública, e um .key, que é a chave privada. Cada um desses arquivos é único para cada instância e todos foram criados a partir do Elastic Server, então depois cada certificado foi transferido para a respectiva instância onde deveria está instalado.

A descrição dos passos para implementação dos certificados de segurança e autenticação estão no anexo III.

5 TESTES E RESULTADOS

Seguindo o modelo *Cyber Kill Chain* da seção 3.2, foram realizados diversos ataques à ferramenta Wazuh para apurar quão eficaz e acurada é a sua performance. Cada um destes ataque será retratado na forma de cenários e explorando como foram as detecções da ferramenta segundo técnicas e táticas do *framework Mitre ATT&CK*, mencionado na seção 3.1.

Os testes e ataques foram realizados durante o período de 30 de junho a 30 de outubro de 2021

5.1 CENÁRIO 1: SCAN NMAP

Conforme dito na seção 4.1, simulando o intruso a partir da instância Kali (100.27.43.63), iremos realizar um escaneamento de portas com a ferramenta *nmap* para o IP público da instância linux-agent (54.236.62.28). Esse cenário implementa a fase de reconhecimento da *Cyber Kill Chain* (seção 3.2).

A ferramenta `nmap` ("*Network Mapper*") é utilizada para realizar descoberta de rede por meio de um escaneamento e determinar quais máquinas estão disponíveis nos ambiente, serviços que essas máquinas oferecem (em qual versão, porta), servidores operacionais em que as máquinas operam e outras características da rede que está sob alvo do escaneamento.

Primeiramente, verifica-se qual IP público intruso para validarmos que a origem do ataque mais adiante pela ferramenta. Depois, executamos o comando de *nmap* e, após a varredura no horário de 1:27 UTC, identificamos que o serviço de SSH está aberto na porta 22 e ainda quais algoritmos são utilizados na chave de acesso SSH utilizada pelo *host* em conjunto com as *fingerprints* da chave pública utilizada, conforme a figura 5.1.

5.1.1 Resultados

Apresenta-se então os resultados obtidos a partir do escaneamento realizado na seção anterior.

Um minuto depois das tentativas terem sido executadas, verificou-se que o Wazuh identificou e alertou o scan como ameaça emergente a partir da origem supracitada, validando que de fato houve um evento de *portscan* "*sshd: insecure connection attempt*"(Tentativa de conexão insegura) com origem do IP 100.27.43.63 (campo *data.src.ip*) em direção ao linux-agent (campo *agent.ip*), conforme figura 5.2.

Além disso, no mesmo log reportado, foi identificado como a técnica T1043 do Mitre de Portas de Uso Comum, que se refere às táticas de Comando e Controle e Exfiltração. Essa correspondência acontece pois atacantes podem tentar usar portas comuns tentando burlar *firewalls* e IDSs na tentativa de se misturarem ao tráfego comum utilizando portas de serviços de uso comum na tentativa de descobrir serviços utilizados e explorar algum *backdoor* ou vulnerabilidade a partir daí.

```
┌──(kali㉿kali)-[~]
└─$ curl 177.71.200.154/getuserip.php


100.27.43.63

┌──(kali㉿kali)-[~]
└─$ nmap -A 54.236.62.28
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-07 01:27 UTC
Nmap scan report for ec2-54-236-62-28.compute-1.amazonaws.com (54.236.62.28)
Host is up (0.00065s latency).
Not shown: 997 filtered ports
PORT    STATE  SERVICE VERSION
22/tcp  open   ssh     OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 84:55:09:01:e2:da:0d:0c:99:b1:ea:21:fb:9f:48:ab (RSA)
|   256 11:38:93:c2:37:5e:51:bb:c5:27:35:46:04:4f:7f:81 (ECDSA)
|_  256 e1:9a:74:e6:35:60:8c:0c:b3:ba:b9:a5:fb:eb:84:25 (ED25519)
80/tcp  closed http
443/tcp closed https

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.15 seconds
```

Figura 5.1: Ataque de varredura de portas com nmap direcionado ao linux-agent, com descobrimento da porta 22 do serviço SSH aberta. Fonte: autora do trabalho.

5.2 CENÁRIO 2: SSH BRUTE-FORCE

Para este teste, foi realizado um ataque de força bruta na conexão SSH com a máquina linux-agent. Esse ataque é utilizado para ganhar acesso à um sistema quando as credenciais são desconhecidas pelo atacante por meio de tentativas de adivinhação dessas credenciais de forma repetitiva. A origem novamente foi da máquina atacante khali-linux a partir da internet, ou seja, direcionado ao IP público da instância linux-agent

A máquina possui acesso via SSH com chaves pré-compartilhadas, portando o ataque de força bruta realizado no teste foi trocando o usuário legítimo "centos"pelo usuário "billy".

Utilizando um cliente SSH, foi realizado algumas tentativas de login pelo comando:

```
# ssh billy@ip-linux-agent
```

5.2.1 Resultados

Apresenta-se então os resultados do experimento realizado na seção 5.2.

Como "billy"não é um usuário autorizado pelo sistema, a conexão é rejeitada antes mesmo de a senha ser solicitada. Após 8 tentativas como essa num espaço de aproximadamente 2 minutos, o Wazuh identificou como um ataque de força bruta SSH.

Esse evento em sua forma bruta se encontra localizado no arquivo /var/log/auth.log, porém o Filebeat é responsável por formatar esses logs somando outros dados que a API do Wazuh adiciona e entregá-los ao Elasticsearch de uma forma amigável ao usuário que vai buscar esses logs na ferramenta. Esse log após o tratamento do Filebeat está armazenado no arquivo /var/ossec/logs/alerts/alerts.json.

Figura 5.2: Alerta referente ao scan realizado no ataque. Fonte: autora do trabalho.

Acessando a ferramenta Kibana, é preciso buscar por "billy"na barra de busca no escopo de tempo grande o suficiente para englobar as tentativas de acesso.

O identificador da regra cuja tentativa bateu é *5710: sshd: Attempt to login using a non-existent user* na figura 5.4. Os detalhes dessa regra mostram que ela procura pelos textos "illegal user"ou "invalid user"que apareceram nos eventos de log já gatilhados pela regra-pai 5700, que detecta todos os tipos de ventos sshd e aciona regras mais específicas derivadas dela. Pode-se ver também que o nível de severidade da regra é 5, ou seja, baixo, pois até então é apenas uma tentativa de login sem sucesso isolada. Porém, podemos notar na figura 5.5 que, após 7 tentativas do ataque de força bruta, a 8ª é caracterizada pela ferramenta como ataque de fato. Nesse caso, a regra gatilhada é a *5712: sshd: brute force trying to get access to the system*.

Acessando a coleção de regras no arquivo /var/ossec/ruleset/rules/0095-sshd_rules.xml, é possível ver que essa regra é gatilhada quando a mesma regra 5710 é ativada no total de 8 vezes em um período de 2 minutos (*timeframe*="120") pelo mesmo IP de origem. O nível de severidade dessa regra é mais alto pois um grupo de tentativas falhas de login via SSH da mesma origem comumente é um forte sinal de ataque de força bruta.

A autenticação por força bruta se encaixa na segunda fase da *Cyber Kill Chain*, armamento. Uma vez que o intruso é capaz de adivinhar corretamente quais são as credenciais de acesso à um sistema, possuirá então o controle de visualizar e copiar arquivos importantes e também executar códigos maliciosos. É importante que a ferramenta tenha detectado essas tentativas pois, embora tenha sido apenas um teste de poucas tentativas, atacantes do mundo real testam centenas de credenciais, criando vários eventos que causam ruído na rede. Assim que o analista confirmar que foi um ataque de força bruta, pode partir para ações de remediação, como por exemplo bloquear o acesso SSH ao host.

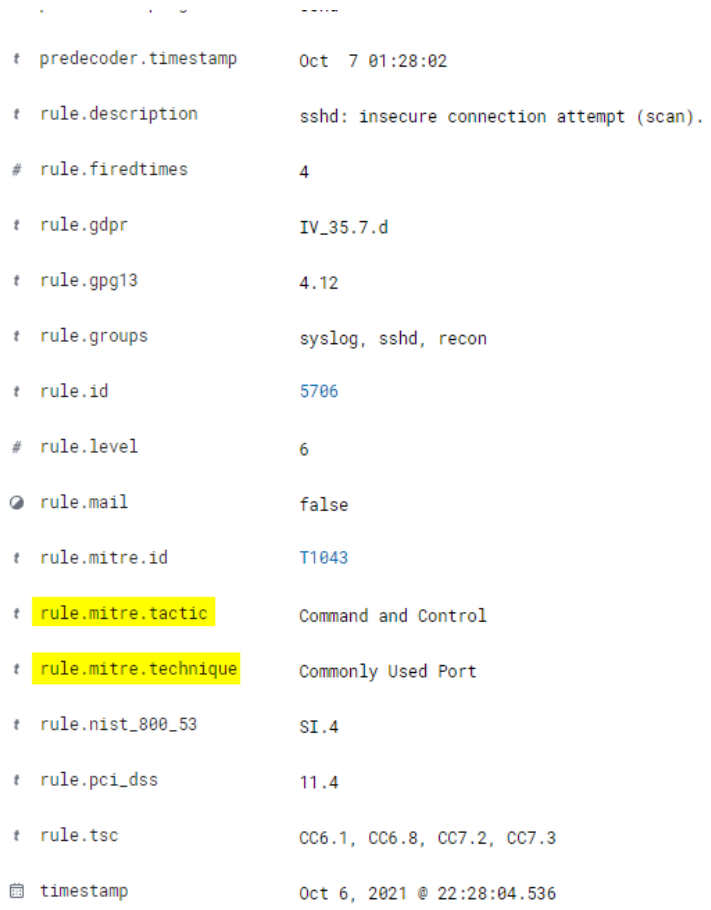

Figura 5.3: Técnica de Portas de Uso Comum reportadas pelo Wazuh durante o ataque de *nmap*. Fonte: autora do trabalho.

5.3 CENÁRIO 3: RDP BRUTE-FORCE

Nesse cenário, foi executado um ataque de força bruta via RDP na instância Windows Agent. O RDP é o protocolo de acesso remoto do sistema operacional Windows. O ataque de força bruta via RDP ocorre quando o atacante inicia várias tentativas de adivinhação de senha tendo como alvo uma máquina conectada à internet, como é o caso da instância linux-agent, se tornando uma porta de entrada à atacantes.

5.3.1 Resultados

Análogo aos resultados da seção 5.2, apresenta-se os resultados obtidos na ferramenta a partir da realização do ataque de Força Bruta de RDP.

Da mesma forma que no ataque de força bruta SSH, há uma detecção para cada falha de login e outra de severidade mais alta quando uma quantidade suficiente de falhas de login sucessivas são identificadas. Análogo ainda ao ataque da seção 5.2, os ataques partiram do Kali Linux em direção ao IP público da instância linux-agent.

Usando o cliente de Desktop Remoto do Windows, foram realizadas algumas tentativas de login na máquina windows-agent em uma janela de tempo curta de aproximadamente 2 minutos. Similarmente,

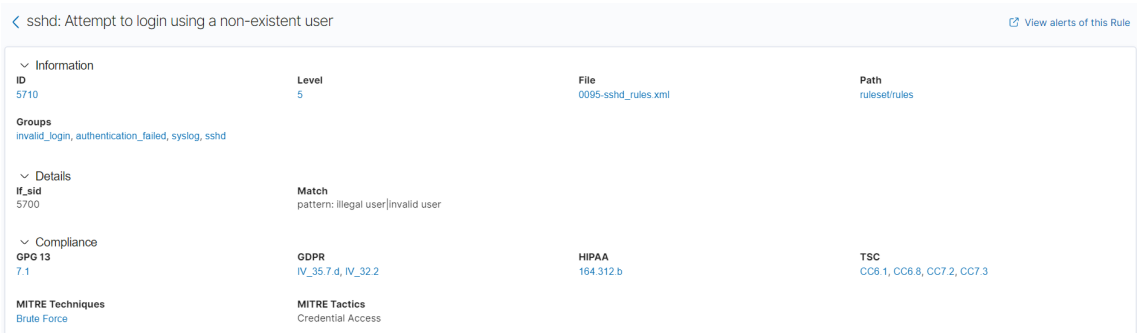

Figura 5.4: Regra 5710, referente à técnica de Força Bruta dentro da tática de Acesso Credencial.

Jul 16, 2021 @ 08:37:32.430	linux-agent	sshd: brute force trying to get access to the system.	10	5712	189.6.16.81	billy
Jul 16, 2021 @ 08:37:26.398	linux-agent	sshd: Attempt to login using a non-existent user	5	5710	189.6.16.81	billy
Jul 16, 2021 @ 08:37:18.393	linux-agent	sshd: Attempt to login using a non-existent user	5	5710	189.6.16.81	billy
Jul 16, 2021 @ 08:37:10.411	linux-agent	sshd: Attempt to login using a non-existent user	5	5710	189.6.16.81	billy
Jul 16, 2021 @ 08:37:02.376	linux-agent	sshd: Attempt to login using a non-existent user	5	5710	189.6.16.81	billy
Jul 16, 2021 @ 08:36:54.389	linux-agent	sshd: Attempt to login using a non-existent user	5	5710	189.6.16.81	billy
Jul 16, 2021 @ 08:36:46.354	linux-agent	sshd: Attempt to login using a non-existent user	5	5710	189.6.16.81	billy
Jul 16, 2021 @ 08:36:38.346	linux-agent	sshd: Attempt to login using a non-existent user	5	5710	189.6.16.81	billy

Figura 5.5: Eventos de tentativas de brute force SSH. Fonte: autora do trabalho.

pode-se ver que após 7 tentativas falhas de login, o Wazuh identifica o evento como força bruta, conforme a figura 5.6.

Jul 16, 2021 @ 08:33:43.160	Multiple Windows Logon Failures	10	60204	189.6.16.81	billy
Jul 16, 2021 @ 08:33:40.989	Logon Failure - Unknown user or bad password	5	60122	189.6.16.81	billy
Jul 16, 2021 @ 08:33:38.723	Logon Failure - Unknown user or bad password	5	60122	189.6.16.81	billy
Jul 16, 2021 @ 08:33:36.379	Logon Failure - Unknown user or bad password	5	60122	189.6.16.81	billy
Jul 16, 2021 @ 08:33:34.147	Logon Failure - Unknown user or bad password	5	60122	189.6.16.81	billy
Jul 16, 2021 @ 08:33:31.583	Logon Failure - Unknown user or bad password	5	60122	189.6.16.81	billy
Jul 16, 2021 @ 08:33:28.816	Logon Failure - Unknown user or bad password	5	60122	189.6.16.81	billy
Jul 16, 2021 @ 08:33:25.785	Logon Failure - Unknown user or bad password	5	60122	189.6.16.81	billy

Figura 5.6: Eventos de tentativas de brute force via RDP.

Assim como no ataque de força bruta via SSH, esse ataque via RDP foi detectado como técnica de Brute Force no *framework* Mitre sob a tática de *Credential Access*, em detalhe na figura 5.7. Quando o RDP está habilitado em uma instância virada para a internet, ela se torna facilmente uma porta de entrada aos intrusos por vias deste ataque que visa tomar o acesso contando com a possibilidade de este acesso estar protegido por senhas fracas e facilmente adivinháveis.

Algumas formas simples de se prevenir o ataque de força bruta via RDP é simplesmente desabilitando o protocolo no *host* ou deixando acessível somente via rede privada fazendo uso de VPN. Usar senhas fortes em conjunto com segundo fator de autenticação (MFA) também é um método efetivo para evitar comprometimentos via RDP, pois dessa forma, o atacante não será capaz de adivinhar a senha utilizada e ainda terá que ultrapassar a segunda autenticação com MFA.

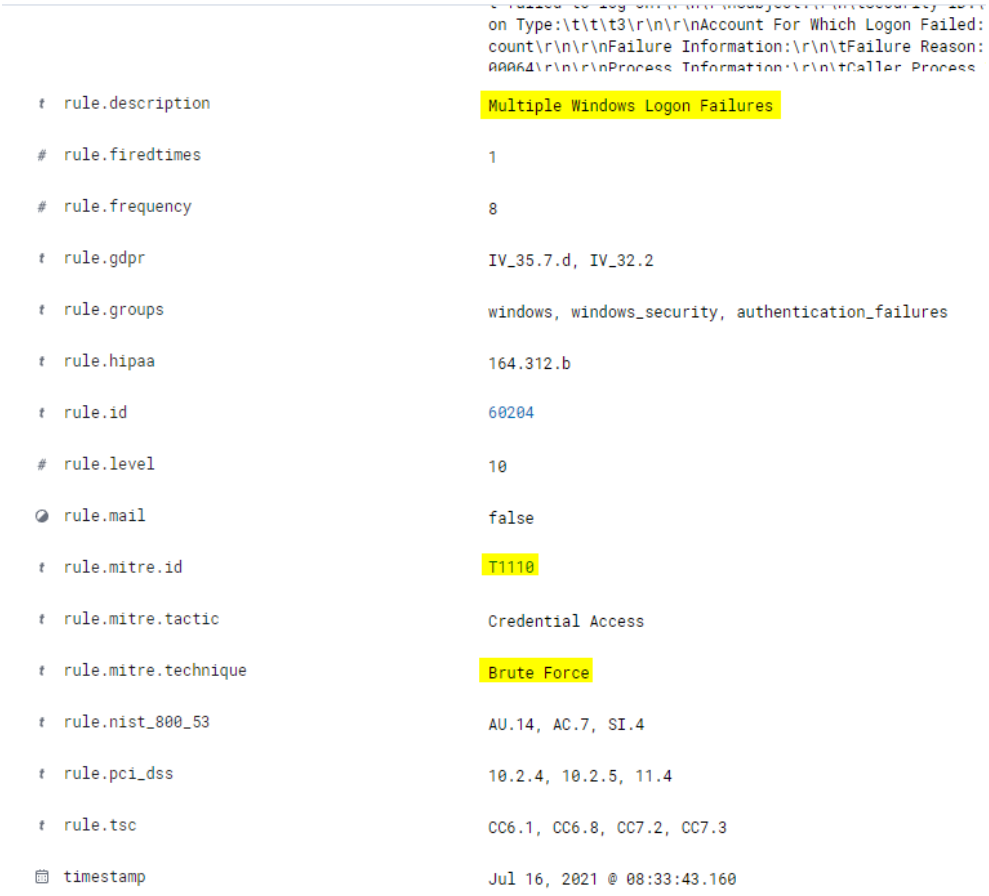

Figura 5.7: Eventos de tentativas de brute force via RDP.

5.4 CENÁRIO 4: EXPOSIÇÃO DE KERNEL-MODE ROOTKIT

Nesse cenário, é proposto a implementação de um *rootkit* em modo *kernel* na máquina alvo como prova de conceito do bom funcionamento do Wazuh para detecção de *rootkits*. Um *rootkit* é utilizado por atacantes para esconder a presença de programas, acessos, serviços ou, por exemplo, malware. Isso é feito interceptando e modificando chamadas do sistema operacional modificando informações do sistema.

Esse ataque pode ser considerado a fase de armamento da Cyberkill chain mencionada na seção 3.2 já que pode ser considerada como uma forma de preparação do ambiente realizada pelo atacante para esconder um código malicioso ou escalar privilégios de froma indetectável.

De forma detalhada, comportamento do *rootkit* é de se esconder da lista de processos módulo *kernel* e também esconder processos selecionados do "ps"(*process status*), módulo nativo do linux que provê informações sobre processos rodando no sistema.

Iremos simular que o atacante ganhou o acesso remoto após ter obtido com sucesso as credenciais de acesso remoto via shell seguro (SSH) e está visando esconder um *malware* utilizando um *rootkit* para escondê-lo de chamadas do sistema operacional que poderiam detectá-los.

Para este cenário, foi escolhido o Diamorphine, *rootkit* para *kernels* Linux 2.6.x/3.x/4.x/5.x e ARM64. O diamorphine esconde e revela qualquer processo enviando um sinal 31 pelo comando kill() a nível de

sistema operacional no linux. No entanto, o Wazuh ainda é capaz de detectar as chamadas de sistemas setsid(), getpid() e kill(), ou seja, detecta processos com comportamento de baixo nível.

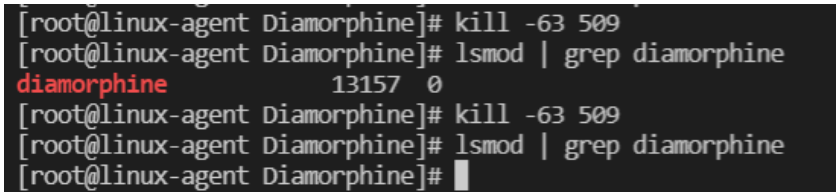

Figura 5.8: Primeiramente tornando o diamorphine detectável para o sistema e identificando-o e depois tornando-o indetectável.

Após a instalação do *rootkit*, a partir do repositório oficial da ferramenta Diamorphine [36], basta carregá-lo e por padrão, ele está escondido e rodando o comando `lsmod grep diamorphine` não é possível detectá-lo. É necessário enviar um sinal 63 do comando kill() para torná-lo visível. O *rootkit* foi mantido escondido para propositalmente dificultar a detecção pelo Wazuh, conforme demonstrado na figura 5.8.

5.4.1 Resultados

Referente ao experimento de execução de um *rootkit*, apresenta-se como os resultados se encaixam na metodologia empregada.

Sendo assim, quando o processo está escondido, na ferramenta podemos visualizar as regras que foram compatíveis com esse evento. Buscando por "rootkit", foi possível visualizar alguns logs no período de tempo e na máquina linux-agent em que o teste foi realizado. O log obtido foi o da figura 5.9. O campo "full_log"exibe uma mensagem informado o identificador do processo que está escondido, .

Podemos observar no log da figura 5.9 que o Wazuh não identificou exatamente em qual técnica de defesa do *framework* Mitre o ataque se encaixou, conforme observamos nos cenários 5.3, 5.1, 5.2. A regra 510 referente ao monitoramento de processos escondidos na figura 5.10 mostra que nela estão faltando os campos `rule.mitre.id`, `rule.mitre.tactic` e `rule.mitre.technique`, referentes, respectivamente, ao identificador, nome da tática e nome da técnica correlacionada à matriz do Mitre. Esta pode ser considerada uma falha da ferramenta, uma vez que a correlação com a matriz é feia de forma nativa no Wazuh e, para este caso, não foi feita corretamente.

Assim, o uso de *rootkits* se encaixa na técnica de mesmo nome e que pode ser utilizada sob a tática de Evasão de Defesa do Mitre, cujo ID é T1014, e consiste em empregar técnicas para evitar uma detecção ao longo do comprometimento da máquina, ou ainda pode ser utilizada sob a técnica de Função de Inibição de Resposta (*Inhibit Response Function*), cujo objetivo é evitar que qualquer alarme ou resposta seja ativada durante o cenário de ataque.

É uma tarefa difícil mitigar o uso de ferramentas como essa porém de ferramentas de EDR como o Wazuh, pode-se ver que é possível detectá-la com informações que podem ser utilizadas para ações de reposta e contenção do incidente por analista de segurança, por exemplo, em qual host o analista precisa atuar e qual o identificador do processo malicioso que deve ser eliminado.

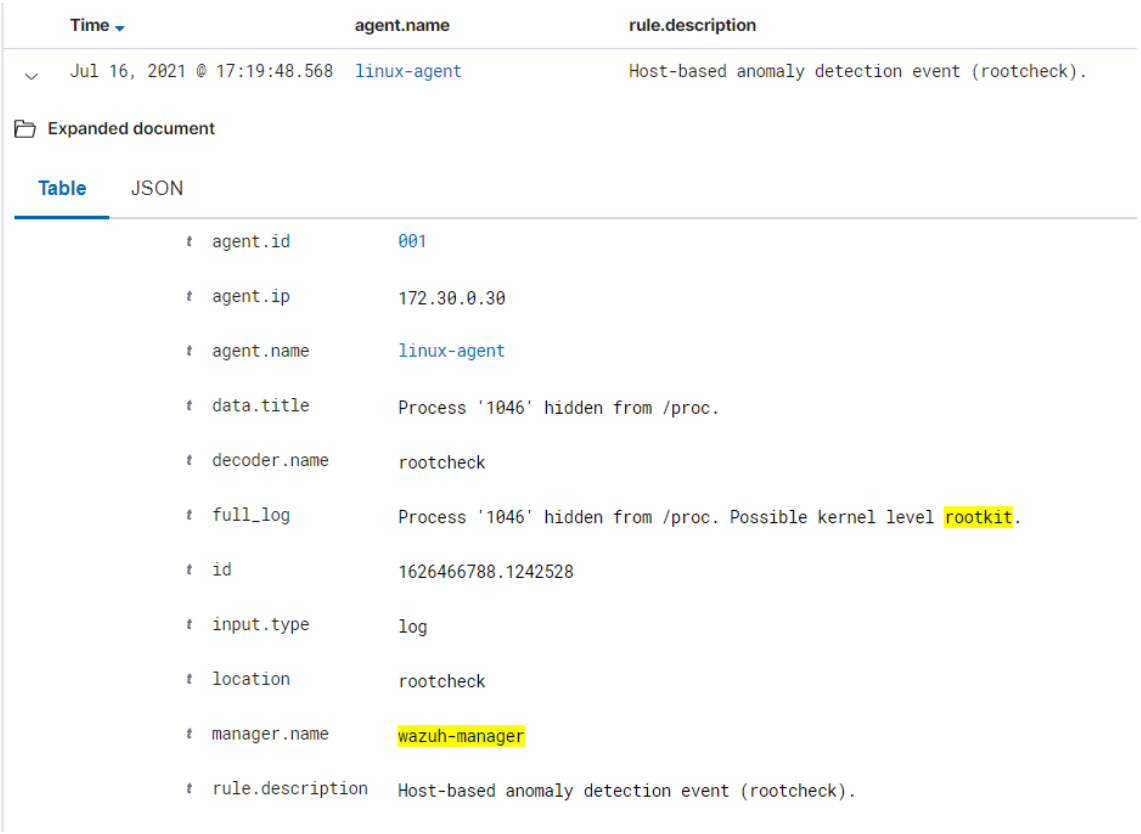

Figura 5.9: Detalhes do log rootkit.

5.5 CENÁRIO 5: ATAQUE SHELLSHOCK

Para este cenário, explora-se uma falha de segurança presente no Bash de sistemas operacionais Linux que foi descoberta em 2014 e possui alguns identificadores de vulnerabilidades associados: CVE-2014–6271, CVE-2014–6277, CVE-2014–6278, CVE-2014–7186.

Este é um exemplo de vulnerabilidade do tipo Execução de Códigos Arbitrários (ACE - *arbitrary code execution*)

Essa vulnerabilidade torna possível que um atacante execute linhas de comando em seu interesse a partir de requests HTTP maliciosos vindo da Internet. Assim, nesse experimento, iremos simular que a máquina linux-agent é um servidor web recebendo *requests* web com comandos embutidos característicos do Shellshock, e portanto, verificando qual alerta foi produzido na ferramenta em decorrência desse ataque.

Com o servidor web `nginx` instalado na instância, configura-se o arquivo de configuração do agente `/var/ossec/etc/ossec.conf` Wazuh para monitorar logs de acessos e de erros, conforme configuração abaixo:

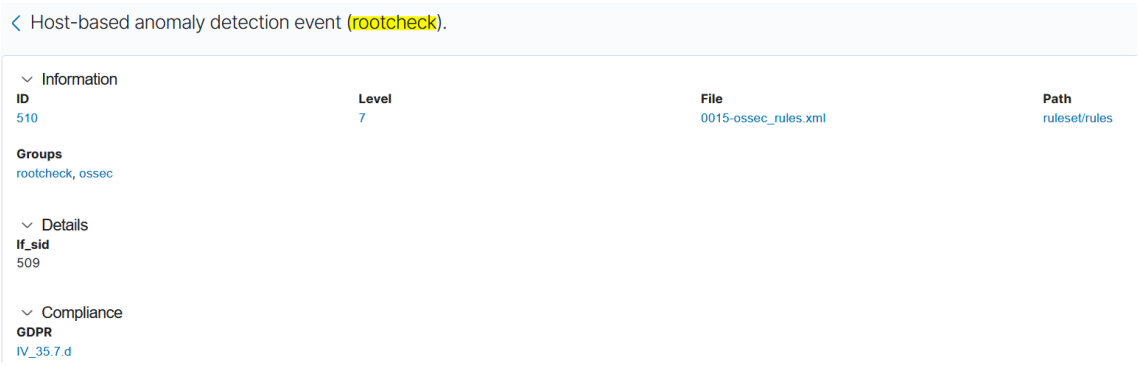

Figura 5.10: Regra 510 referente ao monitoramento de rootkits a nível de host, na qual o ataque com rootkit Diamorphine se enquadrou.

```
<ossec_config>
    <localfile>
        <log_format>apache</log_format>
        <location>/var/log/nginx/access.log</location>
    </localfile>

    <localfile>
        <log_format>apache</log_format>
        <location>/var/log/nginx/error.log</location>
    </localfile>
</ossec_config>
```

A partir da máquina atacante kali linux, cujo IP de origem é 3.238.69.169, cria-se uma variável (`ShellshockTarget`) para representar a instância que será vítima do Shellshock, que no caso será a instância linux-agent e possui IP públixo 3.238.57.131, e depois executa-se o comando da figura 5.11.

No cabeçalho User Agent, está presente a sequência de caracteres `() { :; };`, que é utilizada para interpretar o que estará contido nesse cabeçalho não como uma sequência de caracteres, mas sim como um comando Nesta linha de comando, o atacante realiza uma tentativa de injetar no cabeçalho User Agent a linha `/bin/cat /etc/passwd` na tentativa de se obter como reposta uma leitura do conteúdo do arquivo de senhas localizado em `/etc/passwd`.

5.5.1 Resultados

Nesta seção, detalha-se a detecção dessa injeção de comandos via *request web* no ataque conhecido como Shellshock da seção 5.5.

Através do log da figura 5.12, pode-se ver como a ferramenta identificou os ataques "Shellshock attack detected". Podemos ver que a origem (`data.srcip`) é exatamente o IP da instância Kali linux, localizada na região de Norte da Virgínia nos Estados Unidos, e com direção ao linux-agent.

Ainda, no mesmo log e na figura 5.13, vemos exatamente qual foi o *request* HTTP solicitado: "GET

```
┌──(kali㉿kali)-[/]
└─$ ShellshockTarget="3.238.57.131"

┌──(kali㉿kali)-[/]
└─$ curl --insecure $ShellshockTarget -H "User-Agent: () { :; }; /bin
/cat /etc/passwd"
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Welcome to CentOS</title>
  <style rel="stylesheet" type="text/css">
```

Figura 5.11: Exploração de vulnerabilidade do bash pra execução de comandos via *requests web*.

/ HTTP/1.1() :; : /bin/cat /etc/passwd". Nesta mesma entrada, pode-se ver que a mensagem que originou essa detecção está nos arquivos de log do servidor web nginx que foi utilizado para simular o servidor web. Portanto, pode-se verificar uma vantagem na capacidade do EDR em se detectar intrusão em *hosts* é que o tráfego malicioso sobre HTTPS não passa indetectável pelo sistema, já que ele monitora os logs do servidor web em vez do conteúdo de pacotes que poderiam estar criptografados por transmissões HTTPS.

Por último, verifica-se que a ferramenta foi capaz de identificar e classificar este ataque segundo o *framework* Mitre ATT&CK, identificando-o entre as táticas de *Privilege Escalation* (escalação de privilégios) e *Initial Access* (acesso inicial), sob as técnicas de Exploitation for Privilege Escalation, na qual um ataque utiliza de erros em um software ou serviço para executar códigos e escalar privilégios. Está também identificado sob a técnica de Exploit Public-Facing Application (Exploração de aplicações com acesso público), onde o atacante obtém vantagem de uma brecha de segurança localizada num servidor que está aberto para acesso da Internet.

As formas conhecidas de detecção consistem em monitorar os logs da aplicação por comportamento malicioso que indiquem tentativas ou explorações bem sucedidas, conforme realizou-se nesse experimento, e também consiste em inspecionar pacotes buscando por padrões conhecidos de tráfego malicioso, utilizando por exemplo Web Application Firewalls (WAF - Firewalls de Aplicações Web).

Formas de se mitigar ataques à aplicações que permitem acessos de origem da Internet consistem em: alocar em rede apartada os servidores web que estão expostos, pois então em caso de comprometimento, o atacante não conseguirá realizar movimento lateral; realizar escaneamento de vulnerabilidades nos sistemas buscando por vulnerabilidades conhecidas; atualizar atualizações nos softwares e serviços com *patches* de segurança que geralmente corrigem vulnerabilidades críticas conhecidas;

Figura 5.12: Detecção pelo Wazuh referente ao ataque Shellshock.

Figura 5.13: *Requests* identificados como maliciosos pelo EDR.

Figura 5.14: Táticas e técnicas de classificação do Mitre em relação ao ataque Shellshock realizado pelo EDR.

6 CONCLUSÃO

Nesse projeto, foi desenvolvido um ambiente controlado para testes de intrusão e ataques de segurança para analisar resultados referentes à proteção de *endpoints*. A solução consistiu na implementação de uma ferramenta de EDR para análise dos eventos de segurança com a capacidade de correlacionar os logs com táticas e técnicas do modelo Mitre ATT&CK.

Assim, o ambiente controlado foi desenvolvido em uma rede apartada da rede em que o atacante estava localizado, simulando ataques vindo de origens externas não conhecidas. Assim, cada ataque foi classificado entre as fases da *Cyber kill chain*.

Para validar os testes, implementou-se em conjunto as ferramentas de EDR e e a pilha Elastic em conjunto para formar a ferramenta de SIEM, assim provendo o acompanhamento das atividades intrusivas. A infraestrutura do projeto foi implementada seguindo boas práticas de segurança, ou seja, os componentes se comunicavam através de tráfego criptografado com TLS e autenticado com certificados se segurança. O ambiente possuía regras mínimas de roteamento e controle de tráfego para evitar exposições desnecessário e melhor simulasse um ambiente do mundo real.

Por fim, como formas de tentativa de comprometimento do ambiente, cinco cenários foram propostos. Avaliou-se então como foi a postura da ferramenta de EDR diante dos cenários e os dados reportados personalizados de acordo com cada regra em que o ataque se encaixou defensivamente de acordo com o *framework* Mitre ATT&CK.

6.1 TRABALHOS FUTUROS

Como trabalhos futuros, sugere-se: testar outros ataques e o comportamento do Wazuh, podendo-se até mesmo testar os mesmo ataques deste projeto e testar como são as repostas e análises de um EDR de outro fabricante. Ou ainda, executar todo o ambiente por tempo indeterminado e vulnerável a ataques originados da Internet e daí obter análises do mundo real.

Em outra linha, sugere-se realizar esta implementação dos agentes do EDR Wazuh em *endpoints* de outros sistemas operacionais suportados, por exemplo, macOS, máquinas AIX, Solaris e então realizar ataques direcionados à estas versões. Além disso, ainda utilizando o Wazuh, sugere-se testar suas outras funcionalidades, por exemplo, a que se especializa em realizar segurança de contêineres, e daí realizar ataques direcionados a essa forma de ambiente, ou visando a segurança de serviços na nuvem, performando ataques também personalizados para este fim.

REFERÊNCIAS BIBLIOGRÁFICAS

1 CROWDSTRIKE. *2020 Global Threat Report [Online]*. aug 2020. Acessado em agosto de 2021. Disponível em: <https://go.crowdstrike.com/rs/281-OBQ-266/images/Report2020CrowdStrikeGlobalThreatReport.pdf>.

2 CANADIAN Centre for Cyber Security. Cyber threat and cyber threat actors. aug 2021 [Online]. Acessado em agosto de 2021. Disponível em: <https://cyber.gc.ca/en/guidance/cyber-threat-and-cyber-threat-actors>.

3 REDHAT. *Red Hat: O que é IaaS? [Online]*. Acessado: outubro de 2021. Disponível em: <https://www.redhat.com/pt-br/topics/cloud-computing/what-is-iaas>.

4 YADAV, T.; RAO, A. M. Technical aspects of cyber kill chain. In: ABAWAJY, J. H.; MUKHERJEA, S.; THAMPI, S. M.; RUIZ-MARTÍNEZ, A. (Ed.). *Security in Computing and Communications*. Cham: Springer International Publishing, 2015. p. 438–452. ISBN 978-3-319-22915-7.

5 CENTRO de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil - CERT.br - Estatísticas dos Incidentes Reportados ao CERT.br [Online]. aug 2021. Acessado em agosto de 2021). Disponível em: <https://www.cert.br/stats/incidentes/>.

6 KEMMERER, R. Cybersecurity. In: *25th International Conference on Software Engineering, 2003. Proceedings.* [S.l.: s.n.], 2003. p. 705–715.

7 BLACKLEY, J. A.; PELTIER, T. R.; PELTIER, J. Information security fundamentals. *Auerbach Publications*, v. 1, n. 1, p. 1–100, 2004. Disponível em: <https://doi.org/10.1201/9780203488652>.

8 YEBOAH-BOATENG, E. *Cyber-Security Challenges with SMEs in Developing Economies: Issues of Confidentiality, Integrity & Availability (CIA)*. Tese (Doutorado), Denmark, 2013.

9 WIKIPEDIA. *Communication Endpoint [Online]*. aug 2020. Acessado em outubro de 2021. Disponível em: <https://en.wikipedia.org/wiki/Communication_endpoint>.

10 YOO, S. J. Study on improving endpoint security technology. *Convergence Security Journal*, v. 18, n. 3, p. 19–25, 2018.

11 BHATT, S.; MANADHATA, P. K.; ZOMLOT, L. The operational role of security information and event management systems. *IEEE Security Privacy*, v. 12, n. 5, p. 35–41, 2014.

12 ANASTASOV, I.; DAVCEV, D. Siem implementation for global and distributed environments. In: *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*. [S.l.: s.n.], 2014. p. 1–6.

13 LAUREANO, M. A. P.; MAZIERO, C. A.; JAMHOUR, E. Detecção de intrusão em máquinas virtuais. *Simpósio de Segurança em Informátical*, v. 18, n. 3, p. 19–25, 2003.

14 KUROSE, J. F.; ROSS, K. W. *Computer Networking: A Top-Down Approach*. 7. ed. Boston, MA: Pearson, 2016. ISBN 978-0-13-359414-0.

15 LEAVITT, N. Internet security under attack: The undermining of digital certificates. *Computer*, v. 44, n. 12, p. 17–20, 2011.

16 SHERMAN, A. T.; DELATTE, D.; NEARY, M.; OLIVA, L.; PHATAK, D.; SCHEPONIK, T.; HERMAN, G. L.; THOMPSON, J. Cybersecurity: Exploring core concepts through six scenarios. *Cryptologia*, Taylor & Francis, v. 42, n. 4, p. 337–377, 2018. Disponível em: <https://doi.org/10.1080/01611194.2017.1362063>.

17 ABLON, L. *Data Thieves: The Motivations of Cyber Threat Actors and Their Use and Monetization of Stolen Data*. Santa Monica, CA: RAND Corporation, 2018.

18 JADEJA, Y.; MODI, K. Cloud computing - concepts, architecture and challenges. In: *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*. [S.l.: s.n.], 2012. p. 877–880.

19 MELL, P.; GRANCE, T. The nist definition of cloud computing [online]. In: . [s.n.], 2011. NIST Special Publication 800-145. Disponível em: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>.

20 WIKIPEDIA. *Amazon Web Serivces [Online]*. aug 2021. Acessado em outubro de 2021. Disponível em: <https://pt.wikipedia.org/wiki/Amazon_Web_Services>.

21 ALI, S. H. Amazon web services (aws) – an overview of the on-demand cloud computing platform. *International Conference for Emerging Technologies in Computing*, v. 332, p. 399–418, 2020.

22 GEORGIADOU, A.; MOUZAKITIS, S.; ASKOUNIS, D. Assessing mitre att&ck risk using a cyber-security culture framework. *Sensors*, v. 21, n. 9, 2021. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/21/9/3267>.

23 ATT&CK, M. *Defense Evasion*. October 2018. Acessado em outubro de 2021. Disponível em: <https://attack.mitre.org/tactics/TA0005/>.

24 WAZUH Documentation [Online]. aug 2021. Acessado em setembro de 2021. Disponível em: <https://wazuh.com/>.

25 MICROSOFT Security Response Center [Online]. aug 2021. Acessado em setembro de 2021. Disponível em: <https://www.microsoft.com/msrc>.

26 CVES from the National Vulnerability Database [Online]. aug 2021. Acessado em setembro de 2021. Disponível em: <https://nvd.nist.gov/>.

27 DEBIAN. *CVEs for Debian Linux distributions [Online]*. aug 2021. Acessado em setembro de 2021. Disponível em: <https://www.debian.org/>.

28 REDHAT. *CVEs for Red Hat and CentOS Linux distributions [Online]*. aug 2021. Acessado em setembro de 2021. Disponível em: <https://access.redhat.com/>.

29 CANONICAL. *CVEs for Ubuntu Linux distributions [Online]*. aug 2021. Acessado em setembro de 2021. Disponível em: <https://canonical.com/>.

30 TEDRE, M. *The Science of Computing: Shaping a Discipline*. [S.l.]: CRC Press, 2014. ISBN 978-1-48-221770-4.

31 GUSTEDT, E. J. J.; QUINSON, M. Experimental methodologies for large-scale systems: A survey. *Parallel Processing Letters*, v. 19, n. 3, p. 399–418, 2009.

32 REKHTER, Y.; MOSKOWITZ, B.; CORP., C.; KARRENBERG, D.; NCC, R.; GROOT, G. J. de; NCC, R.; LEAR, E.; INC., S. G. *Request for Comments: 1918 - Address Allocation for Private Internets*. February 1996. Acessado em outubro de 2021. Disponível em: <https://datatracker.ietf.org/doc/html/rfc1918>.

33 SRISURESH, P.; HOLDREGE, M.; TECHNOLOGIES, L. *RFC 2663: IP Network Address Translator (NAT) Terminology and Considerations*. August 1999. Acessado em outubro de 2021. Disponível em: <https://datatracker.ietf.org/doc/html/rfc2663>.

34 VOIGT, P.; BUSSCHE, A. von dem. Practical implementation of the requirements under the gdpr. In: ____. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Cham: Springer International Publishing, 2017. p. 245–249. ISBN 978-3-319-57959-7. Disponível em: <https://doi.org/10.1007/978-3-319-57959-7_10>.

35 REPOSITóRIO Wazuh - Lista de pacotes [Online]. aug 2021. Acessado em outubro de 2021. Disponível em: <https://documentation.wazuh.com/current/installation-guide/packages-list.html#packages>.

36 MELLO, V. R. *Diamorphine*. Acessado em outubro de 2021. Disponível em: <https://github.com/m0nad/Diamorphine>.

37 WAZUH Guia de Instalação. aug 2021. Acessado em setembro de 2021. Disponível em: <https://documentation.wazuh.com/current/installation-guide/index.html>.

38 ELASTIC. *Documentation elasticsearch-certutil [Online]*. Acessado em junho de 2021. Disponível em: <https://www.elastic.co/guide/en/elasticsearch/reference/current/certutil.html>.

ANEXOS

I. INSTALAÇÃO ELASTIC STACK

A instalação dos componentes Wazuh Manager e Elastic Stack pode ser feita de forma *all-in-one*, isto é com ambos instalados na mesma instância, ou de forma distribuída, com cada componente instalado em uma instância diferente, suportando até um cenário escalável multi-nó. A documentação do Wazuh disponibiliza também dois métodos de instalação: o automatizado utilizando **scripts** ou passo a passo.

Portanto, essa instalação englobará no mesmo host o serviço do Elasticsearch, Kibana e o *plugin* do Wazuh para Kibana. Além disso, esse processo deve ser executado com o usuário root.

```
[centos@elastic-server ~]$ sudo su -
[root@elastic-server ~]#
```

Como o sistema operacional não vem por padrão com todos os pacotes necessários instalados, faremos a instalação pelo método `yum` da biblioteca `curl` para transferência de dados utilizando diversos protocolos, `unzip` para fazer operações em arquivos .zip, e `wget` para realizar downloads.

```
#  yum install curl unzip wget
```

Depois, deve-se importar a chave GPG usada para verificar a integridade dos pacotes baixados do repositório.

```
# rpm --import https://packages.wazuh.com/key/GPG-KEY-WAZUH
```

Adicionar o repositório localmente onde instalaremos os pacotes.

```
# cat > /etc/yum.repos.d/wazuh.repo << EOF
   [wazuh]
   gpgcheck=1
   gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
   enabled=1
   name=EL-\$releasever - Wazuh
   baseurl=https://packages.wazuh.com/4.x/yum/
   protect=1
   EOF
```

Agora pode-se iniciar a instalação da distribuição aberta do elasticsearch através do comando:

```
# yum -y install elasticsearch-7.10.2
```

Realizaremos a habilitação e iniciação do serviço elasticsearch:

```
# systemctl daemon-reload
# systemctl enable elasticsearch.service
# systemctl start elasticsearch.service
```

Para otimizar o uso do elasticsearch, incluiremos a fragmentação otimizada do processo de indexação e também os valores de memória heap que serão alocados pelo Elasticsearch. Ao final, reiniciaremos o serviço.

```
# sed -i 's/#bootstrap.memory_lock: true/bootstrap.memory_lock:
true/' /etc/elasticsearch/elasticsearch.yml

# mkdir -p /etc/systemd/system/elasticsearch.service.d/

# echo -e "[Service]\nLimitMEMLOCK=infinity" >
  /etc/systemd/system/elasticsearch.service.d/elasticsearch.conf

# sed -i 's/^-Xms.*/-Xms3g/;s/^-Xmx.*/-Xmx3g/'
/etc/elasticsearch/jvm.options

# systemctl daemon-reload

# systemctl restart elasticsearch
```

O segundo componente a ser instalado será o Kibana. Instalação do pacote kibana

```
# yum install -y kibana-7.10.2
```

Para que o Kibana funcione corretamente, é necessário que o diterório principal onde está instalado pertença ao usuário Kibana.

```
# mkdir /usr/share/kibana/data
# chown -R kibana:kibana /usr/share/kibana
```

Para que o Wazuh seja integrado ao Kibana, instala-se o plugin do Wazuh para Kibana a partir do repositório do Wazuh

```
# cd /usr/share/kibana/
# sudo -u kibana bin/kibana-plugin install
  https://packages.wazuh.com/4.x/ui/kibana/wazuh_kibana-4.2.2_7
  .10.2-1.zip
```

Por padrão, a interface do Kibana ouve apenas o tráfego na interface de *loopback*, endereço 127.0.0.1, então só pode ser acessado localmente. Portanto, alteraremos o arquivo de configuração do Kibana para

ser acessado a partir de qualquer IP na porta 443.

```
# cat >> /etc/kibana/kibana.yml << EOF

server.host: "0.0.0.0"
server.port: 443
EOF
```

Como conexões a portas específicas requerem privilégios root, iremos tornar o Kibana capaz de realizar conexões pela porta 443:

```
setcap 'CAP_NET_BIND_SERVICE=+eip' /usr/share/kibana/node/
bin/node
```

Para que todas essas configurações sejam aplicadas, reiniciaremos o Kibana.

```
# systemctl daemon-reload
# systemctl enable kibana.service
# systemctl start kibana.service
```

A API do Wazuh precisa de permissões explícitas para que seja acessível pelo Kibana. Portanto incluiremos suas credenciais diretamente no arquivo de configuração do Wazuh nesse servidor. Configuraremos a API no mesmo host e também na porta 55000, que é a padrão especificada na documentação.

```
# cat >> /usr/share/kibana/data/wazuh/config/wazuh.yml << EOF

  - wazuhapi:
     url: https://172.30.0.10
     port: 55000
     username: wazuhapiuser
     password: wazuhlab
EOF
```

Por último, para prevenir que a Elastic Stack faça a atualizações que conflitem ou prejudiquem a execução dos demais serviços (por exemplo do Wazuh), é necessário desabilitar o repositório que baixamos no início desse anexo.

```
# sed -i "s/^enabled=1/enabled=0/" /etc/yum.repos.d/elastic.repo
```

II. INSTALAÇÃO COMPONENTES SERVIDOR WAZUH

Esse anexo contém um passo a passo de como realizar a instalação dos componentes do Wazuh Manager, API e Filebeat no servidor Wazuh seguindo e adaptando para o projeto o guia de instalação do Wazuh [37].

O repositório do Wazuh manager deve ser primeiramente adicionado ao repositório do servidor Wazuh. Incluiremos a chave GPG para verificar a integridade do arquivo que será adicionado. Privilégios de administrador são necessários em todo esse processo de instalação.

```
[centos@wazuh-manager ~]$ sudo su -
[root@wazuh-manager ~]# cat > /etc/yum.repos.d/wazuh.repo <<\EOF

[wazuh_repo]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
enabled=1
name=Wazuh repository
baseurl=https://packages.wazuh.com/4.x/yum/
protect=1
EOF
```

Para instalar o software do Wazuh Manager e iniciar o serviço:

```
# yum -y install wazuh-manager
# systemctl start wazuh-manager
```

O gerenciador Wazuh será configurado para permitir o auto-registro de agentes Wazuh com autenticação de senha. Dessa forma, toda vez que um agente novo for detectado e estiver configurado para se conectar ao endereço IP do gerenciador Wazuh com as credenciais corretas, essa permissão será concedida.

Portanto, comando abaixo modifica a forma de autenticação padrão de não utilização de senhas para o utilizar a senha customizada (no comando, está representada por "password") que será salva no arquivo `/var/ossec/etc/authd.pass`. Nos anexos, subsequentes, veremos como os agentes são configurados para auto-registro no gerenciador Wazuh.

```
# grep "<use_password>" -B7 -A8 /var/ossec/etc/ossec.conf
# sed -i 's/<use_password>no/<use_password>yes/'
  /var/ossec/etc/ossec.conf
# grep "<use_password>" -B7 -A8 /var/ossec/etc/ossec.conf
# echo "password" > /var/ossec/etc/authd.pass
```

É necessário reiniciar o serviço do gerenciador para salvar estas configurações de autenticação.

```
[root@wazuh-manager ~]# systemctl restart wazuh-manager
```

Utiliza-se a porta padrão 1514 para ouvir os eventos que chegarão dos outros agentes no caso em que se emprega conexão segura. Para eventos de syslog, utiliza-se a porta padrão 514. Porém, ambas podem ser customizadas e alteradas no range de 1 a 65535. A porta padrão para novas conexões de auto-registro dos agentes é a 1515, utilizando TLS. Ao final da configuração, recomenda-se verificar pelo comando de verificação de estatísticas de conexões TCP/UDP `netstat` e o servidor é capaz de ouvir em ambas as portas supracitadas.

```
[root@wazuh-manager ~]# netstat -natp | egrep "(:1514|:1515)"
```

Uma nota importante é que, como este projeto utiliza a versão 4.2 do Wazuh, não são necessários passos extras para instalar a API do Wazuh neste servidor que se conectará ao Kibana instalado no servidor do ElasticStack, porque, nesta versão, a API já está inclusa no pacote de instalação do gerenciador.

Para encaminhar de forma segura os alertas e eventos recebidos pelo servidor Wazuh para o servidor do Elasticsearch, deve-se instalar a ferramenta Filebeat, que será a responsável por realizar esse envio. O primeiro passo é importar o repositório do Elastic juntamente com as chaves GPG mencionadas anteriormente e que verificar a integridade dos arquivos que estão sendo importados.

```
# rpm --import https://packages.elastic.co/GPG-KEY-elasticsearch
# cat > /etc/yum.repos.d/elastic.repo << EOF
[elasticsearch-7.x]
name=Elasticsearch repository for 7.x packages
baseurl=https://artifacts.elastic.co/packages/7.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
EOF
```

Prosseguir com a instalação do Filebeat.

```
# yum install filebeat-7.10.2
```

Felizmente, o repositório do Wazuh do Wazuh provê um arquivo de configuração do Filebeat pré-configurado para o encaminhamento de alertas para o Elasticsearch. Além disso, vamos adicionar a permissão de leitura para grupos e outras entidades ao arquivo de configuração `filebeat.yml`.

```
# curl -so /etc/filebeat/filebeat.yml
https://raw.githubusercontent.com/wazuh/wazuh/v4.2.2/extensions/
filebeat/7.x/filebeat.yml

# chmod go+r /etc/filebeat/filebeat.yml
```

Realizar a importação do módulo do Filebeat desenvolvido para o Wazuh:

```
# curl -s https://packages.wazuh.com/4.x/filebeat/wazuh-filebeat-
0.1.tar.gz | sudo tar -xvz -C /usr/share/filebeat/module
```

Iremos configurar o destino do encaminhamento de eventos do Filebeat para o endereço IP do servidor Elastic.

```
# sed -i 's/YOUR_ELASTIC_SERVER_IP/172.30.0.20/'
/etc/filebeat/filebeat.yml
```

Finalmente, podemos recarregar a árvore de dependências, habilitar e iniciar o serviço do Filebeat no servidor.

```
# systemctl daemon-reload
# systemctl enable filebeat.service
# systemctl start filebeat.service
```

Para prevenir que atualizações automáticas das ferramentas gerem algum conflito ou impeçam seu funcionamento, desabilita-se os repositórios.

```
# sed -i "s/^enabled=1/enabled=0/" /etc/yum.repos.d/wazuh.repo
# sed -i "s/^enabled=1/enabled=0/" /etc/yum.repos.d/elastic.repo
```

III. INSTALAÇÃO DO X-PACK SECURITY

Este é o procedimento de instalação de certificados TLS na comunicação entre os servidores Wazuh e Elastic e também para comunicação segura com o servidor Kibana, que é publicamente acessível da internet. Para isso, utiliza-se a ferramenta nativa do Elasticsearch para geração de certificados, a `elasticsearch-certutill`, [38].

A partir do servidor Elastic, cria-se uma pasta com as instâncias que se deseja configurar a comunicação segura:

```
[root@elastic-server ~]# cat > /usr/share/elasticsearch/
instances.yml << EOF
instances:
    - name: "wazuh-manager"
      ip:
        - "172.30.0.10"
    - name: "elasticsearch"
      ip:
        - "172.30.0.20"
    - name: "kibana"
      ip:
        - "172.30.0.20"
EOF
```

Cria-se os certificados, as chaves privadas e a autoridade certificadora (CA) para cada uma das instâncias especificadas anteriormente. Esse comando cria uma CA localmente (parâmetro `ca` e especifica que serão gerados novos certificados X.509 (parâmentro `cert`) e chaves no formato PEM em vez de PKCS12 (parâmetro `--pem`). Por fim, especificamos o arquivo de entrada `instances.yml` (parâmetro `--in`) e o arquivo de saída com as certificados solicitados `certs.zip` (parâmetro `--out`).

```
[root@elastic-server ~]# /usr/share/elasticsearch/bin/elasticsearch
-certutil cert ca --pem --in instances.yml --out certs.zip
```

Assim, no arquivo`certs.zip` encontra-se um arquivo `ca.cert` que é único e compartilhado por todas as instâncias. E dois arquivos `.cert` e `.key` que são um par único para cada instância.

Vamos extrair e descompactar o arquivo `/usr/share/elasticsearch/certs.zip` gerado.

```
# unzip /usr/share/elasticsearch/certs.zip -d /usr/share/
elasticsearch/
```

III.1 CONFIGURAÇÃO DE TLS NO ELASTICSEARCH

Cria-se um diretório `/etc/elasticsearch/certs` para armazenar os arquivos da autoridade certificados, o certificado em si e a chave no servidor Elastic.

```
# mkdir /etc/elasticsearch/certs/ca -p

# cp /usr/share/elasticsearch/ca/ca.crt
/etc/elasticsearch/certs/ca

# cp /usr/share/elasticsearch/elasticsearch/elasticsearch.crt
/etc/elasticsearch/certs

# cp /usr/share/elasticsearch/elasticsearch/elasticsearch.key
/etc/elasticsearch/certs

# chown -R elasticsearch: /etc/elasticsearch/certs
# chmod -R 770 /etc/elasticsearch/certs
```

Altera-se o arquivo de configuração `/etc/elasticsearch/elasticsearch.yml` habilitando o uso de certificados e os caminhos para cada certificado tanto na camada de transporte de comunicação entre os servidores quanto no protocolo HTTP da camada de aplicação.

```
[root@elastic-server ~]# cat >> /etc/elasticsearch/
elasticsearch.yml << EOF

# Unbind to a specific IP:
network.host: 0.0.0.0
discovery.seed_hosts: ["172.30.0.20"]

# Transport layer
xpack.security.transport.ssl.enabled: true
xpack.security.transport.ssl.verification_mode: certificate
xpack.security.transport.ssl.key: /etc/elasticsearch/certs/
elasticsearch.key
xpack.security.transport.ssl.certificate: /etc/elasticsearch/certs/
elasticsearch.crt
xpack.security.transport.ssl.certificate_authorities:
[ "/etc/elasticsearch/certs/ca/ca.crt" ]


# HTTP layer
xpack.security.http.ssl.enabled: true
xpack.security.http.ssl.verification_mode: certificate
xpack.security.http.ssl.key: /etc/elasticsearch/certs/
elasticsearch.key
xpack.security.http.ssl.certificate: /etc/elasticsearch/certs/
elasticsearch.crt
xpack.security.http.ssl.certificate_authorities:
[ "/etc/elasticsearch/certs/ca/ca.crt" ]
EOF
```

Reinicia-se o serviço para que todas essas mudanças sejam aplicadas.

```
# systemctl restart elasticsearch
```

III.2 CONFIGURAÇÃO TLS NO KIBANA

Para configurar o serviço do Kibana com a utilização de comunicação segura, segue-se os seguintes passos. Primeiro, cria-se um diretório `/etc/kibana/certs` para o qual transferiremos o certificado da CA, os certificados desse serviço e a chave.

```
# mkdir /etc/kibana/certs/ca -p
# cp /usr/share/elasticsearch/ca/ca.crt /etc/kibana/certs/ca
# cp /usr/share/elasticsearch/kibana/kibana.crt /etc/kibana/certs
# cp /usr/share/elasticsearch/kibana/kibana.key /etc/kibana/certs
# chown -R kibana: /etc/kibana/certs
# chmod -R 770 /etc/kibana/certs
```

Altera-se o arquivo de configuração do Kibana para habilitar a comunicação segura com o serviço do Elasticsearch e também para a utilização de HTTPS.

```
[root@elastic-server ~]# cat >> /etc/kibana/kibana.yml << EOF

# Elasticsearch from/to Kibana
elasticsearch.hosts: ["https://172.30.0.20:9200"]
elasticsearch.ssl.certificateAuthorities:
["/etc/kibana/certs/ca/ca.crt"]
elasticsearch.ssl.certificate: "/etc/kibana/certs/kibana.crt"
elasticsearch.ssl.key: "/etc/kibana/certs/kibana.key"

# Browser from/to Kibana
server.ssl.enabled: true
server.ssl.certificate: "/etc/kibana/certs/kibana.crt"
server.ssl.key: "/etc/kibana/certs/kibana.key"
EOF
```

Reinicia-se o serviço para que todas essas mudanças sejam aplicadas.

```
# systemctl restart kibana
```

III.3 CONFIGURAÇÃO DE TLS PARA O FILEBEAT

Como os certificados foram gerados no servidor Elastic e o Filebeat é um serviço hospedado no servidor Wazuh, deve-se copiar de um para o outro os arquivos gerados referentes à autoridade certificados, chave privado e o certificado em si.

```
[root@elastic-server ~]# scp -i /home/centos/Wazuh_Lab.pem
/usr/share/elasticsearch/ca/ca.crt /usr/share/elasticsearch/
wazuh-manager/wazuh-manager.* centos@172.30.0.10
```

Assim como nos dois serviços anteriores, iremos criar o diretório `/etc/filebeat/certs` para copiar os arquivos gerados.

```
[root@wazuh-manager ~]# mkdir /etc/filebeat/certs/ca -p
[root@wazuh-manager ~]# mv /home/centos/ca.crt /etc/filebeat/certs/
ca
[root@wazuh-manager ~]# mv /home/centos/wazuh-manager.crt
/etc/filebeat/certs
[root@wazuh-manager ~]# mv /home/centos/wazuh-manager.key
/etc/filebeat/certs
[root@wazuh-manager ~]# chmod 770 -R /etc/filebeat/certs
```

Adicionando as entradas no arquivo de configuração do filebeat `/etc/filebeat/filebeat.yml` para utilizar comunicação segura com certificados TLS.

```
[root@wazuh-manager ~]# sed -i "s#http://##g" /etc/filebeat/
filebeat.yml
[root@wazuh-manager ~]# cat >> /etc/filebeat/filebeat.yml << EOF
output.elasticsearch.protocol: https
output.elasticsearch.ssl.certificate: "/etc/filebeat/certs/
wazuh-manager.crt"
output.elasticsearch.ssl.key: "/etc/filebeat/certs/
wazuh-manager.key"
output.elasticsearch.ssl.certificate_authorities:
["/etc/filebeat/certs/ca/ca.crt"]
EOF
```

Reiniciando o serviço para as configurações serem aplicadas:

```
# systemctl restart filebeat
```

III.4 AUTENTICAÇÃO PARA O ELASTICSEARCH

Adiciona-se uma camada de autenticação entre os componentes que se comunicam com o servidor Elastic. Para isso, habilita-se o pacote do *X-Pack Security* no arquivo de configuração do elasticsearch localizado no servidor.

```
[root@elastic-server ~]# echo 'xpack.security.enabled: true' >>
/etc/elasticsearch/elasticsearch.yml
```

Para atualizar a configuração, reinicia-se o serviço.

```
[root@elastic-server ~]# systemctl restart elasticsearch
```

É necessário então gerar as credenciais para cada serviço. Após o comando abaixo, para cada entrada, configuramos as senhas que serão utilizadas para autenticação de cada componente, que foram omitidas aqui para manter o sigilo e segurança do trabalho. Depois, reinicia-se o serviço para aplicação das configurações.

```
[root@elastic-server ~]# /usr/share/elasticsearch/bin/elasticsearch-
setup-passwords interactive

[root@elastic-server ~]# systemctl restart elasticsearch
```

Com a senha configurada anteriormente no serviço do Elasticsearch, vamos adicioná-la ao serviço do Kibana pelo arquivo de configuração `/etc/kibana/kibana.yml`.

```
[root@elastic-server ~]# cat >> /etc/kibana/kibana.yml << EOF

xpack.security.enabled: true
elasticsearch.username: "elastic"
elasticsearch.password: "senha_configurada_no_passo_anterior"
EOF
```

Depois reinicia-se o serviço do Kibana.

```
[root@elastic-server ~]# systemctl restart kibana
```

A partir do servidor Wazuh, adiciona-se as credenciais configuradas ao serviço do Filebeat, no arquivo de configuração `/etc/filebeat/filebeat.yml`.

```
[root@wazuh-manager ~]# cat >> /etc/filebeat/filebeat.yml << EOF

output.elasticsearch.username: "elastic"
output.elasticsearch.password: "senha_configurada_no_passo_anterior"
EOF
```

Depois reinicia-se o serviço do Filebeat:

```
[root@wazuh-manager ~]# systemctl restart filebeat
```

IV. INSTALAÇÃO DO AGENTE DO WAZUH NO LINUX

Para realizar a instalação dos agentes Wazuh nas instâncias que possuem SO linux, deve-se possuir privilégios administrativos. É necessário adicionar também o repositório do Wazuh [35] a partir de onde faremos os downloads.

```
[centos@linux-agent ~]$ sudo su -

# cat > /etc/yum.repos.d/wazuh.repo <<\EOF

[wazuh_repo]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
enabled=1
name=Wazuh repository
baseurl=https://packages.wazuh.com/4.x/yum/
protect=1
EOF
```

Agora, vamos instalar o agente na máquina já com as configurações necessárias para que ele se conecte ao gerenciador Wazuh através do endereço IP (parâmetro WAZUH_MANAGER), realize o autoregistro usanho a senha de autenticação pré-definida (WAZUH_REGISTRATION_PASSWORD, representa nesse tutorial por "please123"), e o protocolo padrão que deve ser utilizado, que é o TCP (parãmetro WAZUH_PROTOCOL).

```
# WAZUH_MANAGER="172.30.0.10" WAZUH_REGISTRATION_PASSWORD=
"please123" \ WAZUH_PROTOCOL="tcp" yum -y install wazuh-agent
```

Reinicia-se o serviço do agente Wazuh para que as configurações sejam aplicadas.

```
# systemctl start wazuh-agent
```

Para checar que o serviço possui conexão com gerenciador Wazuh, execute o comando abaixo que buscará qual o status no arquivo que registra logs acerca do estado do serviço do agente.

```
# grep ^status /var/ossec/var/run/wazuh-agentd.state
```

Por último, para que nenhuma atualização automática seja realizada indiscriminadamente e conflite com a instalação atual, deve-se desabilitar o repositório que incluímos nessa máquina.

```
# sed -i "s/^enabled=1/enabled=0/" /etc/yum.repos.d/wazuh.repo
```

V. INSTALAÇÃO DO AGENTE DO WAZUH WINDOWS

A partir da instância Windows, é necessário ter privilégios de administrador para logar e realizar os procedimentos de instalação do agente na máquina. Portanto, o primeiro passo é executar o Windows PowerShell como administrador a partir do diretório de `Downloads`. Para isso, abra o explorador de arquivo no diretório `C:\Users\Administrator\Downloads` e na barra superior, selecione `Arquivo > Abrir o Windows PowerShell > Abrir o Windows PowerShell como administrador`, conforme a figura VI.1.

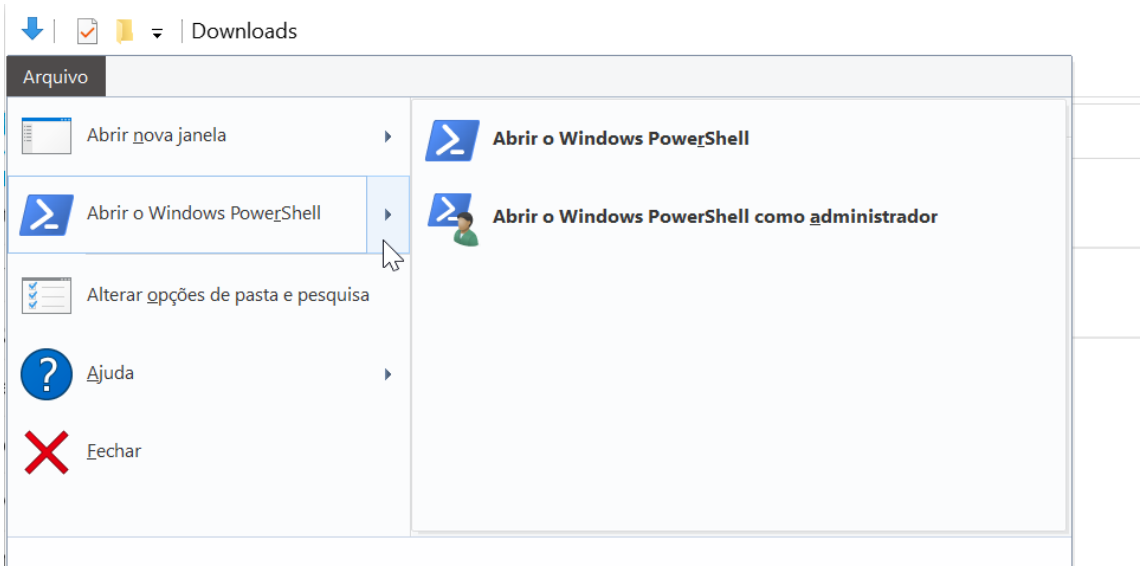

Figura VI.1: Execução powershell como administrador a partir do diretório de Downloads.

Pela console do PowerShell, vamos realizar o download do agente para o Windows no formato de instalador padrão `.msi` e em seguida realizar a instalação informando ao agente, respectivamente, qual o endereço IP do gerenciador Wazuh, qual o endereço IP do servidor que fará o registro do novo agente (que nesse caso é a mesma máquina do gerenciador Wazuh), a senha para registro que foi definida quando configuramos a autenticação de novos agentes no servidor Wazuh e, por fim, um nome para melhor identificação dos agentes na plataforma.

A senha "please123"substitui genericamente uma senha que pode ser customizada.

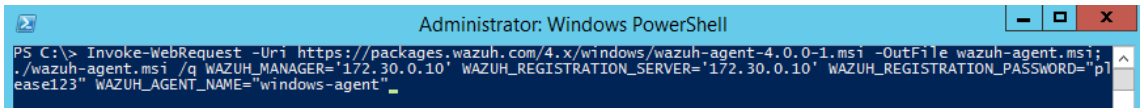

Figura VI.2: Comando para instalação de agente em SO Windows.

Para verificar se o agente está se comunicando com o servidor Wazuh corretamente, execute o agente e depois navegue até `View>View Logs`. Se houver um registro que diz:

```
wazuh-agent: INFO: (4102): Connected to the server
(172.30.0.10:1514/tcp).
```

Então está se comunicando com o gerenciador Wazuh que está instalado na máquina Wazuh Server.