DISSERTAÇÃO DE MESTRADO PROFISSIONAL

Arquitetura multiagente para detecção passiva de rootkits com enriquecimento de dados

Maickel Josué Trinks

Programa de Pós-Graduação Profissional em Engenharia Elétrica
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
FACULDADE DE TECNOLOGIA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO PROFISSIONAL

Arquitetura multiagente para detecção passiva de rootkits com enriquecimento de dados

Maickel Josué Trinks

Dissertação de Mestrado Profissional submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica

Banca Examinadora

Prof. João José Costa Gondim, Ph.D, CIC/UnB _____
Orientador

Prof. Georges Daniel Amvame Nze, Ph.D, FT/UnB _____
Examinador Interno

Prof. Dino Macedo Amaral, Ph.D, Banco do Brasil _____
Examinador externo

FICHA CATALOGRÁFICA

TRINKS, MAICKEL JOSUÉ
Arquitetura multiagente para detecção passiva de rootkits com enriquecimento de dados [Distrito Federal] 2023.
xvi, 76 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2023).
Dissertação de Mestrado Profissional - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Detecção de Rootkit
2. Enriquecimento de Dados
3. Inteligência de Ameaças
4. Cibersegurança
I. ENE/FT/UnB
II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

TRINKS, M.J. (2023). *Arquitetura multiagente para detecção passiva de rootkits com enriquecimento de dados*. Dissertação de Mestrado Profissional, Publicação PPEE.MP.046, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 76 p.

CESSÃO DE DIREITOS

AUTOR: Maickel Josué Trinks
TÍTULO: Arquitetura multiagente para detecção passiva de rootkits com enriquecimento de dados.
GRAU: Mestre em Engenharia Elétrica ANO: 2023

Maickel Josué Trinks
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

DEDICATÓRIA

Dedico esse trabalho a todos os analistas de segurança cibernética e sua missão hercúlea de proteger informações, estas cada vez mais visadas em ações maliciosas

AGRADECIMENTOS

Primeiramente, gostaria de agradecer aos professores João José Costa Gondim e Robson de Oliveira Albuquerque pelo apoio e paciência durante esse trabalho e por me apresentar novos caminhos em momentos de dificuldade.

Aos meus colegas do Programa de Pós Graduação em Engenharia Elétrica, que dividiram vários momentos dessa jornada, tanto em momentos de adversidades quanto de conquistas.

Ao Mateus Bernardo de Souza Terra, pelas valiosas dicas de funcionamento dos *rootkits* e por compartilhar códigos utilizados no sistema NERD (*Network Exfiltration Rootkit Detector*).

Aos meus pais, Rui e Nera, pelo apoio e por nunca me deixarem desistir dos estudos, fazendo o impossível para me oferecer as melhores opções de capacitação e proporcionar tempo e liberdade para estudar.

Ao meu irmão, Ismael, que sempre esteve ao meu lado incentivando nos estudos, principalmente ao decidirmos adentrar no mundo do serviço público.

À minha queria esposa, Virgínia, que embarcou nessa jornada comigo, não só em incentivo, suporte e carinho, mas também participando comigo do programa de Pós Graduação, apresentando um trabalho formidável.

Aos meus filhos, Miguel, Noah e Theo, que mesmo com pouca idade, a felicidade e incentivo demonstrados nas minhas criações e a paciência nos momentos em que demandas me fizeram ausente impulsionam a sempre tentar algo novo e nunca desistir.

Por fim, ao Programa de Pós Graduação em Engenharia Elétrica (PPEE), da Universidade de Brasília - UnB, pelo excelente programa, pelo apoio desprendido sempre que necessário e por permitir o alcance dessa meta pessoal.

RESUMO

O crescente valor agregado das informações trafegadas em meio cibernético acarretou em um cenário de sofisticação de ações maliciosas que visam a exfiltração de dados e, no atual ambiente de ameaças cibernéticos avançado e dinâmico, as organizações precisam produzir novos métodos para lidar com sua defesa cibernética. Em situações com atores maliciosos não convencionais, como *Advanced Persistent Threats* (APTs), técnicas de ofuscação de atividades danosas são utilizadas para garantir manutenção nos alvos estratégicos, evitando detecção por sistemas de defesa conhecidos e encaminhando os dados de interesse para elementos externos com o menor ruído possível. As arquiteturas MADEX e NERD propuseram soluções de análise de fluxos para detecção de *rootkits* que ocultam o tráfego de rede; entretanto, apresentam algum custo operacional, seja em volume de tráfego, seja por falta de informação agregada. Nesse sentido, esse trabalho altera e aprimora as técnicas de análise de fluxos ofuscados de forma a eliminar os impactos no tráfego rede, com enriquecimento de dados em bases locais e remotas, detecção de domínios consultados por *rootkits* e agregação de informações para geração de inteligência de ameaças, mantendo alta performance e permitindo uso concomitante com os sistemas de defesa cibernética previamente existentes. Os resultados demonstram a possibilidade de agregar informações aos fluxos de dados utilizados por *rootkits* para ações de defesa mais efetivas contra ameaças cibernéticas sem impactos significativos à infraestrutura de rede existente.

Palavras-chave: Detecção de Rootkits, Enriquecimento de Dados, Inteligência de Ameaças, Cibersegurança.

ABSTRACT

The added value of the information transmitted in a cybernetic environment has resulted in a sophisticated malicious actions scenario aimed at data exfiltration, and, in today's advanced and dynamic cyber threat environment, organizations need yeld new methods to address their cyber defenses. In situations with unconventional malicious actors, like APTs, obfuscating harmful activity techniques are used to ensure maintenance on strategic targets, avoiding detection by known defense systems and forwarding data of interest to external elements with as little noise as possible.The MADEX and NERD architectures proposed flow analysis solutions to detect rootkits that hide network traffic; however, it presents some operational cost, either in traffic volume or due to lack of aggregated information. In that regard, this work changes and improves user flow analysis techniques to eliminate impacts on network traffic, with data enrichment on local and remote bases, detection of domains consulted by rootkits and aggregation of information to generate threat intelligence, while maintaining high performance and allowing concomitant use with previously existing cyber defense systems. The results show the possibility of aggregating information to data flows used by rootkits in order to have effective cyber defense actions against cybernetic threats without major impacts on the existing network infrastructure.

Keywords: Rootkit Detection, Data Enrichment, Threat Intelligence, Cybersecurity.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ACRÔNIMOS

API	Application Programming Interface
APT	Advanced Persistent Threat
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
CISA	Cybersecurity and Infrastructure Security Agency
CPL	Current Privilege Level
CTI	Cyber Threat Intelligence
DIKI	Data Information Knowledge Intelligence
DHS	Department of Homeland Security
DML	Detection Maturity Model
DNS	Domain Name System
EAud	Elemento Auditor
EBlk	Elemento Bloqueante
ECol	Elemento Coletor
EDR	Endpoint Detection and Response
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IoC	Indicator of Compromise
IoT	Internet of Things
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
JSON	JavaScript Object Notation
MADEX	Multi-Agent Data Exfiltration Detection Architecture
NERD	Network Exfiltration Rootkit Detector
PARDED	Passive Rootkit Detector With Enriched Data
RAM	Random-access Memory
S.O	Sistema Operacional
STIX	Structured Threat Information eXpression
TCP	Transmission Control Protocol
TEE	Trusted Execution Environment
TTP	Tactics, Techniques, and Procedures
UDP	User Datagram Protocol
VMBR	Virtual Master Boot Record
XML	Extensible Markup Language

1 INTRODUÇÃO

Os últimos anos trouxeram mudanças significativas no ambiente cibernético. Além do constante aumento de usuários conectados à internet [10] e da migração de serviços para plataformas online [11], a transformação causada pela pandemia de COVID-19 forçou a adoção em larga escala de tecnologia da informação para superar as consequências causadas pela doença [12] e diminuir sua proliferação. Como exemplos, existem as diversas iniciativas de adoção de regimes de teletrabalho, ensino à distância, teleconferências e migração de serviços para plataformas digitais, entre outros.

Nesse cenário, além da tendência crescente do uso de recursos tecnológicos avançados pelos agentes de ameaças [12], a rápida adaptação imposta pela pandemia dificultou a adoção de controles de cibersegurança em locais adequados, tornando os sistemas de informação ainda mais vulneráveis a ataques cibernéticos. Morgan [13] estimou um custo global com crimes cibernéticos de 10,5 trilhões de dólares anuais em 2025, frente a 6 trilhões de dólares em 2021. O aumento de informações sensíveis em meio digital elevou também a sofisticação de técnicas maliciosas que visam a exfiltração de dados [14] e a extorsão digital, principalmente com o advento de *ransomwares*. Técnicas de *phishing* direcionado (*spearphishing*) para instalação de malwares em alvos de interesse, como os *Advanced Persistent Threats* (APTs), em parte patrocinados por estados-nação, tem sido utilizado para ciberespionagem em grandes corporações [15].

Considerando que mais de 17 milhões de novas instâncias de *malware* são registrados a cada mês [1] e a dependência e a complexidade cibernética pelas organizações é cada vez maior, torna-se mais difícil detectar todos os possíveis vetores de ataques e responder de forma adequada. As medidas de defesa tradicionais (e.g. firewall, sistemas de detecção de intrusão e auditorias de segurança) mostram-se esporadicamente inadequados na atuação contra novos tipos de ataques de rede com um grau cada vez maior de coordenação e inteligência. A inevitabilidade das vulnerabilidades e as limitações dos métodos de defesa percebidos obrigam os administradores a mudar as estratégias de detecção e inovar os mecanismos de defesa, de forma a reverter a situação passiva de propensão a ataques e de difícil efetivação na segurança cibernética [16].

1.1 MOTIVAÇÃO

No atual cenário, onde a tecnologia da informação alcançou o status de produto, ou seja, informações são facilmente valoradas e convertidas em moeda, a defesa dos dados em meio digital tornou-se um grande desafio. O *Kaspersky Global Research & Analysis Team* verificou, em 2020, que o principal alvo de Ameaças Persistentes Avançadas (APTs) foi o setor governamental [17]. Considerando que *malwares* utilizados por grupos maliciosos avançados são tipicamente desenvolvidos para manutenção da persistência no alvo, eles não apenas recebem atualizações regulares, como também evitam sua detecção limitando o volume de atividade e sua intensidade, movendo-se dentro da organização (técnica de lateralização) e encobrindo seus rastros [15].

Os *malwares* que se diferenciam por sua capacidade elevada de ocultar presença nos sistemas infectados, utilizados por diversos atores cibernéticos em ações maliciosas e com características que os tornam de grande valia para grupos APTs e operadores de *ransomware*, permitindo ao atacante acesso privilegiado em um sistema computacional, são conhecidos genericamente como *rootkits*. Quando executados corretamente, os ataques decorrentes do uso de *rootkits* são altamente eficientes e de difícil detecção.

Qualquer nó de rede físico ou virtual que utilize um sistema que permita operacionalizar ações é um alvo potencial para um *rootkit*. Considerando a expansão da Internet das Coisas (IoT), cuja segurança tanto de aplicação quanto de *endpoint* são as principais preocupações em virtude de pouca priorização de segurança e privacidade em detrimento à funcionalidades [18], acarretou no crescimento exponencial de dispositivos interconectados com potenciais vulnerabilidades, o que eleva a superfície de ataques e dificulta a detecção de ações maliciosas [19].

O número de *malwares* para os sistemas operacionais mais utilizados cresce em taxas alarmantes. A Av-Test [1] detectou mais de 70 milhões de novos *malwares* em 2021 somente para o sistema operacional Microsoft Windows, como demonstrado na figura 1.1., além de aproximadamente 1 milhão em dispositivos móveis Android e 2 milhões em sistemas baseados em Linux.

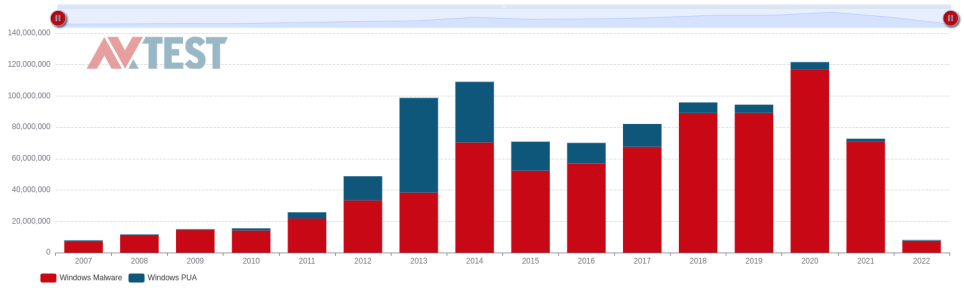


Figura 1.1: Av-Atlas: Novos Malwares detectados para MS Windows [1].

Esse cenário motivou a busca por técnicas que fossem capazes de detectar códigos maliciosos com desenvolvimento direcionado para obtenção de informações (exfiltração de dados) de alvos específicos, como o governo. Esses códigos normalmente atuam como APTs, ou seja, tem o propósito de manutenção no alvo, o que permite deduzir o uso técnicas que evitam detecção por sistemas de defesa existentes no alvo. Tais sistemas também necessitam de técnicas que enviem as informações a algum elemento externo sem muito ruído, ou seja, de forma que o tráfego de dados gerado pelo *malware* não seja facilmente percebido.

Sistemas tradicionais de defesa costumam ser executados diretamente no terminal analisado, como sistemas antivírus e EDRs, ou em pontos concentradores de rede, como IPS e Firewalls. Caso os elementos de segurança do terminal não detectem o tráfego de um código malicioso que esteja atuando de forma persistente, é necessário que os elementos da infraestrutura de rede possam analisar, classificar e, se necessário, bloquear esse tráfego. Entretanto, com as técnicas de defesa convencionais, é improvável que um tráfego ofuscado que não possua um padrão de assinatura, grandes alterações de chamadas de *Kernel* ou um destino reconhecidamente danoso, como previsto em uma comunicação realizada por APTs, seja bloqueado. Torna-se então necessário uma correlação de informações geradas no terminal com as analisadas na infraestrutura de rede para entender se esse tráfego foi legitimamente gerado pelo terminal ou por *malwares*.

O conjunto de situações descritas gerou um desdobramento em algumas questões, quais sejam:

- Como é possível detectar ofuscação de informações? E como automatizar esse processo?
- Como diferenciar informações trafegadas legitimamente de outras exfiltradas maliciosamente?
- Como gerar inteligência, ou seja, aproveitar o conhecimento obtido em uma possível detecção para aprimorar a defesa cibernética existente?

1.2 OBJETIVOS

Esse trabalho tem por objetivo principal descrever métodos que permitam detectar sistemas maliciosos persistentes que utilizam técnicas de ofuscação e gerar conhecimento através dos dados obtidos durante as detecções, de forma integrada com sistemas de defesa já existentes. De forma especifica, visa:

- estruturar uma arquitetura de detecção automática de ofuscação de tráfego por *rootkits* que possa ser utilizado em infraestruturas existentes;
- ser complementar a sistemas de defesa existentes;
- gerar inteligência através do correlacionamento e enriquecimento dos dados de detecções;
- gerar resultados de fácil interpretação e que permitem a obtenção de *insights* por analistas de tratamento de incidentes cibernéticos.

Esses objetivos resultaram em uma proposta de sistema cibernético intitulado *Passive Rootkit Detector With Enriched Data* (PARDED), uma arquitetura para detecção de *rootkits* baseados em ocultação de pacotes de rede de forma passiva e que permite alimentar outros sistemas de defesa através de centralização e enriquecimento de informações (*footprints*), fornecendo capacidade de reconhecer e atuar em tempo hábil sobre indicadores de comprometimento (IoC). A detecção ocorre de forma distribuída e não convencional, ou seja, utilizando métodos inexistentes nos sistemas de defesa comumente utilizados.

A metodologia utilizada baseou-se nos seguintes passos:

- Realizar pesquisa bibliográfica buscando referências que descrevam a utilização, aplicabilidade, funcionamento e comportamento de *rootkits*;
- Realizar pesquisa bibliográfica buscando referências que descrevam as técnicas de detecção de *rootkits* e o funcionamento das ferramentas de deecção atuais;
- Realizar pesquisa bibliográfica buscando referências que descrevam a utilização de inteligência voltada para a área de ameaças cibernéticas e métodos de enriquecimento de informações;
- Realizar testes em ferramentas de ofuscação de tráfego e de detecção dessa ofuscação e analisar seu funcionamento;

- Testar meios de detecção de tráfego, armazenamento, enriquecimento e visualização de dados;
- A partir de um modelo proposto, realizar o planejamento e o desenvolvimento dos módulos que interligam todos os objetivos previstos;
- Realizar experimentos com a arquitetura desenvolvida, analisando, analisando a viabilidade de todas as técnicas utilizadas e verificando possíveis gargalos e pontos de melhoria.

1.3 CONTRIBUIÇÕES ACADÊMICAS

O Passive Rootkit Detector With Enriched Data utiliza como base técnicas apresentadas nas arquiteturas MADEX, proposta por Marques et al. [8] e NERD, proposta por Terra e Gondim [9].

A solução proposta nesse trabalho também está presente no artigo aceito e apresentado no *IV Congreso de Ciência de la Computacion, Eletronica e Industrial* (*International Conference on Computer Science, Electronics and Industrial Engineering* - CSEI 2022), publicado pela editora Springer Nature [20], intitulado *Multi-agent Architecture for Passive Rootkit Detection with Data Enrichment*.

1.4 ORGANIZAÇÃO

A organização deste trabalho contém, além desta introdução, a seção Conceitos e Trabalhos Correlatos, que definem as expressões e os conteúdos importantes para entendimento da proposta apresentada, elenca técnicas e trabalhos prévios que visam a detecção de *rootkits* ou que apresentam inteligência de ameaças e explica sucintamente as arquiteturas utilizadas como base para o projeto; a seção Arquitetura Proposta, que descreve o sistema proposto (PARDED), demonstrando o fluxo dos dados em cada etapa de processamento, detalhando os módulos da solução e apresentando as configurações de detecção possíveis para melhor integração em infraestruturas existentes; e a seção Testes e Resultados, que apresenta dados de teste de desempenho e enriquecimento de fluxos obtidos em laboratório e demonstra os principais resultados alcançados. Por fim, a conclusão, que expõe a síntese do trabalho e possíveis caminhos futuros.

2 CONCEITOS E TRABALHOS CORRELATOS

2.1 CONCEITOS

Alguns conceitos são de grande relevância para melhor compreensão do sistema PARDED e das ações maliciosas a qual o PARDED se propõe a detectar:

2.1.1 Arquitetura multiagente

Sistema distribuído específico, cujos componentes integrantes são autônomos e autocentrados, buscando satisfazer unicamente os objetivos para o qual foram criados. Além disso, esses sistemas também se destacam por serem abertos e sem design centralizado [21, 22] e atuam como um conjunto de agentes autônomos que interagem entre si para resolver um determinado problema. Um agente, em um escopo amplo, pode ser definido como um elemento que representa e raciocina ambientes com várias propriedades diferentes, geralmente com o objetivo de tomar decisões, por exemplo, sobre a melhor forma de agir nesse ambiente. Quando há vários agentes em um mesmo ambiente, é necessário que um agente represente e raciocine também sobre os demais agentes do ambiente.

Ferber [23] define agente como um sistema computacional, posicionado em algum ambiente, que é capaz de agir com autonomia flexível visando atingir os objetivos para o qual foi projetado. Essa autonomia, possibilidade de agir sem a intervenção direta de usuários ou de outros agentes, depende de como o agente atua para atingir os seus objetivos, e envolve certas capacidades, como:

- Habilidade de perceber o seu ambiente e responder adequadamente às mudanças que ocorrem nele;
- Proatividade para o cumprimentos de seus objetivos;
- Capacidade de interagir com outros elementos e agentes.

A arquitetura multiagente se baseia na ideia de que um sistema complexo pode ser resolvido mais eficientemente por meio de múltiplos agentes separados do que por um único agente central, o que permitiria facilitar a adaptação e evolução do sistema de acordo com as condições externas. Em suma, adicionaria vantagens de escalabilidade, flexibilidade e resiliência. Entretanto, gera diversos desafios, dos quais pode-se citar:

- Comunicação: necessidade de um protocolo de comunicação eficaz que permita trocar informações e coordenar ações entre agentes;
- Coordenação: necessidade de um mecanismo de coordenação das ações dos agentes para atingir objetivos comuns;
- Autonomia: necessidade de um mecanismo eficaz de tomada de decisão com base em conhecimentos e objetivos locais;

- Escalabilidade: necessidade uma arquitetura escalável que possa lidar com um grande número de agentes;
- Heterogeneidade: necessidade de acomodar agentes com diferentes capacidades, objetivos e comportamentos;
- Segurança: necessidade de garantir segurança contra ataques, como adulteração de dados ou negação de serviço.

Em relação ao sistema proposto PARDED, a arquitetura permite compartilhar informações de pontos diferentes da infraestrutura (tanto as existentes no sistema operacional analisado quanto nos pacotes de rede transmitidos), compondo resultados para atingir um nível mais alto de precisão e confiabilidade.

2.1.2 Enriquecimento de dados

O enriquecimento de dados é um processo que envolve o aprimoramento de dados brutos com informações adicionais para melhorar sua qualidade e utilidade. Visa anexar ou aperfeiçoar dados coletados, obtidos de fontes adicionais, com contexto relevante [24], ou seja, aprimora as informações existentes, complementando os dados ausentes ou incompletos. Como em qualquer outro esforço para obtenção de dados de valor agregado, o enriquecimento de dados deve servir a um propósito do negócio [25]. Existem várias técnicas para enriquecer dados, como mineração de dados, correspondência de dados e limpeza de dados.

Normalmente, o enriquecimento de dados é obtido usando bases de dados externas. Por exemplo, um endereço IP pode ser usado para enriquecer dados de um pacote de rede, seja através de serviços de geolocalização, sistema autônomo ao qual o IP pertence, análise de serviços hospedados ou mesmo a reputação desse IP através de bases de dados de ameaças conhecidas. Os dados enriquecidos fornecem uma imagem mais completa do assunto em investigação, permitindo que os analistas identifiquem padrões, tendências e relacionamentos que podem não ser aparentes nos dados brutos. Isso, por sua vez, ajuda as organizações a otimizar suas operações, melhorar os resultados e, em contexto de segurança cibernética, agir com proatividade na detecção, mitigação e remoção de possíveis ameaças.

Se as informações existentes estiverem incorretas ou muito incompletas, pode ser impossível recorrer a uma fonte de referência para complementar as lacunas. Ademais, ao combinar dados com outras fontes, sempre existe o risco de que a correspondência não seja precisa [25] ou que existam dados conflitantes (por exemplo, a tentativa de enriquecimento obtém dados que são incompatíveis com os existentes).

Se for necessário automatizar o processo de correspondência dos dados para enriquecimento, o que é a situação mais comum no tratamento de grandes volumes de dados ou processamento em tempo real, pode ser necessário definir um sistema de preferência da informação, ou seja, em caso de conflitos, qual a informação deve ser considerada. Outra possibilidade seria a definição de níveis de confiabilidade, ou seja, o grau de assertividade de cada parcela de informação obtida de fontes externas.

2.1.3 Inteligência de Ameaças Cibernéticas

Não há uma definição amplamente aceita de Inteligência de Ameaça Cibernética (CTI) [26]. Em geral, os pesquisadores tendem a definir o termo com base em seu ambiente de trabalho e natureza empresarial, permitindo que as organizações tomem decisões de segurança mais rápidas e embasadas. Em linhas gerias, é um processo que auxilia as organizações a adquirir, processar, analisar e disseminar informações que identifiquem, rastreiem e prevejam ameaças, riscos e oportunidades dentro do domínio cibernético para oferecer cursos de ação que melhorem a tomada de decisões [27].

O termo Inteligência sugere a aplicação de conhecimento adquirido através de análise de determinadas informações, conforme apresentado no modelo *'DIKI'* (*Data-Information-Knowledge-Intelligence*) [2], onde, de forma sucinta, os dados (registros, fatos brutos coletados que não possuem significado direto) são processados, explorados, permitem avaliação, geram conclusões e norteiam uma ação. O modelo está descrito na Figura 2.1.

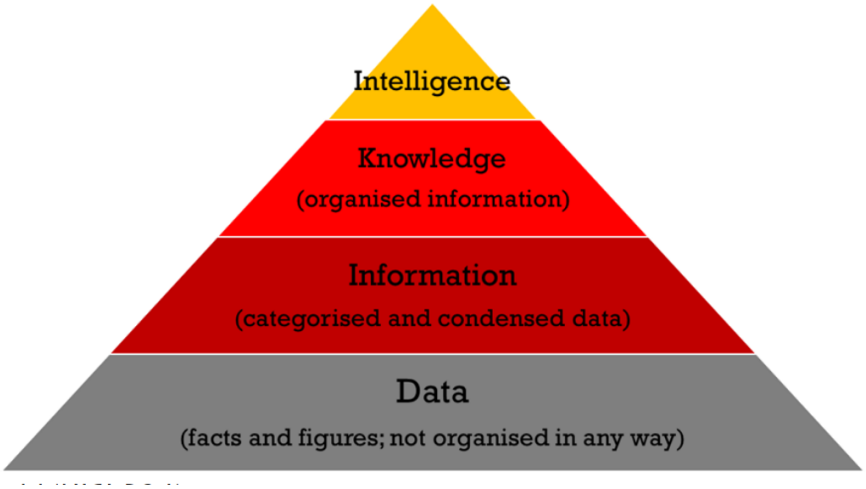


Figura 2.1: Modelo "DIKI"[2]

A Inteligência possui várias vertentes. A visão de Inteligência de Estado, conforme Lei 9.883/99 [28], é a atividade que objetiva a obtenção, análise e disseminação de conhecimentos dentro e fora do território nacional sobre fatos e situações de imediata ou potencial influência sobre o processo decisório e a ação governamental e sobre a salvaguarda e a segurança da sociedade e do Estado.

No guia de Inteligência Conjunta do exército norte-americano [3], este com foco militar, define que a inteligência não é uma ciência exata e representa a relação entre dados, informação e inteligência, conforme demonstrado na figura 2.2. De acordo com o guia, sempre há incertezas na avaliação do ambiente operacional, assim como no planejamento e execução das operações. A inteligência é gerada a partir de uma grande massa de dados (capacidade inimiga, terreno, condições climáticas, população local, entre outros) as quais darão suporte a uma estimativa preditiva da situação, bem como das capacidades e intenções do adversário.

Entretanto, a Inteligência nesse trabalho é entendida e aplicada com um viés cibernético, mais especificamente no campo de ameaças presentes no ciberespaço. Diversos autores apresentam definições de CTI, sendo algumas mais amplas e outras mais estritas. Tounsi e Rais [29], em seu trabalho de prospecção,

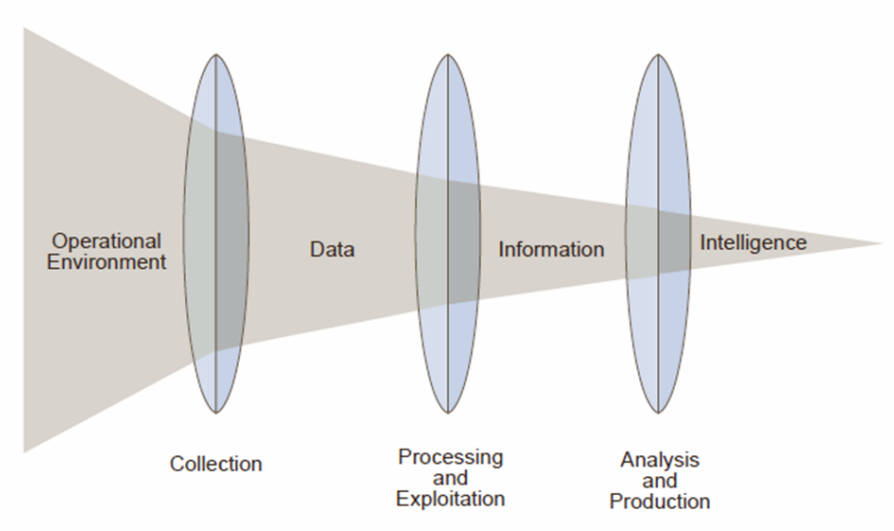


Figura 2.2: Relação entre dados, informações e inteligência [3]

elencam algumas destas definições, como:

- Qualquer conhecimento baseado em evidências sobre ameaças que pode aconselhar decisões, com o objetivo de prevenir um ataque ou encurtar a janela entre o comprometimento e a detecção;
- Uma informação que, ao invés de auxiliar decisões específicas, auxilia no esclarecimento dos riscos e das possibilidades de contorno;
- Uma informação que deve ser relevante (ou seja, potencialmente relacionada à organização e/ou objetivos), acionável (ou seja, específica o suficiente para solicitar alguma resposta, ação ou decisão) e valiosa (ou seja, as informações devem contribuir para algum resultado organizacional útil).

O melhor entendimento depende diretamente do contexto existente, o qual permite ao analista de segurança entender o tipo ou agente de ameaça com o qual está lidando, para que assim possa formular um plano de resposta apropriado. Em suma, a inteligência de ameaças inclui indicadores técnicos, contexto, mecanismos, implicações e conselhos acionáveis sobre uma ameaça existente ou emergente [29]. A consideração desses indicadores é necessária para garantir que apenas dados de ameaças relevantes sejam coletados, analisados e processados de maneira oportuna e o resultado possa produzir ações concretas e que auxiliem na tomada de decisão. No entanto, independente da escolha da definição, os métodos atuais de vincular incidentes com relatórios de CTI não são suficientemente acionáveis e exigem muito esforço do operador [30], principalmente para inteligência de ameaças cibernéticas táticas, onde são adicionados conhecimentos das Táticas, Técnicas e Procedimentos (TTPs) usadas pelos adversários, muito úteis para a resposta a incidentes porque são mais difíceis de serem alterados por um invasor do que Indicadores de Comprometimento (IoCs), os quais são alterados pelos atores maliciosos com mais facilidade.

Um dos formatos atuais para a representação de informações de inteligência de ameaças cibernéticas é o STIX - *Structured Threat Information eXpression*. Trata-se de um esforço liderado pelo Departamento de Segurança Interna (DHS) dos Estados Unidos da América. O MITRE, operando como uma parceria público-privada de pesquisa do DHS, gerenciou o *website* STIX, o envolvimento da comunidade e as listas

de discussão para permitir a colaboração aberta e pública com todas as partes interessadas. Atualmente, o STIX está na versão 2.1 é mantido pelo Comitê Técnico de CTI OASIS (OASIS CTI TC) [4]

Conforme a organização MITRE [31], o padrão STIX é um esforço colaborativo e orientado pela comunidade para definir e desenvolver uma linguagem estruturada para representar informações sobre ameaças cibernéticas. A estrutura representada pelo STIX transmite todas as informações necessárias sobre ameaças cibernéticas em potencial e visa ser expressiva, flexível, extensível, automatizável e o mais legível possível.

O STIX define diversos objetos de domínio e de relacionamentos entre eles, como padrões de ataque, campanhas, indicadores, ferramentas, vulnerabilidades, atores, entre outros, permitindo representações complexas de situações cibernéticas. Um exemplo de uma ação de ator malicioso perpetrando padrões de ataque e *malwares* está apresentado na figura 2.3.

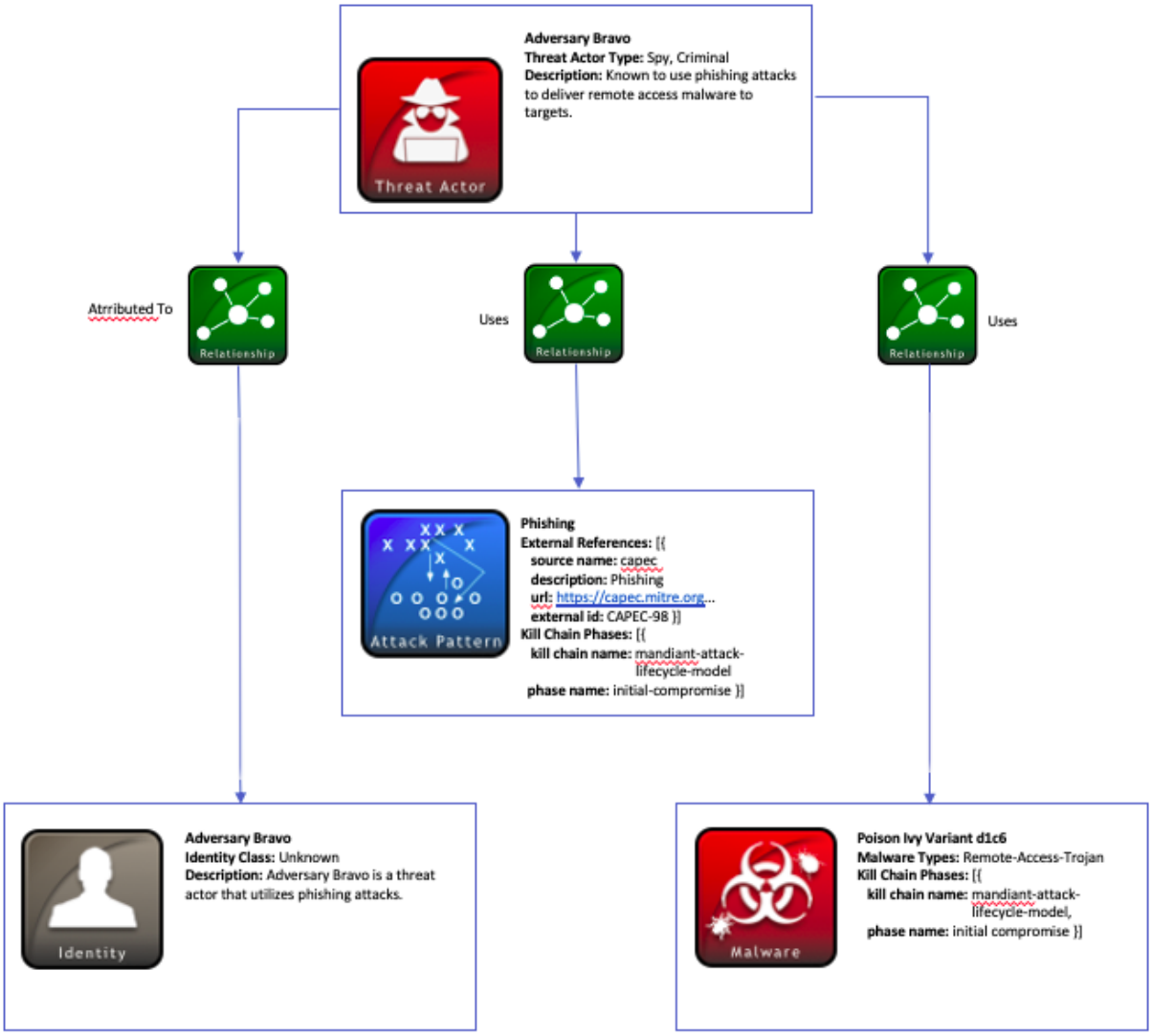


Figura 2.3: Exemplo de cenário representado na linguagem STIX [4]

A base de Técnicas e táticas adversárias que utiliza o formato STIX é o *framework* intitulado MITRE ATT&CK [32]. Esta é talvez a base de informações mais conhecida de táticas e técnicas maliciosas e é

utilizada como fundamento para o desenvolvimento de modelos e metodologias de ameaças específicas no setor privado, no governo e na comunidade de produtos e serviços de segurança cibernética.

O padrão STIX foi utilizado nesse trabalho para geração de CTI simplificado de forma automática a partir dos fluxos maliciosos que podem ser detectados pela arquitetura proposta, permitindo a integração em *frameworks* conhecidos, como o Mitre ATT&CK.

2.1.4 Modelo de Níveis de Maturidade de Detecção

Um modelo de maturidade pode ser definido como um conjunto de características, atributos, indicadores ou padrões que representam capacidade e progressão em uma determinada disciplina. O conteúdo do modelo normalmente exemplifica as melhores práticas e pode incorporar padrões ou outros códigos de prática da disciplina [33].

O *Detection Maturity Level Model* (DML), proposto por Stillions [34] é um modelo de maturidade de capacidade para referenciar a maturidade na detecção de ataques cibernéticos. Ele foi projetado para organizações que executam detecção e resposta orientadas por inteligência e que enfatizam ter um programa de detecção maduro. Esse modelo foi estendido por Bromander, Jøsang e Eian [5] e está dividido em nove níveis, divididos pela maturidade que a equipe de resposta a incidentes possui na detecção de ações maliciosas, conforme Figura 2.4.

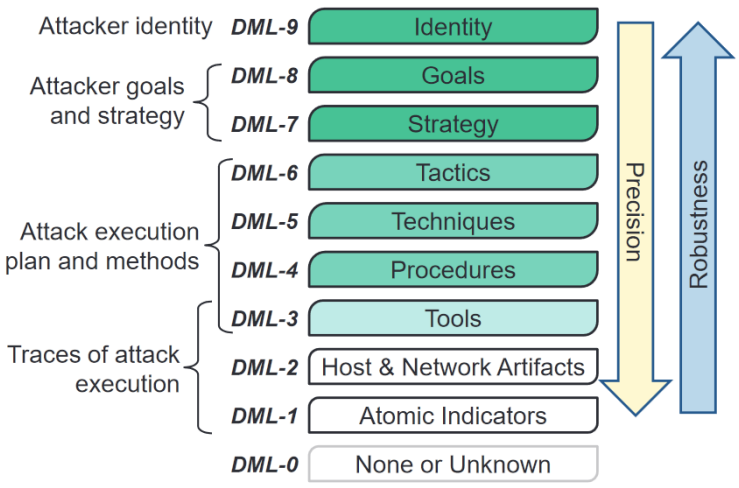


Figura 2.4: Modelo de Níveis de Maturidade de Detecção [5]

É possível verificar que o uso de regras baseadas em IoCs isolados, como artefatos de rede, fazem parte de infraestruturas com baixo nível de maturidade de detecção. No nível DML 0, a organização não possui habilidades de detecção e não tem conhecimento de ameaças de segurança. No DML 1, a organização começa a implementar controles de segurança básicos e monitora a rede para atividades suspeitas. No DML 2, a organização implementa tecnologias de detecção de ameaças mais avançadas e começa a automatizar a coleta e análise de dados de segurança.

CTI de alto nível é definido como aqueles com DML 3 ou superior, que inclui o CTI Tático (uso incorporado de TTPs - DML 4 a 6), CTI Operacional (Estratégia e Metas do Atacante - DML 7 e 8) e

CTI Estratégico (Identidade - DML 9) [28]. À medida que a organização avança nos níveis de maturidade, ela adota tecnologias mais sofisticadas, como análise comportamental de usuários e sistemas, detecção de ameaças em tempo real e *machine learning*.

A solução proposta por esse trabalho visa permitir a elevação no nível de maturidade de detecção, partindo de camadas inferiores (traços da execução do ataque), como apresentados pelas arquiteturas MADEX e NERD, para níveis superiores (Planos e métodos de execução de ataques), a medida em que é possível verificar as técnicas utilizadas pelo atacante pare resolução de nomes, ofuscação de dados, quantidade de dados enviados em cada transferência, uso de redes anonimizadas para envio de dados, entre outros, além da ligação de endereços IPs e de domínio com bases de dados de elementos maliciosos conhecidos.

Existem outros modelos de maturidade em cibersegurança, como o *Cybersecurity Capability Maturity Model* (C2M2) [33] e o *NIST Cybersecurity Framework* (CSF). Esses modelos possuem diferentes níveis de maturidade e funções/camadas, mas todos visam ajudar as organizações a melhorar sua postura de segurança cibernética, fornecendo um roteiro para alcançar um nível mais alto de maturidade na organização.

2.1.5 Rede Onion - TOR

Infraestrutura para fornecimento de conexões anônimas em rede pública. Os aplicativos fazem conexões por meio de uma sequência de máquinas chamadas de roteadores *onion* para alcançar o destino desejado.

É uma rede distribuída, baseada em camadas, projetada para anonimizar aplicativos baseados em TCP, como navegação na Web, interpretador de comandos (*shell*) seguro e mensagens instantâneas. Os clientes escolhem um caminho através da rede e constroem um circuito, no qual cada nó (ou "roteador *onion*") no caminho conhece seu predecessor e sucessor, mas nenhum outro nó no circuito. O tráfego flui pelo circuito em células de tamanho fixo, que são decifradas por uma chave simétrica em cada nó (como as camadas de uma cebola) e retransmitidas ao próximo nó da sequência [35]. Como resultado, as conexões anônimas ocultam quem está conectado a quem e com que finalidade [36]. O roteamento em uma rede *onion* é feito através do agrupamento de dados em várias camadas de criptografia compreendidas em um *pool* de nós anônimos de rede onion. Alguns nós deste *pool* são selecionados aleatoriamente para encaminhar a mensagem da origem ao destino.

O projeto TOR, rede *onion* de segunda geração, assim como em outros designs de anonimato de baixa latência, visa impedir que os invasores vinculem os pontos de origem e destino de uma comunicação, ou que vinculem várias comunicações de um único usuário (ou para um único usuário). Diversas considerações, dentro deste objetivo principal, direcionaram a evolução do TOR, como: solução implementável sem ser muito cara ou muito complexa; boa usabilidade, ou seja, sem introduzir atrasos proibitivos, exigindo o mínimo possível de decisões de configuração e sem necessidade de alteração do sistema operacional ou modificação em aplicações; protocolo flexível e bem especificado, favorecendo novas implementações; ter um design de fácil entendimento, com parâmetros de segurança bem compreendidos [35].

De acordo com a consultoria de segurança cibernética da CISA (*Cybersecurity and Infrastructure Security Agency*) [37], atores cibernéticos maliciosos usam a rede TOR para mascarar sua identidade em atividades que afetam a confidencialidade, integridade e disponibilidade dos sistemas de informação e da-

dos de uma organização. Exemplos dessa atividade incluem realizar reconhecimento, penetrar em sistemas e exfiltrar e manipular dados. O uso do TOR nesse contexto permite que os agentes de ameaças permaneçam anônimos, dificultando que os defensores e autoridades da rede realizem a recuperação do sistema e respondam a ataques cibernéticos. As organizações que não tomam medidas para bloquear ou monitorar o tráfego de rede TOR correm um risco maior de serem visadas e exploradas por agentes de ameaças que escondem sua identidade e intenções através de nós *onion*.

EM resumo, o uso de uma estrutura de nós da rede TOR permite ofuscar o caminho pelo qual a informação trafega da origem ao destino e, quando não previsto, é um indício de possível uso de comunicação para fins maliciosos. Seu monitoramento é de fundamental importância para a segurança da infraestrutura de rede, principalmente em ambientes onde o benefício de seu uso é incomum, como em redes governamentais. A solução proposta nesse trabalho correlaciona detecção de tráfego ofuscado nos terminais por *rootkits* com dados de nós pertencentes a rede TOR, elevando o nível de informação disponível aos analistas de segurança cibernética e facilitando o mapeamento de técnicas e procedimentos utilizados pelo atacante.

2.1.6 Rootkits

Consiste em um pacote de modificações que permitem a um atacante manter por tempo indeterminado controle de um sistema de informação [38]. Pode também ser definido como um software projetado para ocultar a existência de programas correlacionados, evitando inspeção e detecção e mantendo acesso privilegiado a um dispositivo de destino [39]. O termo *rootkit* é uma conexão das duas palavras: *root* e *kit*". *Root* refere-se à conta de maior privilégio em sistemas Unix e Linux, enquanto *kit* refere-se aos componentes de software que implementam a ferramenta [40]. Originalmente, era uma coleção de ferramentas que permitiam acesso em nível de administrador a um computador ou rede.

Sistemas operacionais com arquitetura x86 utilizam um modelo composto por níveis de privilégio. O privilégio de um contexto de execução é definido por uma estrutura de dados de 16 bits chamada seletor de segmento, na qual há um campo de 2 bits para representar o Nível de Privilégio Atual (CPL) [41], permitindo 4 níveis diferentes de privilégio (anéis de proteção - *Rings* 0 a 3). Por padrão, a execução de instruções privilegiadas é apenas permitido para contextos rodando com privilégio máximo (*Ring 0*). O *Ring 3* é o de menor privilégio.

Normalmente, cada sistema operacional tem ao menos dois modos: um modo de usuário (*Ring* 3) e um modo kernel (*Ring* 0). Os códigos reais são executados no modo kernel para cumprir as tarefas principais do sistema operacional, enquanto no modo de usuário, esses códigos são executados de modo indireto, por meio de uma interface de programação de aplicativo (API) [39].

Os *rootkits* que executam em nível de usuário (*Ring 3*) utilizam exploração das APIs (*API hooking*) para alterar a forma com as operações padrão do sistema operacional se comportam. O *rootkit* no nível do kernel (*Ring 0* é difícil de identificar, visto que executa comportamentos ocultos conectando à APIs nativas e é executado no mesmo nível máximo de privilégio, possuindo controle total, ou seja, no mesmo de sistemas de detecção de *malwares* e *drivers* de dispositivos [39, 42]. Camadas abaixo do sistema operacional também podem ser alvo de implantações maliciosas e serem ainda mais difíceis de detectar, como o

monitor de máquina virtual (Hypervisor) que, mesmo não sendo um dos anéis de proteção, um *hipervisor* malicioso pode ser instalado para controlar o sistema operacional em uma máquina virtual e obter acesso administrativo. É considerado como *Ring -1* [43] e *malwares* nessa camada são conhecidos como *rootkit* baseado em máquina virtual (VMBR). Uma última camada, conhecida como *Ring -3*, é a de hardware (e seu *firmware*). Uma vulnerabilidade de *firmware* pode ser usada para instalar código malicioso e efetuar um ataque ao sistema [43].

É possível diferenciar rootkits em outros níveis, além do modo de usuário e modo kernel [44]. *rootkit Hardware/Firmware* destina-se a infectar *hardware* ou *firmware*, como discos rígidos, roteadores, placas de rede e software operacional de entrada básica (BIOS) de um sistema; *rootkit de bootloader* (*bootkits*) visa a alteração do *bootloader*, ou seja, o programa/código que é executado assim que o sistema é iniciado; *rootkit* de memória esconde-se dentro da RAM (memória de acesso aleatório) do computador, o que dificulta a detecção de alteração de arquivos em disco, mas costuma ter vida útil curta em virtude de estar em área de armazenamento não permanente.

2.1.7 Visualização de dados

A visualização de dados é de extrema importância para analisar informações, principalmente em tratamentos de grandes massas de dados. Ela permite que informações sejam agrupadas, destacadas ou expressas de forma gráfica, facilitando a compreensão de conceitos difíceis ou a identificação de padrões. O principal objetivo da visualização é obter uma nova percepção (*insights*), por meio de gráficos interativos, sobre vários aspectos relacionados a algum processo de nosso interesse, como uma simulação científica ou algum processo do mundo real [45].

Williams et al. [46] define que, para a ciência da informação, a visualização é a representação visual de um espaço de domínio usando gráficos, imagens, sequências animadas e adição de sons para apresentar os dados, estrutura e comportamento dinâmico de conjuntos de dados grandes e complexos que representam sistemas, eventos, processos, objetos e conceitos.

Telea [6] descreve dois tipos de informações do que representaria um *insight* obtido por um aplicativo de visualização:

- respostas a questões concretas sobre um determinado problema;
- fatos sobre um determinado problema dos quais não tínhamos conhecimento.

Questões concretas que se reduzem a um número, como valores máximos ou mínimos, podem ser respondidas sem necessidade gráfica. Entretanto, distribuições, correlações e tendências de um conjunto de valores normalmente são melhor compreendidas quando representadas visualmente. No caso de questões qualitativas, é muito difícil obter resultados automatizados quando a definição do que se busca é muito vaga ou se há alta variabilidade dos dados de entrada. A entrada de decisão de análise humana, apoiada por visualizações interativas, torna-se indispensável [6]. Uma visão conceitual do processo de visualização é demonstrada na Figura 2.5.

Em um contexto de inteligência de ameaças cibernéticas, um dos principais desafios para o analista

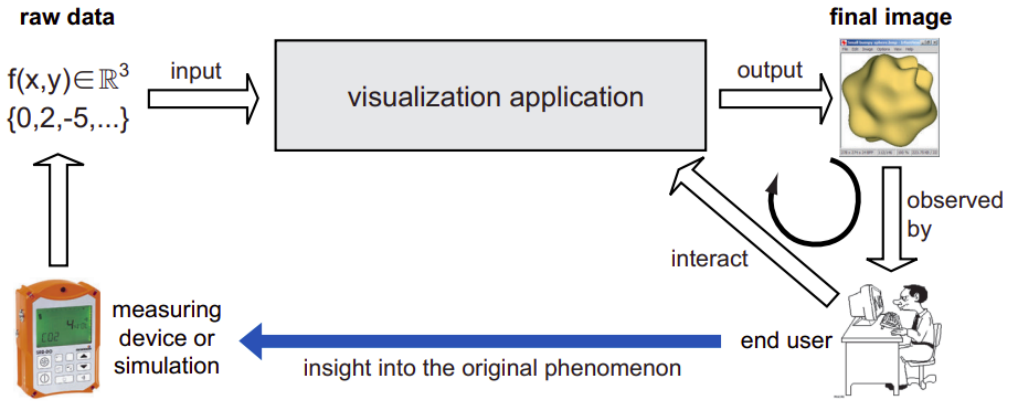


Figura 2.5: Visão conceitual do processo de visualização de dados [6]

de segurança é o grande volume de registros de CTI, gerados por fontes diversas, como antivírus, EDRs, Firewalls, IDS, entre outros, e utilizá-lo efetivamente para torná-lo uma inteligência acionável [47]. Dessa forma, a apresentação visual de CTI auxilia de várias perspectivas, com intuito de permitir a percepção de um comportamento anômalo que seria difícil ou inviável através de informações em texto, como registros de eventos em padrão XML ou syslog.

Cabe ressaltar que o resultado proveniente da visualização de dados depende de análise crítica. Dessa forma, as equipes de inteligência de ameaças cibernéticas precisam entender os dados que estão sendo visualizados e interpretá-los corretamente para obter informações úteis. Quanto melhor a interface de apresentação ao analista, mais facilmente são obtidos *insights*. Visualizações de dados imprecisas ou desatualizadas impedem a tomada de decisões bem informadas e precisas.

2.2 TRABALHOS CORRELATOS

Em resposta à crescente ameaça de *rootkits*, diversos métodos de detecção foram desenvolvidos [48, 49]. As principais técnicas consistem em detecção baseada em assinaturas (*signature-based*), baseada em comportamento ou heurísticas (*behaviour-based*), baseada em visão cruzada (*cross-view-based* ou *diff-based*), baseada em Integridade (*integrity-based*) e baseada em Hardware (*hadware-based*). Outras técnicas podem ser utilizadas [50], como detecção baseada em interceptação de chamadas de execução de código (*hooking*) ou baseada em rede (*network-based*), mas que costumam ser enquadradas dentro das cinco categorias principais.

A técnica de detecção baseada em assinatura é a estratégia utilizada com mais regularidade para detecção de *rootkits*. Esse método consiste em verificar os arquivos no disco em busca de sequências de bytes e utiliza um banco de dados de assinaturas para detectar padrões conhecidos, semelhante ao comportamento de um antivírus. O ponto fraco da detecção baseada em assinatura é que ela funciona apenas para *malwares* conhecidos. *Rootkits* novos ou suficientemente modificados, que não compartilhem frações conhecidas de código malicioso, não serão detectados [48]. Exemplos de ferramentas são o McAfee RootkitRemover [51] e o Anti-Rootkit Scanner [52], Normalmente as ferramentas de detecção por assinatura são desenvolvidas para localizar diversos tipos de *malwares*.

A detecção baseada em comportamento permite identificar novos *rootkits* que ainda não possuem assinaturas conhecidas [49], através da comparação da conduta do sistema com o que seria considerado normal. Distinções de comportamento podem ser uma característica de ações maliciosas presentes em um sistema. Uma das grandes complicações é a necessidade de conhecimento dos padrões de comportamento de um sistema não infectado para criação das linhas de base de comparação. Diferenças entre o ambiente real e o controlado podem diminuir a precisão de tais abordagens [48]. O GMER é um dos softwares *anti-rootkit* mais conhecidos e que realiza, entre outros, detecção por comportamento.

A detecção baseada em visão cruzada parte do pressuposto de que um código malicioso não é capaz de emular perfeitamente todos os aspectos de um sistema. Para detectar comprometimentos, enumeram-se os parâmetros do sistema de pelo menos duas maneiras diferentes e comparam-se os resultados, ou seja, métodos diferentes de verificar uma mesma informação devem ter o mesmo resultado. A ocorrência de diferença nos resultados retornados pelas duas abordagens indica possível presença de um *rootkit* [53]. Essa técnica é utilizada pelo Microsoft RootKitRevealer [54].

A detecção baseada em integridade compara uma fotografia do sistema analisado com uma linha de base confiável. No caso de arquivos, a linha de base confiável pode ser o valor de *hash* de um arquivo calculado em um ambiente controlado [48]. Durante uma verificação, é realizada comparação entre os *hashes* da linha de base e os *hashes* da versão atual [53]. AIDE [55] e Samhaim [56] são exemplos de sistemas de detecção com base na integridade de arquivos e diretórios.

A detecção baseada em *Hooking* é comumente utilizada para identificar interceptação de chamadas de funções, mensagens ou eventos passados entre componentes de software de forma maliciosa. *Hooking* é uma particularidade existente em sistemas operacionais e permite modificações ou melhoria do comportamento do sistema, de aplicações ou de outros componentes de software. Pode ser utilizada por *rootkits* para alterar as chamadas funções ou eventos, executando um código injetado ao invés do previsto pelo sistema,

e é tratada por alguns autores como uma técnica de detecção específica [50] ou como parte das técnicas acima descritas [48].

A detecção baseada em rede (*network-based*) é uma técnica menos usual e não é citada por alguns dos autores pesquisados, mas a percepção de seu uso por *rootkits* não é recente [50]. Utiliza o tráfego de rede gerado pelo sistema analisado para detectar possível presença de um *malware* que usa técnicas de ofuscação de rede. A constatação de tráfego de rede recebido em um elemento externo, mas que não foi detectado pelo sistema gerador, indica um provável comprometimento por *rootkit*. Isso pode ocorrer, da mesma forma, com pacotes de rede que apresentam portas de protocolo que não aparecem com abertas em um sistema, mas geram estabelecimento de conexão. Essa técnica está presente na solução proposta por esse trabalho e é apresentada na descrição da arquitetura proposta.

Inúmeros trabalhos recentes apresentam novas propostas para detecção de *rootkits*, baseadas em diversas técnicas, isoladas ou combinadas. Para analisar a relevância do tema, foi realizada pesquisa, com os termos "*rootkit detection*", em editoras de grande relevância que publicam trabalhos de viés tecnológico. A pesquisa foi feita com base nos últimos 6 anos (2017 a 2022) e abrangeu 3 bibliotecas online fornecidas pelas seguintes editoras: Springer Link [57] (filtrado apenas para artigos e conferências), IEEE Xplore [58] (filtrado apenas para Conferências e *Journals*) e ACM Digital Library [59]. O resultado está demonstrado na Figura 2.6. Além disso, consulta ao sistema Google Acadêmico [60], que permite pesquisas simultâneas em diversas editoras e publicações, demonstrou, somente em 2022, 1.360 resultados para a pesquisa com os termos "*rootkit detection techniques*".

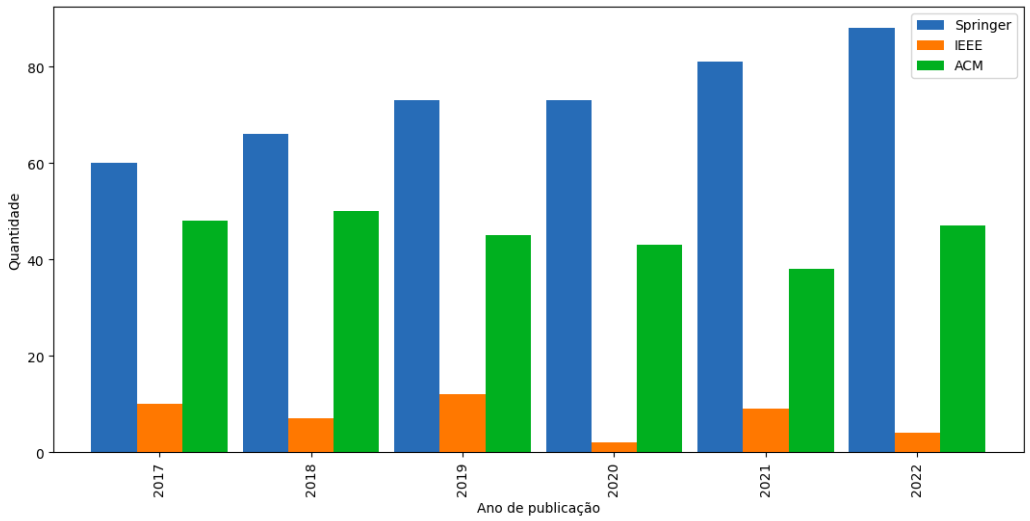


Figura 2.6: Publicações contendo os termos "*Rootkit Detection*"(figura do autor)

O número de pesquisas publicadas neste campo nos últimos anos demonstra que o tema se mantém relevante. As técnicas apresentadas nos artigos e conferências possuem grande variedade de alvos e sistemas e são, em grande parte, complementares. Dentre os pesquisados, foram descritos a seguir os que possuem ligação direta com *rootkits* e que demonstraram comparativos de ferramentas de detecção ou técnicas singulares não presentes em ferramentas conhecidas, além de *surveys* sobre *malwares*, ferramentas, e técnicas que envolvem *rootkits*.

Junnila [61] analisou a efetividade de ferramentas de detecção disponíveis para o sistema operacional

Linux. O estudo abordou uma avaliação empírica da eficácia de cinco ferramentas com recursos para detectar *rootkits* para Linux: OSSEC, AIDE, Rootkit Hunter, Chkrootkit e LKRG. A eficácia de cada ferramenta foi testada injetando 15 *rootkits* disponíveis publicamente em testes de detecção individuais em máquinas virtuais, executando a ferramenta de detecção e capturando seus resultados para análise. 75 testes de detecção foram realizados, sendo 62,7% destes não detectaram qualquer sinal de comportamento anômalo. A combinação das ferramentas teve resultado substancialmente superior, diminuindo os falsos-negativos para valores inferiores a 7%.

Wang, Zhang e Ren [62] apresentaram método para localização dos *rootkits* em máquinas virtuais de nuvens privadas, baseado em aprendizado de máquina e análise forense de memória, incluindo incluindo árvore de decisão, classificadores baseados em regras, teoria Bayesiana de decisão e máquinas de vetor de suporte (SVM). Os recursos maliciosos foram extraídos por meio do método de análise forense de memória e resultados do experimento mostraram que o classificador *Random Forest* teve o melhor desempenho, permitindo detecção de *rootkits* desconhecidos com eficácia superior a 98%.

O sistema ULTRA [63] é focado em IoT com comunicação sem fio, e utiliza técnicas de processamento de sinais para detectar e classificar comportamentos anômalos em dispositivos com arquiteturas MIPS e ARM. O funcionamento do *framework* apresentado é baseado em aprendizado de máquina, sem necessitar de agente executando no dispositivo monitorado. A abordagem proposta pelos autores alcançou resultados promissores, com alta precisão na detecção de *rootkits* conhecidos e desconhecidos durante a fase de aprendizado offline. Em testes comparativos com ferramentas de código aberto (rkhunter, chkrootkit e LKRG) o ULTRA foi capaz de detectar todos os *rootkits*, em um universo de 7 apresentados, sendo que 4 destes não foram detectados pelas demais ferramentas.

LKRDet [64], outro trabalho voltado para dispositivos IoT, verifica a consistência de eventos de hardware ocorrendo em rotinas específicas de chamada do sistema para detectar anormalidades causadas por *rootkits*. Um protótipo foi implementado para a arquitetura ARM TrustZone. A avaliação foi realizada com quatro *rootkits* populares e os detectou com precisão, sem falsos-positivos, sobrepujando resultados obtidos com técnicas de aprendizado de máquina.

Maouda [65] apresentou uma técnica para detectar *hooking* de chamadas de sistema, que implementa uma maneira diferenciada de usar eventos eBPF no *kernel*. O eBPF (*Extended Berkeley Packet Filter*) é uma tecnologia surgida no Linux que permite executar programas em área restrita (*sandbox*) em um contexto privilegiado, como o *kernel* do sistema operacional. Ele é usado para estender com segurança e eficiência os recursos do *kernel* sem a necessidade de alterar o código-fonte do *kernel* ou carregar os módulos do *kernel* [66]. Os programas eBPF são orientados a eventos e são executados quando o kernel ou um aplicativo passa por um determinado ponto de *hook*. *Hookings* predefinidos incluem chamadas de sistema, entrada/saída de função, pontos de rastreamento do *kernel*, eventos de rede e vários outros. A técnica implementada por meio do eBPF aborda meios de encontrar uma maneira de distinguir as funções internas originais (*syscalls* que são afiliadas ao núcleo do *kernel*), e um novo código de módulo do *kernel*, a qual seria uma função manipulada. Demonstrou o funcionamento com sucesso através da detecção do *rootkit* Diamorphine.

Já Nagy [48] propõe o uso do Ambiente de Execução Segura (TEE), disponível em diversas plataformas IoT, para identificar modificações feitas no código do sistema operacional, aos programas do sistema

e aos dados que influenciam o fluxo de controle (com interceptação de chamadas do sistema), além de inconsistências criadas pelo *rootkit* em certas estruturas de dados do *kernel*. A abordagem utilizada pelos autores permitiu a detecção de modificações no código do *kernel* e programas do sistema, ataques de *hooking* na memória e a presença de componentes maliciosos no armazenamento persistente do dispositivo IoT. Foram detectadas algumas limitações, como falta de suporte a processadores multi-core, a randomização de layout de espaço de endereço (utilizados em sistemas operacionais modernos) e a códigos com modificação automática no *kernel*.

Outras publicações focaram em listar ferramentas conhecidas. Entretanto, a maior parte das publicações voltadas à investigação quantitativa já estão defasadas. Arnold [50] comparou em 2011 as principais técnicas utilizadas pelos *rootkits* para sua ofuscação, atuação e persistência, além das principais técnicas de detecção. Listou os principais softwares utilizados para detecção de *rootkit*, em grande maioria empresas líderes do segmento de antivírus e ferramentas de cibersegurança, e descreveu as principais características dos *rootkits* e sistemas de detecção utilizados durante a pesquisa, com testes de desempenho e análise forense.

Uma das *surveys* mais recentes foi realizada por Talukder [67] em 2020, incluindo as voltadas para detecção de*malware*, forense de memória, análise de pacotes, varredores (*scanners*) e *sandboxes*. Conforme informado pelos autores, o artigo fornece a primeira pesquisa de grande abrangência que versa sobre técnicas e ferramentas para detectar e analisar *malware*. Diferencia, ainda, as ferramentas de análise de programas maliciosos com base em domínio e abordagem específicos. Em suma, os principais objetivos são os de auxiliar e orientar pesquisadores e analistas de *malware* na obtenção de ferramentas apropriadas para suas análises específicas de domínio.

Com foco mais amplo, Egele et al. [7] realizaram pesquisa acerca de técnicas e ferramentas automatizadas de análise dinâmica de *malware*. O foco do trabalho reside nas técnicas que podem ser aplicadas para analisar ameaças potenciais e discriminar amostras que são apenas variações de ameaças já conhecidas. Além disso, apresenta as ferramentas disponíveis à época e suas abordagens subjacentes para realizar análises dinâmicas automatizadas de softwares potencialmente maliciosos. O artigo possui número significativo de citações.

No artigo *Dynamic malware analysis in the modern era - A state of the art survey* [43] foi feita uma extensa classificação de tipos de *malwares* com base no comportamento apresentado, com base no base no privilégio em que opera e uma taxonomia de comportamento, com intuito de antecipar o comportamento que um *malware* pode apresentar, projetar o processo de análise de registro de cada operação e defender contra um ataque malicioso. Inclui ainda assuntos que, por serem mais recentes, não foram abordados no trabalho de Egele et al. [7], como o uso de tecnologias e dispositivos embarcados, descrição de análises de plataforma cruzada, criptomoedas e sua associação com *ransomwares*, forense de memória volátil, métodos de detecção de *malware* com aprendizado de máquina.

Foi possível arguir que existem trabalhos abrangendo as cinco principais técnicas de detecção citadas acima, porém não há método eficaz contra todos os tipos de *rootkits* conhecidos. Além disso, a maior parte dos trabalhos de análise possui foco direto no terminal infectado e apresenta uma ideia de projeto de detecção de *rootkits* estruturada conforme a Figura 2.7

Uma das técnicas pouco exploradas é a de detecção baseada em rede. A análise realizada por Arnold

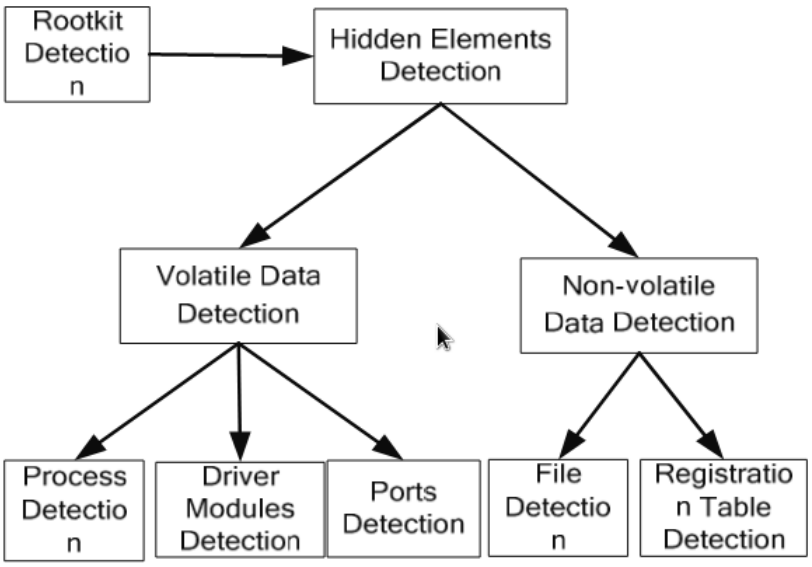


Figura 2.7: Projeto genérico de detecção de *rootkit* [7]

[50] já continha detalhes de análise de rede para detecção de *rootkits*, porém com foco ainda em alterações do sistema operacional detectáveis através de comandos simples, como verificação de portas de conexão abertas (*backdoors*) e informações constantes na tabela de conexões de rede, sem considerar a possibilidade de tais dados também estarem ofuscados.

Duas publicações que utilizam a abordagem de detecção baseada em rede, focadas justamente na detecção de códigos maliciosos que ofuscam tráfego de rede das informações disponibilizas pelos sistema operacional, são descritas com mais detalhes abaixo. As propostas, resultados e deficiências apresentados pelos autores foram utilizados como base para a arquitetura proposta nesse trabalho.

2.2.1 MADEX

Multi-Agent Data Exfiltration Detection Architecture (MADEX) é uma arquitetura multiagente cuja funcionalidade é detectar *rootkits* que executam ofuscação de tráfego através de alteração na tabela de conexões do terminal infectado. O sistema é composto por um Agente Coletor, presente no terminal analisado, o qual é responsável pela coleta de dados; e um Agente Auditor, presente na infraestrutura de rede por onde transitam os pacotes de dados do terminal analisado, responsável por detectar, filtrar e e analisar o tráfego de rede. O Agente Auditor, após receber pacotes de dados para análise, interage com o Coletor para verificar se o tráfego recebido foi percebido pelo Coletor [8].

A suspeita de ações maliciosas ofuscando tráfego é baseada, então, na existência de pacotes de dados oriundos do terminal sendo detectados pelo auditor, porém invisíveis ao coletor. Essa abordagem garante que o elemento de análise não esteja também controlado pelo *rootkit* e ofusque seus resultados. A arquitetura MADEX é demonstrada na Figura 2.8.

A detecção do tráfego pelo Agente Coletor é realizada através de consultas à tabela de conexões de rede ativas do sistema operacional. Em caso de elevado número de conexões, o número de consultas reali-

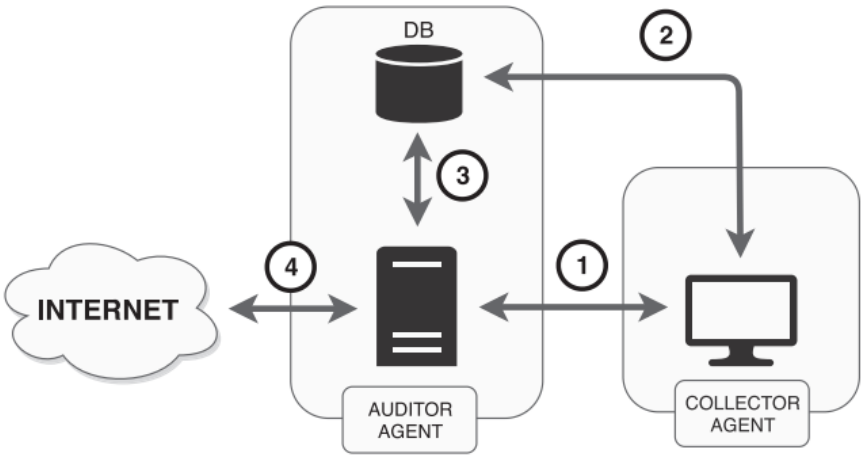


Figura 2.8: Arquitetura MADEX [8].

zadas pelo Auditor é também elevado, obrigando o Coletor a gastar muito tempo executando as consultas, aumentando assim o tempo de resposta de cada solicitação e causando mal funcionamento do sistema. Um grande número de chamadas de sistema represadas repercute diretamente na classificação do tráfego, pois as consultas eventualmente se referem a tráfegos que não existem mais na tabela de conexões. Dessa forma, todo tráfego passa a ser considerado malicioso. O autor [8] realizou experimentos com esta arquitetura para determinar a carga em que o sistema operação satisfatória e adotou como parâmetros limites de operação os valores de 135ms para tempo de resposta dos Coletores e 650 pacotes por segundo de tráfego no Auditor. O experimento demonstrou saturação com tráfegos HTTP e HTTPs, gerados em intervalos fixos de 1 segundo cada, a partir de três Elementos Coletores. Não foram apresentados resultados com geração de tráfego em intervalos menores que 1 segundo.

É importante ressaltar que a implementação foi realizada com uma configuração modesta para os dias atuais. De acordo com testes do desempenho realizados pelo software PassMark [68], o processador utilizado no experimento (Intel Core2 Duo E8400 @ 3.00GHz [8]), tanto no Auditor quanto no Coletor, o valor de referencia de desempenho - 1.175 - é de aproximadamente 50% do obtido por processadores considerados pelo Grupo PassMark [69] como *Low mid range* (valores referência de processamento inferiores a 2.495).

A falta de detalhes de implementação impede a reprodução do experimento com configurações atuais de CPU e memória; entretanto, é factível deduzir que, embora sejam esperados valores de saturação maiores que os apresentados pelo autor [8], o comportamento após a saturação seria semelhante.

2.2.2 NERD

Network Exfiltration Rootkit Detector (NERD), assim como o MADEX, é uma arquitetura multiagente com o propósito de detecção de *rootkits*. Entretanto, ao invés de utilizar a tabela de conexões do terminal para verificação de possível ofuscamento, utiliza captura de tráfego, evitando chamadas constantes ao sistema operacional [9].

O NERD mantém a proposta de um coletor, presente no terminal a ser analisado, e um elemento

auditor, presente na infraestrutura de rede, capturando externamente o tráfego gerado pelo coletor. Como no MADEX, a suspeita de ações maliciosas ofuscando tráfego também é baseada na existência de pacotes de dados oriundos do terminal sendo detectados pelo auditor, porém invisíveis ao coletor. A arquitetura NERD é demonstrada na Figura 2.9.

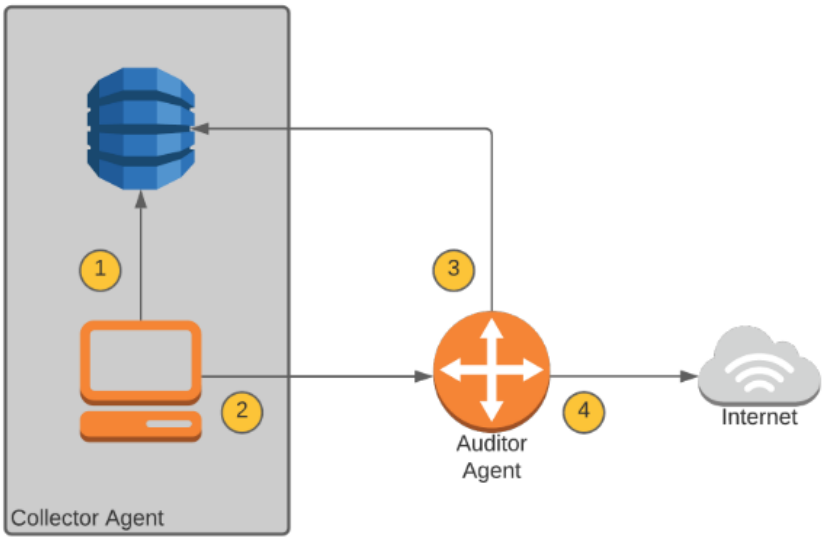


Figura 2.9: Arquitetura NERD [9].

O Elemento Coletor, na arquitetura NERD, utiliza a biblioteca `libpcap` para capturar o tráfego. Dessa forma, todos os pacotes de rede legítimos são analisados pelo coletor. A hipótese utilizada pelo autor [9] é de que os pacotes gerados por *rootkits* que ofuscam o tráfego não serão capturados, evadindo o monitoramento do Coletor. Dessa forma, quando o pacote for percebido pelo Auditor, este consultará o Coletor e receberá, como resposta, que o pacote é desconhecido, visto que não haverá informação sobre ele na base de dados do Coletor. A solução apenas analisa a percepção dos fluxos de dados nos elementos coletor e auditor, sem necessidade de realizar inspeção do conteúdo dos pacotes.

Para implementação do Agente Auditor, foi utilizado um roteador posicionado na borda da rede e, assim como no MADEX, localizado de forma *inline*, ou seja, todo o tráfego que é direcionado para destinos externos à sub-rede do terminal deve passar pelo Auditor. Este componente, portanto, concentra o tráfego e, como verificado pelo autor [9], torna-se o gargalo de desempenho da rede como um todo. O bloqueio é efetivado após uma quantidade definida de pacotes similares (limiares de detecção) serem atingidos, ou seja, é necessária a percepção de um número mínimo de pacotes suspeitos com as mesmas características, evitando assim falsos-positivos.

Para detecção de pacotes suspeitos, o Elemento Auditor deve capturar todos os pacotes que "passam"por ele. Tal captura é realizada com auxílio de um um firewall em nível de pacote em conjunto com um sistema de veredito, que informa ao firewall se o pacote analisado deve ser bloqueado. Para tal, a implementação da arquitetura NERD utilizou o programa `Iptables` e da biblioteca `libnetfilterqueue`. O endereço de destino dos pacotes é comparado com os presentes em uma base local de IPs banidos e, caso não esteja na lista, gera um consulta ao terminal que originou o pacote, onde reside o Elemento Coletor. Caso o pacote seja retornado como desconhecido e o limiar de detecção de pacotes similares for atingido, o IP correspondente ao destino é bloqueado e inserido na base de IPs banidos.

Embora tenha apresentado ganhos de capacidade em relação à arquitetura MADEX, há grande degradação na taxa de transmissão de dados, podendo, conforme o autor [9] alcançar 90% em redes de alto desempenho.

A detecção realizada pelas arquiteturas utilizadas como base desse trabalho, tanto no MADEX quanto no NERD, depende do elemento coletor não perceber seus próprios pacotes de dados, é necessária a presença de códigos maliciosos que ofusquem a comunicação. Dessa forma, caso um *rootkit* trafegue dados por canais visíveis ao Sistema Operacional, aparentará gerar somente pacotes lícitos para ambas arquiteturas e não será bloqueado. Essa característica limita a quantidade de programas maliciosos verificados, mas adiciona um método de detecção inexistente na descrição dos sistemas de defesa mais populares [50] [67].

2.3 LACUNAS DE PESQUISA E IMPLEMENTAÇÃO

As pesquisas realizadas e explanadas nas seções anteriores, tanto relacionadas a conceitos quanto a publicações no tema de interesse, demonstraram um lacuna de pesquisa e desenvolvimento no campo de detecção de ofuscação de tráfego em terminais comprometidos.

No campo conceitual, a detecção baseada em rede não é descrita como uma possível técnica por parte dos autores. Nos trabalhos correlatos, apenas dois projetos foram encontrados que utilizam a detecção em rede de tráfego ofuscado para o terminal infectado. A detecção, nos demais trabalhos, direcionam esforço para agentes instalados no próprio terminal, executando técnicas de visão cruzada, comportamento, integridade do sistema ou assinaturas. Nos trabalhos que possuem foco no tráfego de rede, a detecção é majoritariamente realizada com base em padrões de assinatura, destino conhecidamente comprometido ou tráfego anômalo em relação ao padrão de comunicação esperado.

As publicações que utilizam a mesma técnica descrita nesse trabalho, MADEX [8] e NERD [9], demonstram que estratégia é viável e capaz de detectar tráfegos maliciosos com grande assertividade. Essa estratégia foi a única encontrada que independe, indiretamente, do correto funcionamento de agente instalado no terminal em análise. A independência é considerada indireta pois, mesmo sendo necessário a presença de um elemento de software da solução no terminal analisado (definido nas publicações estudadas e na proposta apresentada nesse trabalho como Agente Coletor), ele não pode ser desabilitado ou ter sua ação impedida por um elemento malicioso sem que tal ação seja detectada pelo demais elementos do sistema.

Entretanto, as implementações realizadas pelas publicações que descrevem o MADEX e o NERD causam grande impacto negativo no desempenho da rede, muito em virtude do processamento utilizado para cada pacote transmitido, dificultando ou inviabilizando seu uso em infraestruturas de rede de comunicação existentes. Dependem também de instalação considerada agressiva, ou seja, adição de um elemento *inline* ativo entre os terminais analisados e o destino da comunicação, assim como ocorre em um IPS, acarretando em um novo ponto de rede passível de falha. Por fim, não apresentam integração com os demais sistemas de detecção de ameaças cibernéticas existentes, enriquecimento dos dados obtidos e possibilidade de geração de *insights* pelos analistas de segurança, a qual necessita de uma interface de visualização dos dados amigável e com informações filtradas, completas e intuitivas.

2.4 SÍNTESE

A arquitetura proposta nesse trabalho tem o intuito de remover ou reduzir o hiato de pesquisa e implementação identificado e explanado acima. Primeiramente, adicionando uma nova solução de detecção de ações maliciosas de ofuscação baseada em rede, além de comprovar a eficácia de sistemas passivos na detecção de *rootkits* que ofuscam tráfego de rede.

Esse trabalho visa contribuir, ainda, com uma estratégia de detecção que não impacte significativamente no desempenho da rede analisada e que possa ser utilizada por analistas de segurança em níveis mais altos de maturidade de detecção de ameaças cibernéticas, provendo interface de visualização intuitiva, enriquecimento de dados e modelos de regras de bloqueio para sistemas como IDS e IPS, além da geração simplificada de modelos de inteligência de ameaças cibernéticas.

O próximo capítulo apresenta com detalhes a arquitetura proposta, ilustrando a estrutura da solução, com a descrição dos módulos, sistemas integrantes e bases de dados utilizadas.

3 ARQUITETURA PROPOSTA

A arquitetura proposta do *Passive Rootkit Detector With Enriched Data* (PARDED) deriva de uma base multiagente, composta por elementos que trabalham em pontos distintos de uma infraestrutura de rede de comunicação. Implementa alterações conceituais previstas nas arquiteturas MADEX [8] e NERD [9], principalmente na forma como o agente auditor atua para detecção e bloqueio da comunicação maliciosa, de forma passiva, a fim de evitar degradação na taxa de transmissão dos equipamentos de rede e prover integração com outros sistemas de defesa contra *malwares*.

O PARDED possui como principal característica o aperfeiçoamento das técnicas de bloqueio em infraestruturas de redes de comunicação sem degradar a taxa de transmissão de dados nos equipamentos de rede, além de prover uma interface para integração com outros sistemas de defesa previamente existentes, como *firewalls* e sistemas de detecção de intrusão (IDS). Além disso, outra inovação importante é a possibilidade de enriquecimento das informações acerca dos destinos maliciosos utilizados pelos *rootkits*, notadamente para sistemas de comando e controle (C2), através de bases de informação locais ou externas. O processo de enriquecimento auxilia, por exemplo, na detecção dos nomes de domínio utilizados, o que permite análises de inteligência posteriores com maior riqueza de detalhes, mantendo o contexto do objeto da avaliação, evitando enriquecimentos desnecessários e que poluem os ambientes de análise.

Para tal, o elemento auditor, que possui a função de comparar o tráfego capturado na rede com o percebido pelo coletor, foi dividido em três componentes, sendo o primeiro composto de um elemento *inline*, o qual possui a função de espelhar o tráfego para análise do segundo componente, *out-of-band*, que recebe a cópia do tráfego e compara com os dados do agente coletor para tomada de decisão, de forma análoga à arquitetura NERD. O terceiro elemento, *inline*, recebe a informação do segundo elemento acerca das conexões consideradas maliciosas e que devam ser bloqueadas.

A alteração no método de captura, análise e bloqueio visa eliminar o gargalo de desempenho causado pelas características do Agente Auditor ativo existente nas demais arquiteturas, permitindo o tratamento dos pacotes, o armazenamento e o enriquecimento das informações sem afetar o tempo de resposta esperado pela comunicação.

A detecção do tráfego suspeito é realizada através da comparação entre o tráfego observado no terminal, obtido pelo Elemento Coletor (ECol) através da captura de fluxos na interface de rede, e o tráfego recebido pelo Elemento Auditor (EAud). Pacotes copiados e recebidos pelo EAud, mas que não foram observados pelo ECol, podem indicar evasão de captura de tráfego no terminal por ação maliciosa de um *rootkit*.

Diferente das arquiteturas MADEX e NERD, a detecção é feita de forma passiva e a ação a ser tomada é definida em conjunto com informações de outros Elementos Coletores, de resolução de domínio e de bases de dados locais e externas, executada e analisada para cada fluxo potencialmente malicioso.

Após a coleta do tráfego suspeito, este é enriquecido com informações provenientes de bases de dados externas, que podem ser locais (*offline*) ou remotas (*online*). Caso, após o enriquecimento, algum fluxo seja considerado malicioso, o sistema gera entradas para sistemas de defesa (representado pelo Elemento Bloqueador na Figura 3.2 ou para sistemas de CTI. Caso configurado, os sistemas de CTI podem reali-

mentar o PARDED, da mesma forma que ocorre com bases externas. A Figura 3.1 demonstra o fluxo de informações de entrada e saída após a captura de um fluxo suspeito, representando o enriquecimento e a alimentação de sistemas de defesa ou de CTI.

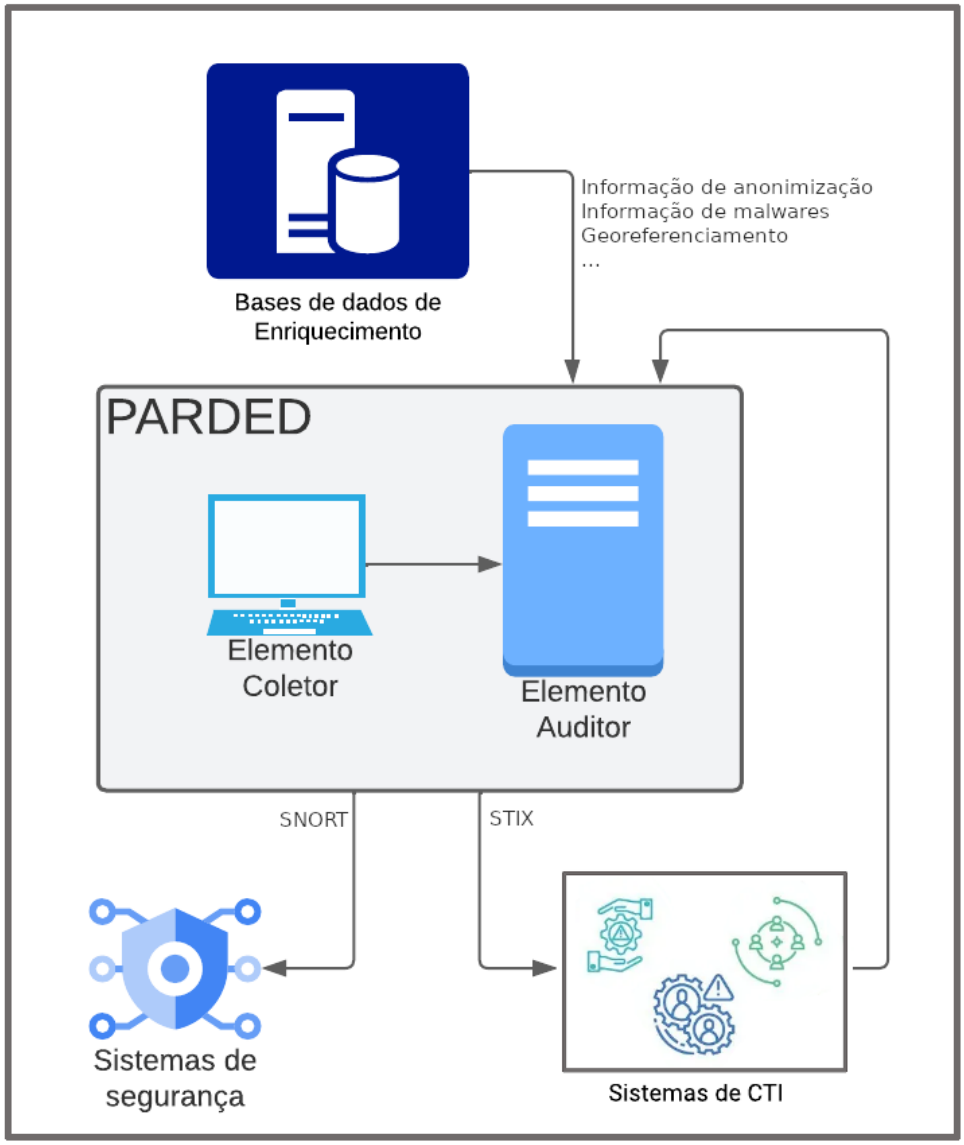


Figura 3.1: Entradas e Saídas relacionadas a fluxos suspeitos na arquitetura proposta (Figura do Autor)

O enriquecimento dos dados coletados possibilita análises de inteligência sobre o comportamento de *rootkits* na rede, seja pela integração de dados de múltiplos terminais, histórico de informações e detecção de domínios maliciosos utilizados, seja pela agregação de novas verificações com auxílio de bases de dados externas. A criação de base de dados com características de comportamento malicioso permite ainda alimentar demais sistemas de defesa da rede, como sistemas que utilizem regras de bloqueio por IP ou domínio. Essa características são necessárias para elevar o nível de maturidade da detecção de ataques pelas equipes de resposta a incidentes.

No modelo DML [34, 5], as arquiteturas utilizadas como base desse trabalho apresentam resultados esperados para a camada DML-2, ou seja, retornam informação que pode ser coletada pela rede e sensores

de ponto final. Nesse nível, considerando links de alta capacidade, a quantidade de informações coletadas pode ser esmagadora e requer boas ferramentas analíticas para analisar e entender o ataque em níveis mais altos de abstração. O PARDED permite trabalhar nas camadas posteriores (método e planejamento de execução de ataques) e pode auxiliar na descoberta de TTPS, como ciclos de alterações de IPs e domínios, padrões de comandos remotos solicitados ao terminal infectado e, em caso de pacotes não cifrados, o conteúdo enviado pelo *malware* para os sistemas de comando e controle.

A Figura 3.2 ilustra a arquitetura proposta e os tópicos seguintes explicam com mais detalhes cada um dos elementos que a compõe.

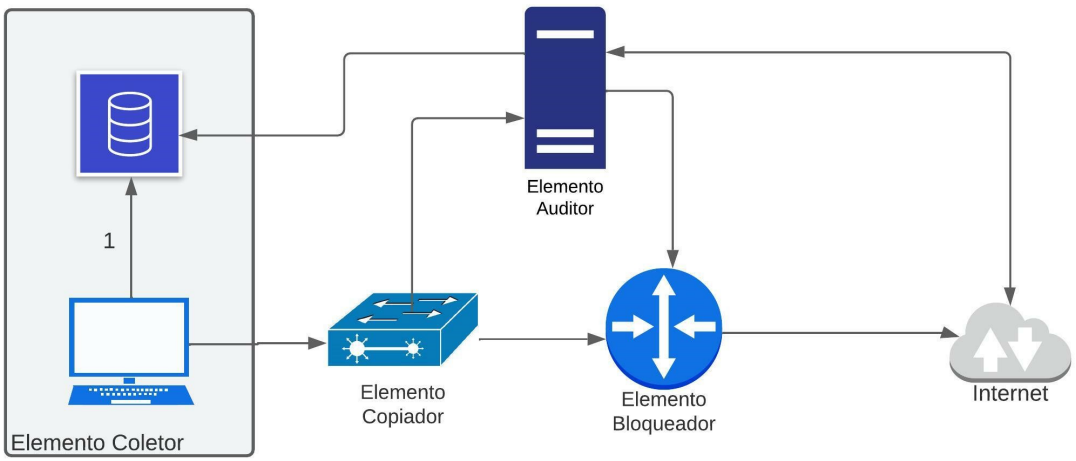


Figura 3.2: Arquitetura PARDED proposta (Figura do Autor)

3.1 DESCRIÇÃO DOS MÓDULOS DA ESTRUTURA

3.1.1 Elemento Coletor (ECol)

Trabalha de forma passiva, através da cópia do tráfego pelo próprio terminal a ser analisado. Em virtude dos resultados apresentados, optou-se por utilizar a abordagem prevista na arquitetura NERD, sem alterações, onde o agente coletor realiza a cópia do tráfego através de *sockets* vinculados ao dispositivo de rede utilizado por meio da biblioteca `libpcap`. A diferença apresentada pelo PARDED é o uso de *port miroring* para alimentar o Elemento Auditor, assim como em um sistema de detecção de intrusão (IDS) [15] . Essa estratégia permite que o auditor processe as informações passivamente e não ocorra interferências no desempenho da rede. O processo de execução do Sistema de Análise Inicial, ao receber um pacote de dados ou uma consulta do EAud, é demonstrado na Figura 3.3.

Os pacotes de dados, ao serem transmitidos pelo terminal analisado, são detectados pelo ECol e armazenados em uma base, intitulada Base de Fluxos. No caso de *rootkits* que ofuscam o tráfego de rede, os pacotes ofuscados não são identificados pela biblioteca `libpcap` e, portanto, não serão adicionados à Base de Fluxos, permitindo que o elemento exterior ao terminal analisado (EAud) possa consultar essa informação e detectar a anomalia.

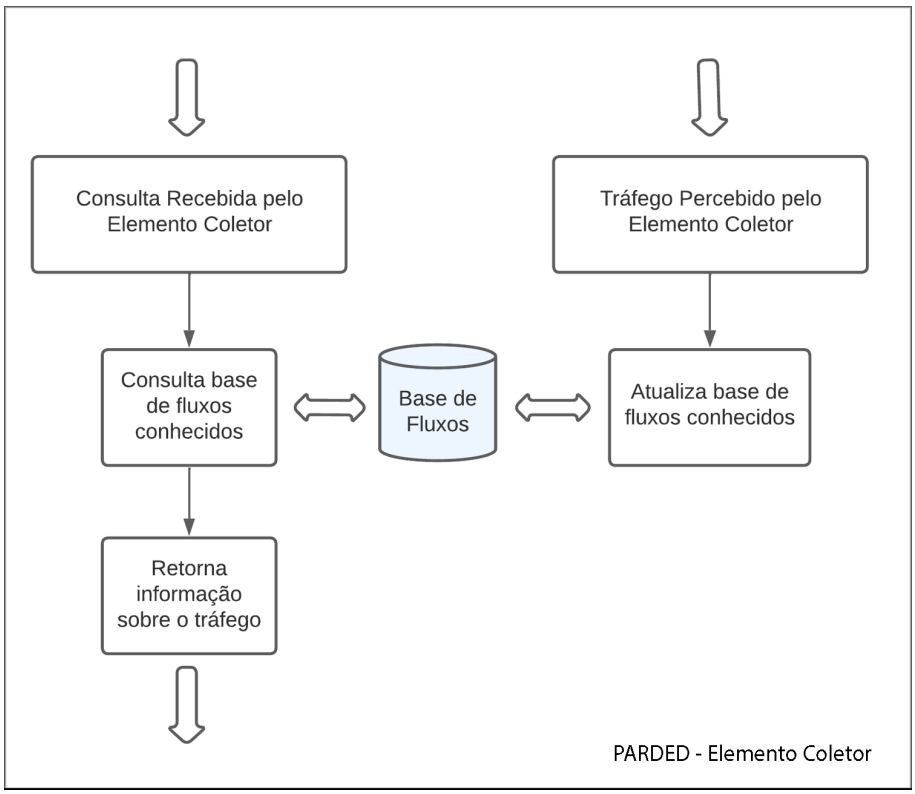


Figura 3.3: Fluxo de funcionamento do ECol (Figura do Autor)

Para garantir que todo fluxo seja detectado e armazenado individualmente na base da dados, cada entrada deve possuir as informações mínimas necessárias que diferenciam um fluxo de dados de outro, quais sejam: protocolo, endereço IP de origem e destino e porta de origem e destino. Esses dados são concatenados e formam uma chave de pesquisa que será armazenada na Base de Fluxos.

```
1      chave = IP_destino + Porta_destino + IP_Origem + Porta_Origem + Protocolo
```

3.1.2 Elemento Auditor (EAud)

Diferente das arquiteturas MADEX e NERD, o EAud não executa internamente a função de bloqueio, permitindo que trabalhe *out-of-band*. Dessa forma, análises mais complexas podem ser realizadas antes do bloqueio sem impacto de desempenho na comunicação do terminal.

O enriquecimento das informações é realizado inicialmente com a coleta e armazenamento das consultas do sistema de nomes de domínio (DNS) recebidas pelo Auditor. Dessa forma, caso a comunicação maliciosa seja precedida por uma solicitação DNS, o domínio consultado pelo *malware* é detectado e adicionado às informações a serem enriquecidas. Em seguida, o EAud realiza consultas ao ECol e em outros sistemas, locais ou externos, que permitem detectar atividades anômalas ou maliciosas.

Para que o EAud considere o fluxo como malicioso, alguns limiares de detecção devem ser atingidos. Os parâmetros iniciais de cada limiar foram definidos com base nas hipóteses abaixo:

- 6 fluxos não detectados pelo mesmo ECol (mesmo terminal) é considerada uma ação maliciosa. Essa premissa baseia-se nos excelentes resultados de falsos-positivos obtidos pela arquitetura NERD com essa configuração [9].
- 2 Terminais cujos coletores não detectem 2 fluxos para um mesmo destino são suficientes para que os pacotes sejam considerados maliciosos. Essa premissa também se baseia nos resultados obtidos pelo NERD, adicionada a improbabilidade de que falsos-positivos, já raros nessa arquitetura, ocorram em coletores diferentes para um mesmo IP de destino.
- 2 fluxos não detectados para um mesmo destino, nos seguintes casos: o destino seja uma rede TOR; o domínio pesquisado seja considerado malicioso por 2 empresas de segurança; o IP seja considerado malicioso por 4 empresas de segurança. Esses parâmetros dependem do conhecimento da infraestrutura de rede na qual o sistema está implantado. Por exemplo, caso a infraestrutura de rede permita ou preveja acesso a nós da rede TOR pelos usuários, valores limiares baixos podem elevar o número de falsos-positivos.

Para facilitar a evolução do Eaud e impedir que falhas em alguma etapa interrompam o funcionamento das demais etapas, as funcionalidades foram separadas de forma modular, conforme Figura 3.4. Dessa forma, o sistema torna-se generalizável e adaptável, além de facilitar atualizações.

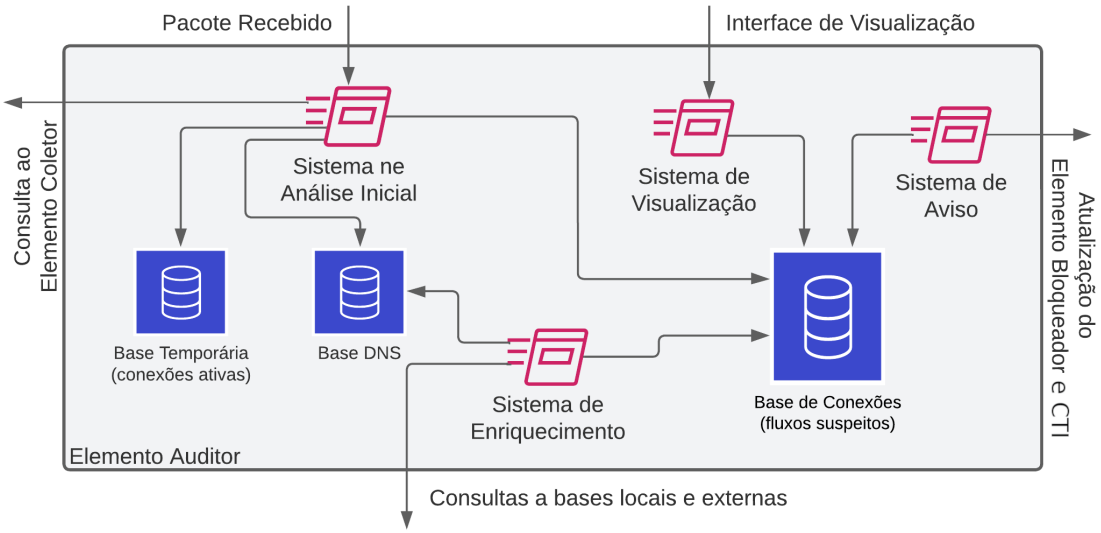


Figura 3.4: Elemento Auditor da estrutura PARDED (Figura do Autor)

Para entendimento dos sistemas que compõem o Elemento Auditor, é necessário entender as bases de dados necessárias para armazenamento dos fluxos e de informações adicionais, como dados de domínios e enriquecimento.

3.1.3 Bases de Dados do Elemento Auditor

3.1.3.1 Base Temporária

Para que o EAud possa diferenciar os fluxos de dados recebidos como legítimos ou suspeitos, é realiza uma consulta ao ECol para verificar se o fluxo foi detectado no terminal que supostamente gerou a transmissão dos dados, conforme descrito na seção Estrutura da arquitetura proposta. Se o fluxo foi detectado, é considerado legítimo. Entretanto, essa consulta é lenta, pois depende do estabelecimento de conexão TCP entre dois elementos de rede e, para garantir a eficiência da solução, deve ser evitada sempre que possível.

Portanto, após a consulta feita ao ECol, pelo EAud, de um pacote de dados considerado legítimo, os demais pacotes da mesma conexão podem também ser considerados legítimos, pois o EAud já sabe que o ECol detectou essa informação e a consulta retornará o mesmo resultado. Dessa forma, para evitar que o EAud consulte o ECol nos pacotes onde o resultado da consulta é conhecido previamente, o EAud adiciona a informação do fluxo já consultado em uma tabela de conexões ativas, chamada de Base Temporária. Assim, antes de realizar consultas ao ECol, é feita consulta prévia à Base Temporária para cada pacote recebido. Somente em caso de inexistência de informação na Base Temporária é que a consulta é feita ao ECol.

Nos casos em que é necessária a consulta ao ECol (inexistência de informação na Base Temporária), ela é feita através de uma chave de pesquisa, gerada da mesma forma em que os dados são armazenados na Base de Fluxos do ECol, concatenando os dados de protocolo, endereço IP de origem e destino e porta de origem e destino do pacote identificado. O Ecol retornará positivo se a chave de pesquisa é conhecida, e negativo caso contrário.

De forma a tornar as consultas mais rápidas, a base de dados é armazenada em memória e acessada diretamente pelo Sistema de Análise Inicial, que será descrito abaixo. Essa base de dados foi chamada de Base Temporária por ser volátil e cada chave ser armazenada por pouco tempo, ou seja, os dados são eliminados ao desligar o EAud, ao atingir o tamanho máximo de chaves de pesquisa armazenadas (eliminação do dado mais antigo) ou quando não há mais motivos para o armazenamento da chave de pesquisa (recebimento de pacote TCP FIN, *timeout* de conexão, entre outros).

3.1.3.2 Base de Conexões

A Base de Conexões armazena os fluxos que são considerados suspeitos, ou seja, quando o ECol informa que não detectou a informação solicitada pelo EAud. Essa base está estruturada em um banco de dados relacional, onde cada fluxo suspeito é adicionado em uma tabela de fluxos para posterior enriquecimento e análise. A estrutura da tabela de fluxos está demonstrada na figura 3.5.

Os fluxos são diferenciados da mesma forma que é feito na Base Temporária, através da chave de pesquisa. Além disso, são armazenadas informações separadas dos IPs de origem e destino (para facilitar por exemplo, a visualização do analista de segurança e realizar pesquisas de enriquecimento, além da quantidade de fluxos com as mesmas características (mesma chave de pesquisa), necessário para verificar os limiares de detecção maliciosa, informações de bloqueio (quando os limiares de detecção foram atingidos e regras de bloqueio e CTI foram geradas), data de detecção e detalhes de enriquecimento das bases locais

Coluna	Tipo	Não nulo	Padrão	Restrições
id	integer	NOT NULL	nextval('"fluxosDetectados_id_seq"'::regclass)	
ipOrigem	inet			
ipDestino	inet			
enriqVT	boolean	NOT NULL	false	
enriqTOR	boolean	NOT NULL	false	
quantidade	smallint	NOT NULL	0	
bloqueado	boolean	NOT NULL	false	
chave	character varying(30)			
data	timestamp without time zone			

Figura 3.5: Tabela de fluxos da Base de Conexões (Figura do Autor)

e remotas. Para cada entrada da tabela, é adicionada uma chave primária - id (exclusiva de cada chave de pesquisa) par facilitar a relação com outras tabelas.

A Base de Conexões possui ainda tabelas relativas a cada base externa (local ou remota). Nessas tabelas, são necessárias algumas informações mínimas para correto funcionamento do sistema. A primeira e mais importante é a vinculação do enriquecimento realizado com o fluxo correspondente, realizada através de uma chave estrangeira, que aponta para a chave primária da tabela de fluxos. Um exemplo de tabela de base externa é apresentada na Figura 3.6. Cada entrada da tabela de fluxos pode ter mais de um enriquecimento, em uma ou mais tabeas de bases externas. ainda, são armazenadas informações especificas do enriquecimento (as quais dependem do tipo de base consultada - no exemplo apresentado, são armazenadas a quantidade de detecções e se a informação é referente ao enriquecimento de IP ou de domínio), além da data da pesquisa.

Coluna	Tipo	Não nulo	Padrão	Restrições
fk_id	integer	NOT NULL	nextval('"enriquecimentoVT_id_seq"'::regclass)	
deteccao	smallint		0	
data	timestamp without time zone			
ip	inet			
dominio	character varying(254)			

Figura 3.6: Tabela de enriquecimento da Base de Conexões (Figura do Autor)

3.1.3.3 Base DNS

Quando um pacote de rede é detectado pelo EAud, é possível obter as informações existentes nesse pacote, como o IP de destino. Se o fluxo é considerado suspeito, torna-se importante ao analista de segurança entender o que está acontecendo na infraestrutura de rede. Entretanto, torna-se inviável ao analista verificar o destino das comunicações se o IP de destino for constantemente alterado. Ainda, de forma a manter uma comunicação permanente, os atores maliciosos dificilmente utilizarão um *malware* que direciona os dados para um único IP, pois inviabilizaria alterações nas estruturas de destino dos dados, como um servidor de comando e controle (C2). Portanto, é de especial valia ao analista entender se o IP de destino foi resultado de uma consulta de domínio e qual seria esse domínio. Essa afirmação torna-se fundamental se o IP de destino apontar para um servidor de serviço compartilhado, ou seja, que concentra diversas aplicações de

diversos clientes, como um serviço armazenado em nuvem pública.

Assim, o EAud, ao detectar uma consulta DNS, armazena os dados da resposta (*DNS query response*) em uma base de dados intitulada Base DNS, permitindo que, ao sinalizar um fluxo como suspeito, ser possível verificar se o destino possui um domínio específico anteriormente consultado. A Base DNS permite ainda entender se fluxos com IPs de destino diferentes foram gerados com base em um mesmo domínio. O Código 3.1 apresenta a estrutura da Base DNS.

Código 3.1: Base DNS do Elemento Auditor

```
1     typedef struct {
2         unsigned char nome_do_domino[TAM_DNS_MAX];
3         __be32 endereco_ip
4     } tuple_dns;
5     static tuple_dns lista_dns[TAM_BASEDNS_MAX];
```

Cada elemento constante da lista_dns possui um par de dados contendo o domínio e o IP correspondente. Se a consulta gerar mais de um IP, são armazenados novos elementos para cada. Essa abordagem simplifica o armazenamento e a consulta posterior. A Base DNS foi feita em linguagem C e, assim como a Base Temporária, é mantida em memória para melhor desempenho. Caso um fluxo seja marcado como suspeito, o EAud transfere a informação dos domínios correspondentes da Base DNS para a Base de Conexões, esta permanente, permitindo o correto enriquecimento e visualização pelo analista de segurança.

3.1.3.4 Sistema de Análise Inicial

O Sistema de Análise Inicial, ao receber um pacote de dados, verifica se é de um fluxo já conhecido (base temporária). Essa estratégia evita que o EAud realize consultas desnecessárias ao coletor, tornando a análise mais eficiente com a diminuição do tráfego gerado entre o auditor e o coletor, além da diminuição de processamento no auditor e no coletor. Se o fluxo não for conhecido pelo EAud, é realizada consulta ao ECol. Caso o pacote tenha sido percebido pelo ECol, é considerado legítimo e o fluxo é armazenado na base temporária para consultas rápidas futuras; caso contrário, é considerado suspeito e o Elemento Auditor armazena (base de conexões) os dados relevantes do fluxo para que, em conjunto com informações de enriquecimento e dos demais ECols, decida se o fluxo será considerado malicioso. Caso o pacote recebido seja uma resposta de DNS, o Sistema de Análise Inicial armazena as informações do domínio consultado e os IPs correspondentes. Tal ação permite a interligação de fluxos para um mesmo domínio, tornando os limiares de detecção mais precisos, além de aumentar o número de informações de enriquecimento. O processo de execução do sistema Inicial, ao receber um pacote de dados, é demonstrado na Figura 3.7

O Sistema de Análise Inicial foi desenvolvido em linguagem C, em virtude de ser o módulo mais crítico e que precisa ser performático. Assim como nas arquiteturas MADEX [8] e NERD [9], o elemento que realiza as consultas ao Coletor é o mais crítico e torna-se o gargalo da solução. Mesmo que a arquitetura PARDED não cause diretamente atrasos na comunicação analisada, grandes fluxos de dados poderiam enfileirar e retardar a análise, o que, em último caso, permitiriam o trânsito de pacotes suspeitos acima dos limiares definidos antes da correta informação de bloqueio ou mesmo o descarte de das réplicas dos

pacotes antes de sua análise.

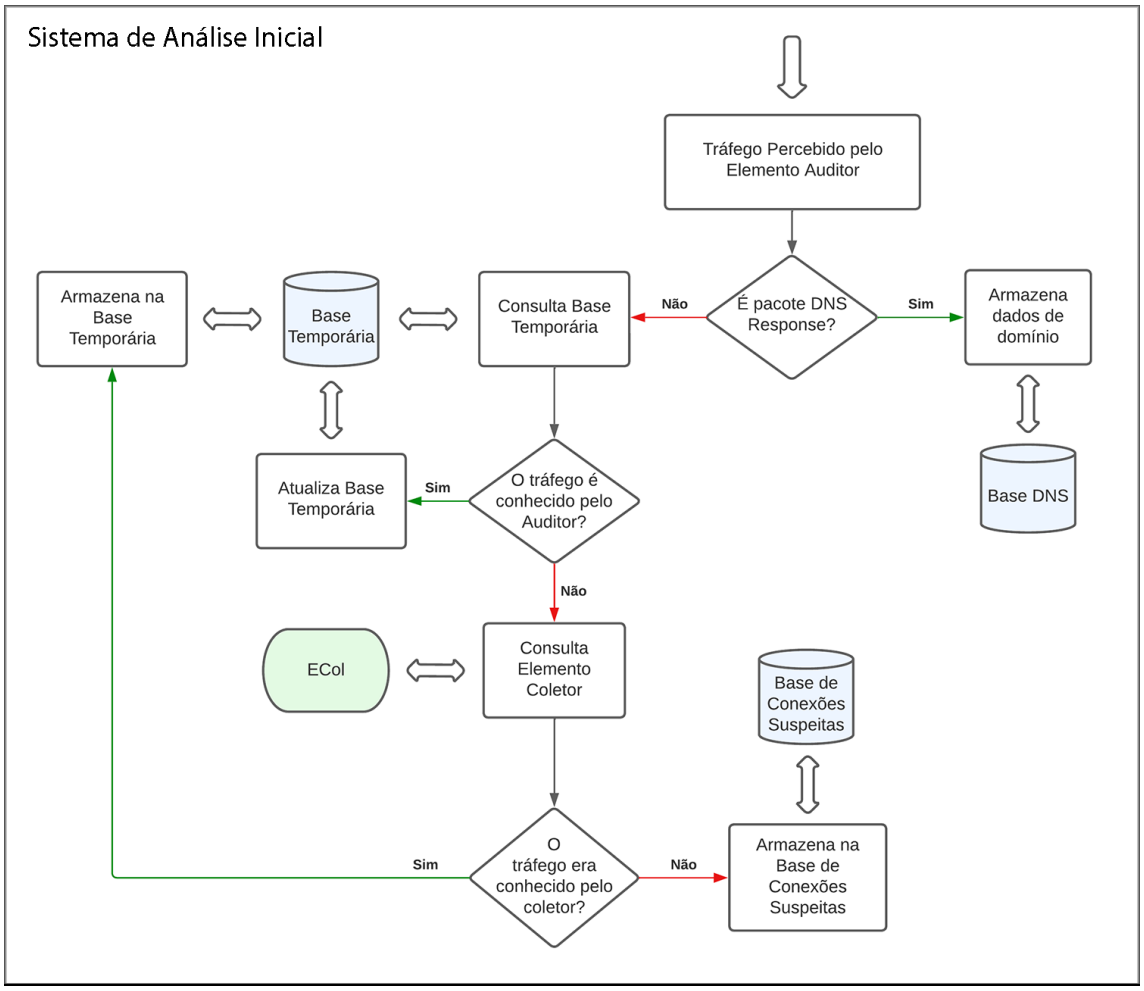


Figura 3.7: Sistema de Análise Inicial do EAud (Figura do Autor)

O sistema utiliza uma lista de bibliotecas disponíveis para linguagem C, em especial a utilização da biblioteca `etcdlib.h` para comunicação com o coletor e `libpq-fe.h` para comunicação com o banco de dados relacional, padrão SQL. Durante a implementação inicial, foi utilizada a biblioteca `libnetfilter_queue.h`, que fornece uma API para pacotes que foram enfileirados pelo filtro de pacotes do kernel. Essa abordagem foi substituída pela captura direta utilizando a biblioteca biblioteca `pcap.h` para criação de um *handle* de captura de pacotes, visto que o auditor do PARDED trabalha de forma passiva e não necessita apresentar um veredito sobre o bloqueio do pacote em tempo real, como ocorreria com um elemento *inline*. O bloqueio, se necessário, será feito pelo Elemento Bloqueante após receber as informações necessárias do Sistema de Aviso. A biblioteca `pthread.h` permite a utilização de paralelismo e torna-se importante em situações de grande volume de tráfego recebido pelo Sistema Inicial.

Para que o sistema consiga tratar o tráfego DNS, é necessário analisar o *payload* do pacote recebido de acordo com o padrão estrutural do protocolo, conforme Algoritmo 3.2. De forma a melhorar o desempenho da solução, o pacote é inicialmente considerado uma possível resposta de DNS apenas se o cabeçalho UDP for compatível. A partir dessa etapa, os campos `qr` e `rcode` do cabeçalho DNS são analisados pra confirmação de resposta sem erros. A partir dessa etapa, o campo `ans_count` é utilizado para obtenção

do número de respostas, visto que diversos IPs podem responder ao domínio consultado. Esses dados permitem ratar o *payload* DNS para obtenção das respostas.

Código 3.2: Estrutura do pacote DNS

```
struct DNS_HEADER
{
    unsigned short id; // identification number

    unsigned char rd :1; // recursion desired
    unsigned char tc :1; // truncated message
    unsigned char aa :1; // authoritive answer
    unsigned char opcode :4; // purpose of message
    unsigned char qr :1; // query/response flag

    unsigned char rcode :4; // response code
    unsigned char cd :1; // checking disabled
    unsigned char ad :1; // authenticated data
    unsigned char z :1; // its z! reserved
    unsigned char ra :1; // recursion available

    unsigned short q_count; // number of question entries
    unsigned short ans_count; // number of answer entries
    unsigned short auth_count; // number of authority entries
    unsigned short add_count; // number of resource entries
};
```

3.1.3.5 Sistema de Enriquecimento

O Sistema de Enriquecimento verifica, através da Base de Conexões, quais dados ainda não foram enriquecidos e consulta as bases (locais e externas) configuradas. O Sistema de Enriquecimento está implementado de tal forma que bases externas adicionais possam ser incorporadas. O processo de execução do Sistema de Enriquecimento, após iniciado, é demonstrado na Figura 3.8.

A execução ocorre de forma independente do Sistema de Análise Inicial. Após o início do processo, ele realiza consultas em intervalos pré-definidos, evitando que o enriquecimento cause atrasos na comunicação de um fluxo de dados, principalmente por depender de acessos externos que não são controlados pelo sistema desenvolvido. É possível também iniciá-lo de forma independente para que fluxos já consultados possam ser novamente avaliados, seja pela adição de bases ou pela atualização das bases já existentes, tanto internas quanto externas.

O Sistema de Enriquecimento, por não ter as mesmas características de desempenho do Sistema de Análise Inicial, foi implementado em linguagem Python 3, facilitando a compatibilização com diversas APIs como bases de dados internas e externas, além de facilitar acessos aos bancos de dados internos (Base de Conexões Suspeitas, Base DNS e Base Temporária).

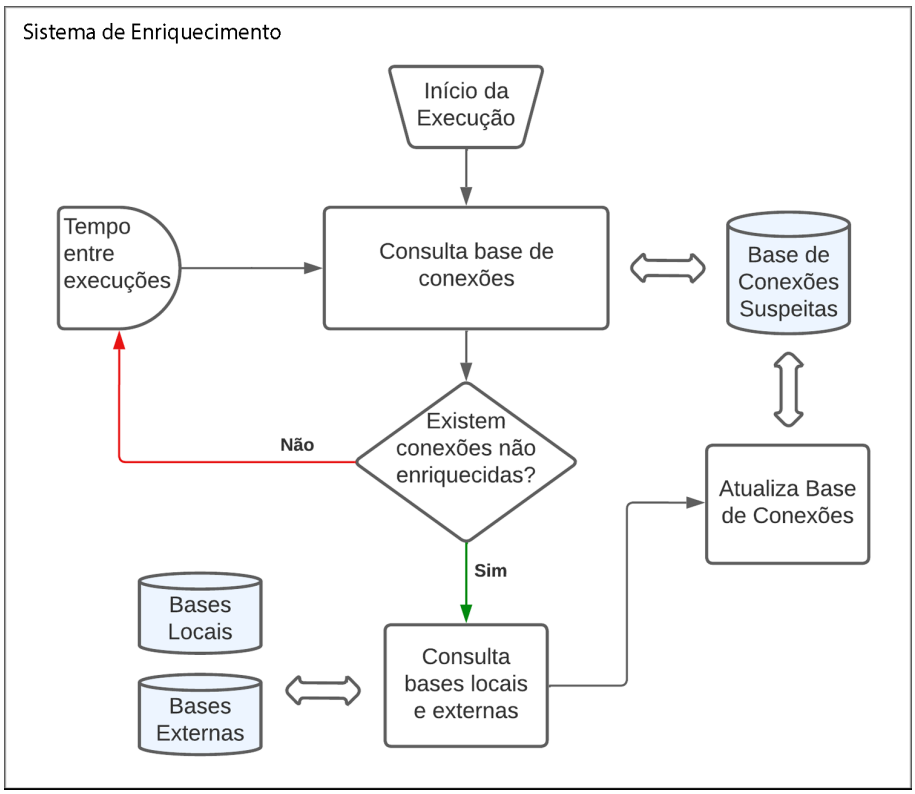


Figura 3.8: Sistema de Enriquecimento do Eaud (Figura do Autor)

3.1.3.6 Sistema de Aviso

O Sistema de Aviso verifica, através da base de conexões, quais destinos atingiram os limiares configurados para que um fluxo seja considerado malicioso e informa o Elemento Bloqueante. O Sistema de Aviso é escalável de tal forma que diferentes elementos bloqueantes possam ser utilizados.

Alterações nos limiares de detecção são automaticamente detectadas pelo Sistema de Aviso e os fluxos são novamente analisados. Por não terem as mesmas características de desempenho do Sistema Inicial, assim como os sistemas de Enriquecimento e Visualização, foram feitas em linguagem Python 3, facilitando a integração com elementos bloqueantes, como firewalls, e compatibilização com linguagens utilizadas na importação de regras, como arquivos de CTI STIX [4] ou SNORT [70]. Um exemplo de regra possível é demonstrado nos Códigos 3.3 e 3.4 (resumido).

O fluxograma de execução do Sistema de Aviso está demonstrado na Figura 3.9.

Código 3.3: Regras de bloqueio simples pdrão SNORT

```
1     alert tcp any any -> ip_malicioso porta
2     drop udp any any -> any any (content:"dominio_malicioso"; (...))
```

Código 3.4: Exemplo de Indicador de URL Maliciosa com STIX

```
{
  "type": "bundle",
  "id": "bundle--2a25c3c8-5d88-4ae9-862a-cc3396442317",
  "objects": [
    {
        "type": "indicator",
        "spec_version": "2.1",
        "id": "indicator--a932fcc6-e032-476c-826f-cb970a5a1ade",
        "created": "2014-02-20T09:16:08.989Z",
        "name": "File hash for Poison Ivy variant",
        "description": "Hash indicates that a sample of Poison Ivy is present.",
        "indicator_types": ["malicious-activity"],
        "pattern": "[file:hashes.'MD5' = '64d5d789c2b765448c8635fb6c782ea1']",
        "pattern_type": "stix",
    },
    {
        "type": "malware",
        "spec_version": "2.1",
        "id": "malware--fdd60b30-b67c-41e3-b0b9-f01faf20d111",
        "created": "2014-02-20T09:16:08.989Z",
        "name": "Poison Ivy",
        "malware_types": ["remote-access-trojan" ],
        "is_family": false
    },
    {
        "type": "relationship",
        "spec_version": "2.1",
        "id": "relationship--29dcdf68-1b0c-4e16-94ed-bcc7a9572f69",
        "created": "2020-02-29T18:09:12.808Z",
        "relationship_type": "indicates",
        "source_ref": "indicator--a932fcc6-e032-476c-826f-cb970a5a1ade",
        "target_ref": "malware--fdd60b30-b67c-41e3-b0b9-f01faf20d111"
    }
  ]
}
```

3.1.3.7 Sistema de Visualização

O Sistema de Visualização permite que o analista de segurança verifique os fluxos assinalados como suspeitos e suas características, além de um panorama de funcionamento da própria ferramenta, facilitando a utilização em ambientes de monitoramento e integração com demais sistemas de segurança. O sistema de visualização, assim como o Sistema de Enriquecimento, foi implementado em linguagem Python 3, com auxílio do *framework* de visualização de dados *Dash* [71], facilitando a criação de *dashboards* interativos e atualizações em tempo real.

Dessa forma, torna-se simples à equipe de resposta a incidentes entender quais os fluxos são possivel-

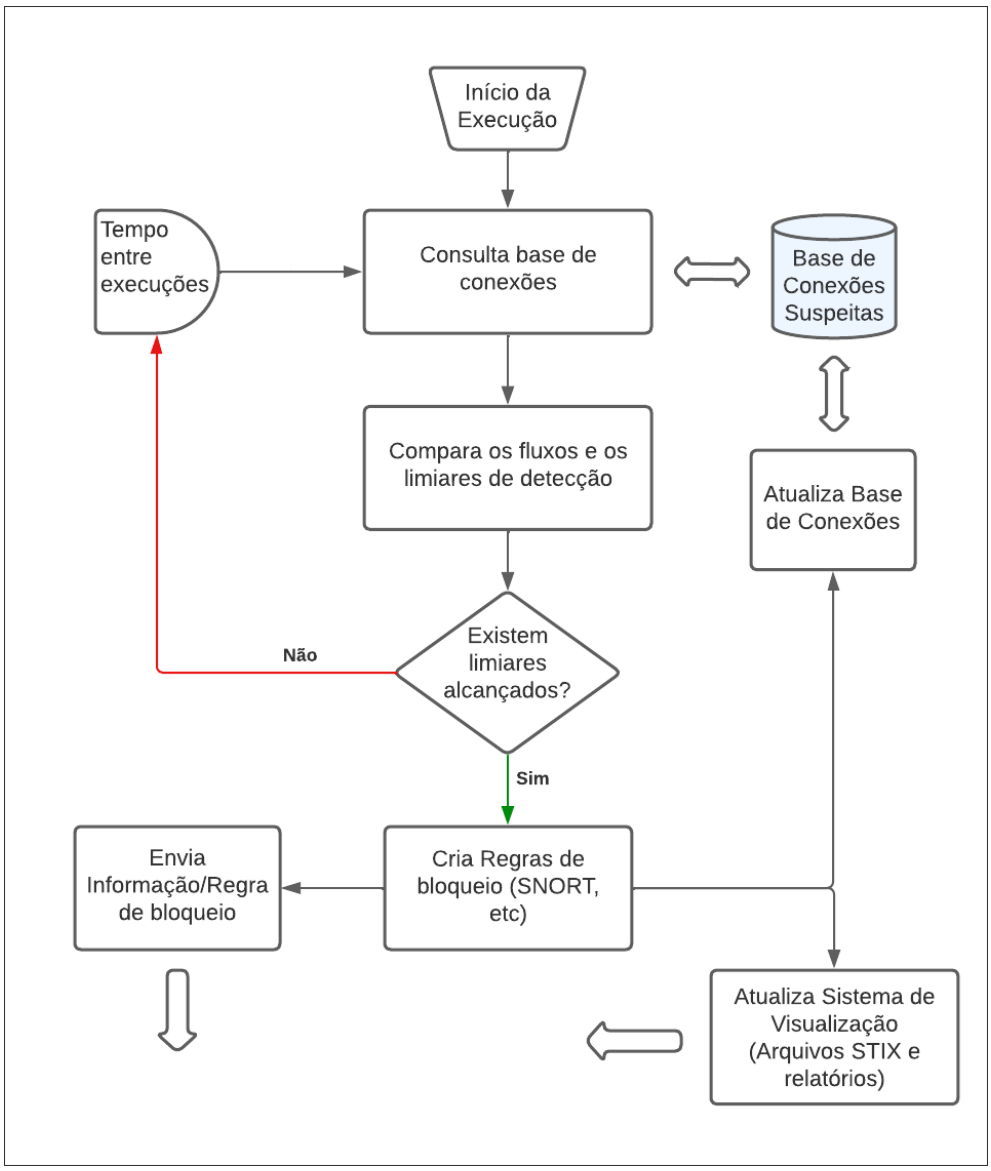


Figura 3.9: Sistema de Aviso do EAud (Figura do Autor)

mente maliciosos e interagir com a ferramenta, como filtrar informações, verificar data de de ocorrência de fluxos e analisar os dados enriquecidos.

O sistema de visualização permite ainda a análise de dados de inteligência a partir de arquivos em formato com patível com CTI, gerados pelo Sistema de Aviso para todos os fluxos que atingiram limiares de detecção, na linguagem utilizada pelo padrão STIX 2.1 (formato JSON), permitindo fácil integração com ferramentas de inteligência de ameaças. Também é possível visualizar de forma interativa, através de gráficos de vínculos, contendo objetos de domínio e relacionamentos, facilitando a observação do evento por um analista de segurança cibernética. A interatividade foi realizada com auxilio da ferramenta de código aberto STIX *Visualizer* [72], sem necessidade de envio da ação detectada para servidores remotos (a visualização é *offline*), garantindo sigilo da análise. Exemplo de visualização de uma possível ação maliciosa obtida através do PARDED, em conjunto com a ferramenta STIX *Visualizer*, está demonstrada na figura 3.10. As descrições, IPs e domínios demonstrados na figura 2.5 são fictícios, alterados a partir

dos fluxos originais.

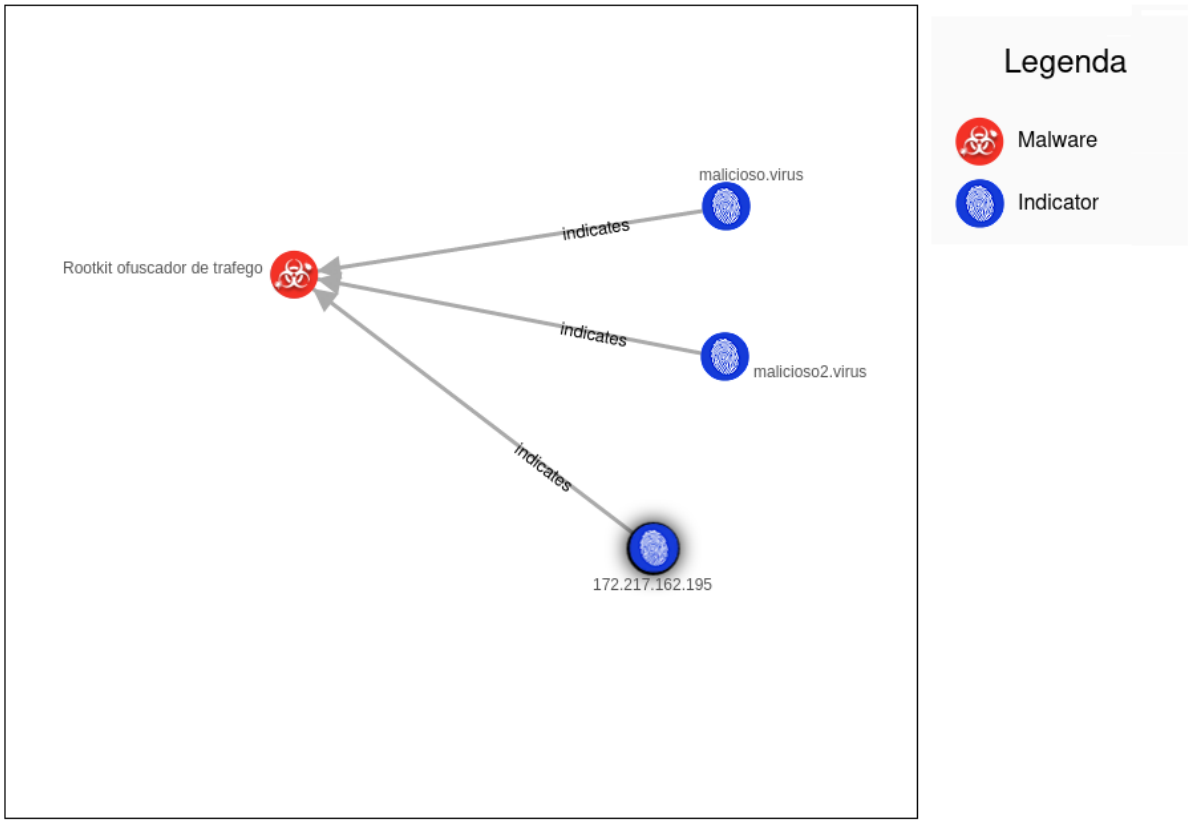


Figura 3.10: Visualização de CTI interativo (Figura do Autor)

A estrutura do Sistema de Visualização foi dividida em 4 partes. A primeira mostra os dados de enriquecimento realizados em bases internas e externas. Um exemplo de enriquecimento dividido em número de detecções e elementos ainda não analisados está demonstrado na Figura 3.11. A segunda parte apresenta um resumo dos fluxos coletados, com dados de fluxos enriquecidos (alimentado pelo Sistema de Enriquecimento), fluxos bloqueados (alimentado pelo Sistema de Aviso) e a quantidade de IPS e domínios detectados (alimentado pelo Sistema de Análise Inicial) conforme demonstrado na Figura 3.12. Também é demonstrado um resumo do status de funcionamento dos sistemas presentes no EAud, incluindo o tempo total de execução de cada sistema.

Na terceira parte do Sistema de Visualização, são demonstrados os dados de cada fluxo detectado, destacando os que atingiriam os limiares de detecção e foram bloqueados. Ao selecionar um fluxo da tabela, os dados de enriquecimento em bases externas e internas são apresentados e, caso seja selecionado um fluxo bloqueado pelo Sistema de Aviso, permite abrir os dados de bloqueio em formato de CTI. Os fluxos, dados de enriquecimento e opções de CTI são demonstrados na figura 3.13 (IPs e domínios alterados para dados fictícios). A última parte do sistema, conforme Figura 3.14 contém mapa com a localização geográfica de cada fluxo, baseado nos endereços IP de destino e incluindo informação da cidade e do país onde o IP está registrado. A geolocalização foi realizada com auxílio da base GeoLite2-City, disponibilizada de forma gratuita pela empresa Maxmind Inc.[73].

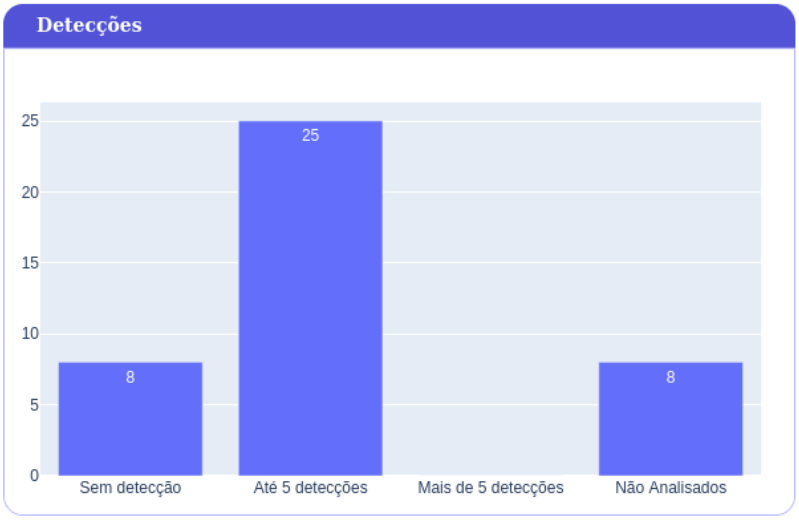
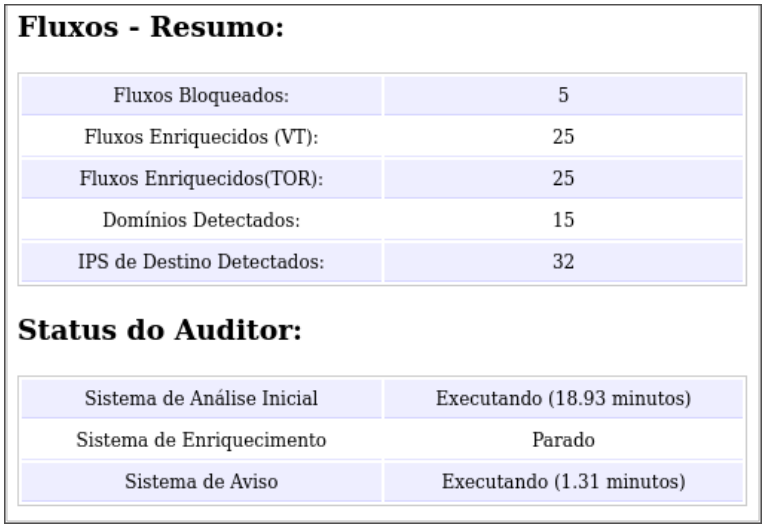


Figura 3.11: Exemplo de base de detecções no Sistema de Visualização (figura do Autor)



Fluxos - Resumo:

Fluxos Bloqueados:	5
Fluxos Enriquecidos (VT):	25
Fluxos Enriquecidos(TOR):	25
Domínios Detectados:	15
IPS de Destino Detectados:	32

Status do Auditor:

Sistema de Análise Inicial	Executando (18.93 minutos)
Sistema de Enriquecimento	Parado
Sistema de Aviso	Executando (1.31 minutos)

Figura 3.12: Resumo dos fluxos e Status da solução no Sistema de Visualização (figura do Autor)

3.1.4 Elemento Bloqueante (EBlk)

O Elemento Bloqueante (EBlk) é um equipamento *inline* que verifica se o pacote de rede recebido deve ou não ser bloqueado, baseado no aviso recebido pelo EAud. Esse elemento pode ser qualquer dispositivo que permita receber atualizações de regras de lerta ou bloqueio de forma remota, como roteadores e firewalls.

O tempo decorrido entre o primeiro pacote suspeito detectado, o enriquecimento da informação, a ação do Sistema de Aviso para bloqueio e a efetiva atuação pelo EBlk depende de diversos fatores, como o tempo de consulta ao ECol, tempo de consulta às bases remotas para enriquecimento, atualização da base de conexões e atingimento dos limiares definidos no auditor.

ID	IP DE ORIGEM	IP DE DESTINO	ENRIQUECIMENTO VT	ENRIQUECIMENTO TOR	BLOQUEADO	DOMÍNIOS
28	192.168.190.11	172.217.162.195	true	true	true	google.com.pt,google2.com.pt
29	192.168.190.11	142.250.218.206	true	true	false	
30	192.168.190.11	142.251.132.3	true	true	false	
31	192.168.190.11	142.251.129.131	true	true	false	
32	192.168.190.11	208.84.244.50	true	true	false	
33	192.168.190.11	142.251.128.99	true	true	false	
34	192.168.10.30	104.244.72.132	true	true	true	
35	192.168.10.16	104.244.74.55	true	true	true	
37	192.168.190.10	20.48.104.6	true	true	true	
38	192.168.190.10	142.250.219.142	true	true	false	google.com
39	192.168.190.10	23.108.123.73	true	true	false	ruyicallback.771658.com
40	192.168.190.10	199.232.114.132	true	true	false	debian.map.fastlydns.net

STIX - Visualização | STIX - Arquivo JSON | Relatório

Fluxos Suspeitos - Virustotal

DETECÇÕES MALICIOSAS	DOMINIO	IP	DATA COLETA
6		104.244.72.132	2022-07-15T12:09:15.529762

Fluxos Suspeitos - Rede TOR

NOME	FLAGS	VERSÃO	DATA COLETA
Quetzalcoatl	EFGRSDV	Tor 0.4.7.8	2022-07-17T00:36:23.099918
Quetzalcoatl	EFGRSDV	Tor 0.4.7.8	2022-07-17T00:38:14.739771

Figura 3.13: Fluxos, enriquecimento e CTI no Sistema de Visualização (figura do Autor)

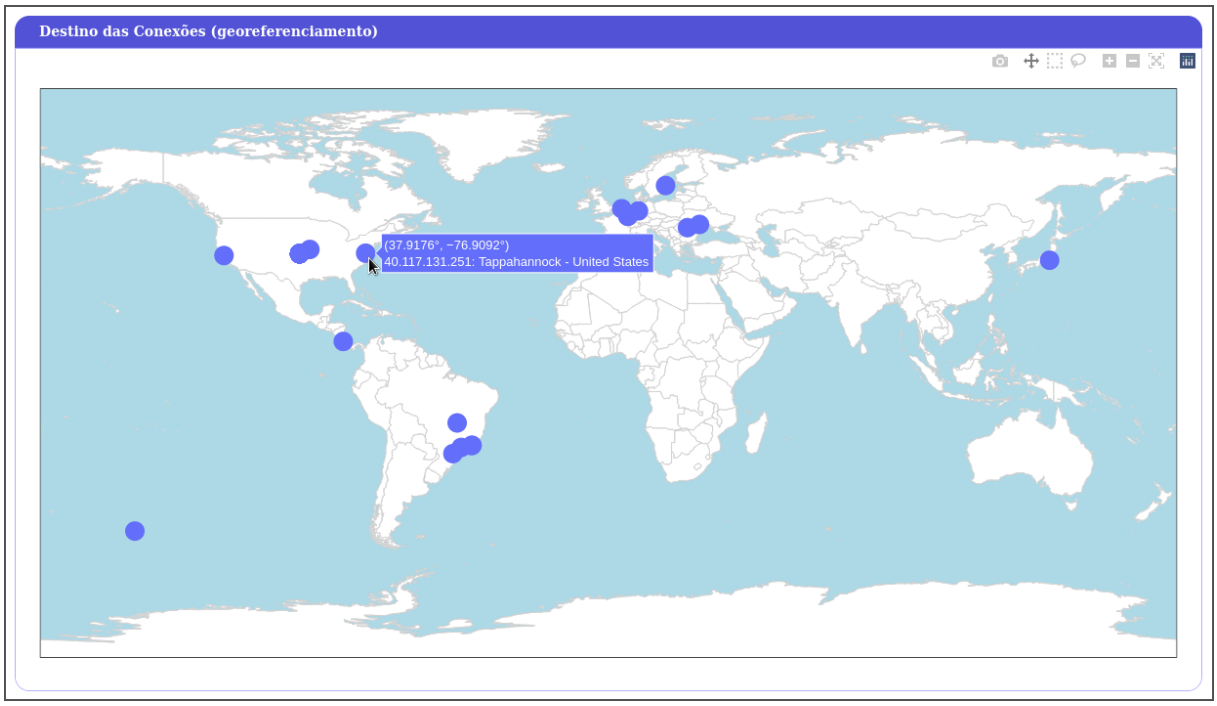


Figura 3.14: Georreferenciamento dos fluxos no Sistema de Visualização (figura do Autor)

3.1.5 Elemento Copiador

A função do Elemento Copiador é permitir o funcionamento passivo do Elemento Auditor, espelhando o tráfego que passa por ele. Desta forma, o tráfego original é encaminhado para o próximo elemento de rede e sua cópia é enviada para análise do EAud, evitando atrasos na transmissão de dados ao destino.

É possível também utilizar o Elemento Copiador para limitar o tipo de tráfego a ser analisado pelo EAud, filtrando pacotes indesejáveis ou espelhando apenas tráfegos com o padrão desejado. A arquitetura PARDED não define como deve ser implementado o Elemento Copiador, pois é possível a utilização de qualquer equipamento existente na infraestrutura que realize cópias dos pacotes trafegados, como comutadores, roteadores, hubs ou firewalls. Uma possibilidade para efetuar a cópia do tráfego seria através do uso de técnicas de *port mirroring*, existentes em todos os comutadores modernos, conforme Figura 3.15.

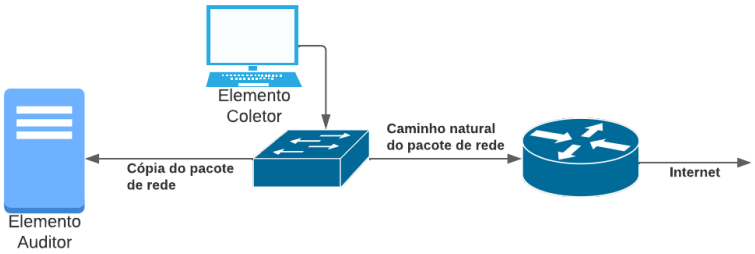


Figura 3.15: Espelhamento de tráfego com *port mirroring* (figura do autor)

Os elementos descritos na arquitetura proposta, contendo os fluxos de funcionamento do EAud e do ECol, foram implementados em ambiente de laboratório. Foi possível, então, a execução de testes de conceito, de forma a validar a estrutura proposta, bem como testes de desempenho, para avaliar o funcionamento em situações próximas a encontrada em uma infraestrutura de rede real e detectar possíveis limitações de uso. O próximo capítulo descreve os testes e resultados obtidos pela solução.

4 TESTES E RESULTADOS

Os testes foram realizados com o intuito de averiguar a viabilidade da arquitetura proposta e implementada e o tempo de execução de cada etapa do processo de análise dos pacotes pelo Auditor, incluindo o tempo gasto para processamento de pacotes com e sem consulta ao Elemento Coletor, o tempo necessário para enriquecimento e o tempo utilizado para análise de limiares de detecção suspeita, marcação dos fluxos e geração de arquivos padrão STIX e SNORT. Para tal, o EAud e o ECol foram diretamente conectados, sem a existência de outros elementos de rede intermediários. O Elemento Coletor foi configurado conforme descrito por Terra e Gondim [9].

Após os testes de desempenho, foram analisadas as funcionalidades do sistema de visualização, como a correta visualização de fluxos suspeitos, marcação dos fluxos que ultrapassaram limiares de detecção, visualização das bases de enriquecimento, georreferenciamento e arquivos gerados pelo Sistema de Aviso do EAud.

4.1 ESTRUTURA UTILIZADA PARA TESTES

Para testar o ECol, foi utilizado o Rootkit Linux Nuk3Gh0st [74]. O motivo do uso desse software é a disponibilização de seu código fonte de forma gratuita e por possuir a funcionalidade de ofuscação de tráfego, necessária para validação da proposta desse trabalho. De acordo com o desenvolvedor, o código é compatível com Linux com versões de kernel entre 2.6.32 e 4.17.9. Os sistemas operacionais testados para verificação do funcionamento do *rootkit* estão discriminados na Tabela 4.1.

Em alguns dos sistemas operacionais (S.O.) não foi possível reproduzir a mesma versão de Kernel indicada, em virtude de não estar mais disponível nos repositórios oficiais de cada distribuição Linux. Optou-se por não recompilar o Kernel da fonte genérica original nos S.Os testados, pois essa ação depende de parâmetros de configuração e o resultado possivelmente apresentaria diferenças do sistema originalmente compilado pela equipe mantenedora de cada distribuição. Desta forma, a recompilação de Kernel poderia gerar comportamentos inesperados e não seria possível garantir igualdade dos testes com os realizados pelo desenvolvedor do *roootkit*.

Tabela 4.1: Distribuições Linux nas quais o *Rootkit* foi testado.

Distribuição Linux	**Arquitetura**	**Compatibilidade (conforme desenvolvedor)**	**Resultado do Teste**
Kali Linux (Rolling)	x86_64	Sim	Instável
Ubuntu 16.04.4 LTS	x86_64	Sim	Instável
Ubuntu 17.04	x86_64	Sim	Instável
Debian 9	i686	Sim	Instável
Ubuntu 12.10	i686	Sim	Instável
Debian 11	x86_64	Não Testado	Instável

Em todos os S.O. testados, foi possível ativar o software e utilizar algumas de suas funcionalidades, como a ofuscação de processos, ofuscação de arquivos selecionados, ofuscação do diretório de instalação do *rootkit* e ofuscação de pacotes de rede, tanto por IP quanto por porta. O resultado apresentou perfeito funcionamento e estabilidade na ofuscação de arquivos, porém ficou instável em alguns testes de ofuscação de processos em versões de Kernel mais recentes e, para ofuscação de pacotes, mostrou-se instável em todos os S.Os. e versões de kernel analisados, tanto em instalações feitas diretamente em hardware físico quanto em instalações feitas em máquinas virtuais, realizadas com auxílio do virtualizador Virtualbox 6.1.

Para que os testes funcionassem com mais de um fluxo de comunicação ofuscado sem instabilidades, foi necessário gerar um *script* que ativa e desativa a função de ofuscação para cada fluxo enviado. Essa situação permite testar o funcionamento do *rootkit* e executar testes simples, mas dificulta ou mesmo inviabiliza testes de alta carga. As simulações executadas e as instabilidades apresentadas ("*Kernel panic*"com travamento completo do Sistema Operacional) corroboram com a informação disponibilizada pelo desenvolvedor [74] para sistemas multiprocessados ou com a função de ofuscação de pacotes habilitada.

Os testes de alta carga de pacotes foram, então, realizados com uma simulação do funcionamento real do *rootkit*, ou seja, sem uma ofuscação em curso feita pelo *rootkit* e sim injetada diretamente no código do Elemento Coletor. Para tal, uma modificação do ECol impediu que este registrasse determinados fluxos. O EAud, ao consultar o Coletor, recebe a mesma informação que seria recebida de um terminal infectado, pois o ECol, ao consultar sua base, considera que o fluxo solicitado pelo EAud não foi percebido pelo terminal. Essa abordagem permitiu a eliminação de instabilidade, além de evitar possíveis gargalos na ação do *rootkit*, ou seja, que problemas na ofuscação interferissem no resultado dos testes dos elementos constantes na solução PARDED.

Considerando a abordagem para eliminação de instabilidade descrita acima, foi possível utilizar versões mais novas de S.O e Kernel para execução dos testes. Dessa forma, optou-se pela utilização do Linux Debian 11 para execução tanto do ECol quanto do EAud, facilitando o gerenciamento dos sistemas, conforme Tabela 4.2.

Tabela 4.2: Distribuição Linux utilizada em laboratório.

Distribuição	**Arquitetura**	**Kernel release**	**Kernel version**
Linux Debian 11 (bullseye)	x86_64	5.10.0-16	5.10.127-1 (30/06/2022)

4.2 CENÁRIOS DE TESTE

Para realização do teste, foram propostos diversos cenários para possibilitar a análise de todos os componentes da arquitetura PARDED. Inicialmente, foram estruturados testes de detecção dos fluxos pelos elementos Coletor e Auditor, considerando um cenário onde não há *malware* ofuscando pacotes de dados, permitindo analisar se os fluxos são armazenados pelo ECol, consultados pelo EAud, armazenados na Base Temporária e na Base DNS (para pacotes DNS Response) e se haveriam falsos-positivos (fluxos legítimos detectados como suspeitos). Após a realização do primeiro cenário, foram realizados testes considerando a transmissão apenas de fluxos maliciosos, ou seja, o ECol não detectou nenhum pacote. Esse

cenário permite verificar a existência de falsos-negativos (fluxos suspeitos detectados como legítimos) e o desempenho de detecção de pacotes maliciosos. O terceiro cenário foi composto de testes de detecção de fluxos legítimos e suspeitos (não detectados pelo ECol) trafegando concomitantemente, permitindo a analisar o armazenamento dos fluxos na Base de Conexões, incluindo os dados de domínio, e se haveriam falsos-positivos ou falsos-negativos. Os cenários descritos englobam as funções do ECol (incluindo a Base de Fluxos) e do Sistema de Análise Inicial do EAud (incluindo a Base Temporária, a Base de Conexões e a Base DNS).

Em seguida, foi testado o enriquecimento dos dados constantes da Base de Conexões, dividido em dois cenários, sendo o primeiro considerando uma base local, armazenada no mesmo S.O. do Elemento Auditor, e o segundo através do uso de uma base remota, permitindo verificar se os dados eram enriquecidos e atualizados, se o relacionamento entre as tabelas presentes na Base de Conexões estavam consistentes e se a análise assíncrona estava impactando negativamente no correto funcionamento do sistema.

Com dados enriquecidos, foi possível estruturar testes de detecção de limiares para bloqueio dos fluxos suspeitos através do Sistema de Aviso. Os testes foram definidos para verificar a correta aplicação dos limiares de detecção, atualização da Base de Conexões com a informação de bloqueio, geração de modelo de bloqueio e geração de modelo de CTI.

Por fim, testes de interface de visualização, permitindo analisar se os dados apresentados estavam condizentes com os presentes na Base de Conexões, se as informações de enriquecimento e bloqueio estavam atualizadas e destacadas, se a aplicação de filtros estava funcional e se os modelos de bloqueio e CTI estavam acessíveis.

4.2.1 Detecção de Fluxos Exclusivamente Legítimos

No cenário de teste de detecção de fluxos legítimos, foram utilizados, para medição de tempo de resposta do sistema, apenas fluxos TCP e pacotes “DNS Response", pois são o foco de análise da solução proposta. O teste foi realizado a partir da simulação de tráfego, com mensagens enviadas pelo terminal que contém o Elemento Coletor e copiadas para o Elemento Auditor por técnica de *port mirroring*. Dessa forma, outros pacotes trafegados durante os testes, por não serem objeto de análise da solução (por exemplo, pacotes ICMP ou UDP em portas diferentes da utilizada pelo protocolo DNS),também foram enviados ao EAud; entretanto, foram descartados sem consulta ao ECol ou qualquer análise do conteúdo do pacote.

Nesse cenário, o *rootkit* não estava ativo (sem tráfego ofuscado) e o resultado esperado era a detecção de todos os fluxos como legítimos, com inclusão dos fluxos conhecidos na tabela Temporária e das respostas de consultas DNS inseridas na Base DNS. Foi realizada, de forma simulada e controlada, a transmissão de pacotes do terminal contendo o ECol para pontos da rede na qual o tráfego era espelhado para o EAud, da mesma forma prevista pela arquitetura PARDED, descrita no Capítulo 3 (Figura 3.2).

Durante a geração do tráfego, foram feitas análises de desempenho do sistema para fluxos TCP legítimos, considerando os pacotes com consulta ao ECol e sem consulta (informação presente na Base Temporária), além de pacotes *DNS response*.

4.2.2 Detecção de Fluxos Exclusivamente Suspeitos

No cenário de teste de detecção de fluxos suspeitos, onde todos os pacotes transmitidos eram maliciosos (não detectados pelo ECol), foram utilizados apenas pacotes TCP SYN com porta de origem diferente para cada pacote, de forma que cada um dos pacotes de dados transmitidos fosse detectado pelo EAud como um novo fluxo. Essa abordagem permitira verificar o desempenho do sistema no pior cenário, onde cada pacote gera uma consulta ao ECol e gera também a necessidade de atualização da Base de Conexões. O teste foi realizado com o *rootkit* ativo e o tráfego foi gerado a partir do envio de pacotes forjados para IPs ofuscados pelo *rootkit*, de forma a evitar detecção pelo Elemento Coletor.

Diferente dos testes de Fluxos Exclusivamente Suspeitos, as mensagens foram enviadas pelo terminal que contém o Elemento Coletor para o Elemento Auditor através de um filtro de pacotes. Esse filtro foi gerado através do firewall IpTables, com regra de descarte de todos os pacotes que não possuíssem a origem e o destino previsto nos pacotes forjados. Dessa forma, o filtro impediu que qualquer outro pacote transmitido pelo terminal analisado (por exemplo, pacotes enviados pelo S.O.) não interferirem nos testes, ou seja, não chegassem ao EAud, garantindo que todos os pacotes analisados pelo EAud fossem conhecidamente maliciosos.

4.2.3 Detecção de Fluxos Legítimos e Suspeitos

Para detecção de fluxos suspeitos em uma situação próxima a uma rede real, foram transmitidos a mesmo tempo pacotes legítimos e maliciosos. Assim como nos testes de Fluxos Legítimos, apenas dados de conexões TCP e pacotes “DNS Response"foram utilizados para medição de tempo de resposta do sistema, pois são o foco de análise da solução proposta. Também foi realizada simulação de tráfego, com mensagens enviadas pelo terminal que contém o Elemento Coletor e copiadas para o Elemento Auditor por técnica de *port mirroring*.

Nesse cenário, o *rootkit* estava ativo e foi realizada transmissão de tráfego ofuscado, correspondente a aproximadamente 2% do tráfego sem carga (100 fluxos de 10 pacotes cada), com intuito de verificar se, entre dados legítimos, os pacotes maliciosos seriam corretamente detectados e os dados inseridos na Base de Conexões. A Base Temporária e a Base DNS, populadas no cenário de Fluxos Legítimos, foi reiniciada antes dos testes, evitando interferência de um cenário no resultado do outro. Durante a geração do tráfego, foram feitas análises de desempenho do sistema, para fluxos legítimos, suspeitos e pacotes DNS, assim como no teste anterior.

4.2.4 Enriquecimento de Fluxos Suspeitos

Os testes de enriquecimento foram realizados com divisão em dois tipos de consulta: local e remota. Essa diferenciação deve-se ao fato de que, em consultas remotas, o enriquecimento depende de fatores externos à solução, como conexão à Internet e dependência de perfeito funcionamento do sistema remoto no momento da consulta.

O enriquecimento local foi realizado com auxílio de duas bases de dados, sendo uma base de nós da rede TOR, obtida através do website *dan.me.uk* [75], e uma de geolocalização, obtida através da base

GeoLite2-City, disponibilizada pelo empresa Maxmind Inc. [73]. Para execução dos testes de desempenho de enriquecimento local, foi escolhida a base dedados de nós da rede TOR.

O enriquecimento remoto foi realizado através de acesso à base de detecção de IPs e Domínios fornecidos pela plataforma Virustotal [76]. Essa base permite a análise de arquivos, domínios, IPs e URLs suspeitos para detectar presença de *malwares* e outras violações.

O teste previu, ainda, interrupção e reinício do enriquecimento, além de interrupção momentânea da internet. Esse teste permitiu verificar a inserção assíncrona dos dados na Base de Conexões e consistência mesmo em casos de impossibilidade de execução da função de enriquecimento.

4.2.5 Detecção de Limiares pelo Sistema de Aviso

Os testes do Sistema de Aviso utilizou os limiares de detecção apresentados no Capítulo 3, onde é definida a arquitetura PARDED, e foram divididos em 4 possibilidades de detecção, da seguinte forma:

- Teste do primeiro limiar através da geração de 6 fluxos de mesma origem para um mesmo destino não detectado pelo ECol. Esse teste depende do sistema de aviso detectar, na base gerada pelo Sistema de Análise Inicial, esses fluxos;
- Teste do segundo limiar através da geração de 4 fluxos, sendo 2 de cada origem, para um mesmo destino, sem detecções pelos respectivos ECol. Esse teste depende do sistema de aviso concentrar, na base gerada pelo Sistema de Análise Inicial, os fluxos para um mesmo destino gerados de ECols diferentes;
- Teste do terceiro limiar (base local) através da geração de 2 fluxos de mesma origem para um destino não detectado pelo ECol, sendo esse destino um nó da rede TOR. O teste depende do correto enriquecimento por base local e a detecção do fluxo enriquecido pelo Sistema de Aviso;
- Teste do terceiro limiar (base remota) através da geração de 2 fluxos de mesma origem para um destino não detectado pelo ECol, sendo esse IP de destino considerado malicioso por ao menos 4 empresas de segurança. Esse teste depende do correto enriquecimento por base remota e a detecção do fluxo enriquecido pelo Sistema de Aviso;

Para execução dos testes de desempenho do Sistema de Aviso, foram definidos dois cenários possíveis: o primeiro, quando a consulta não encontra fluxos que atingiram os limiares de detecção, ou seja, nenhuma ação é realizada além das consultas à Base de Conexões Suspeitas; o segundo, quando um limiar de detecção é atingido e ações posteriores são executadas. A separação dos cenários foi realizada em virtude de como o Sistema de Aviso foi estruturado, pois, no caso de detecção, é necessário atualizar a Base de Conexões Suspeitas com a informação de bloqueio e criar arquivos estruturados, em formato compatível com linguagem HTML, contendo as informações de bloqueio e de CTI, para que o Sistema de Visualização receba as informações de forma legível para apresentação ao analista de segurança, gerando maior processamento.

A medição de desempenho do segundo cenário foi possível através de alteração manual no banco de

dados no número de fluxos detectados para um mesmo destino, permitindo a simulação de várias detecções pelo do Sistema de Aviso sem o envolvimento do Sistema de Enriquecimento.

4.2.6 Interface de Visualização

No cenário realizado em laboratório, o Sistema de Visualização foi executado após a adição de dados suspeitos pelo Sistema de Análise Inicial, com a inclusão de fluxos que ultrapassaram os limiares de detecção previstos. Os primeiros fluxos foram adicionados com o Sistema de Enriquecimento ativo e os oito últimos com o Sistema de Enriquecimento inativo, para que as colunas que contém a informação de ausência do enriquecimento fossem testadas. Com a Base de Conexões Suspeitas e a Base DNS povoadas, foram realizadas as seguintes ações de teste:

- Verificação da inclusão dos fluxos no Sistema de Visualização, incluindo marcação dos fluxos que ultrapassaram os limiares de detecção testados (fluxos bloqueados);
- Verificação da inclusão dos fluxos nas tabelas de enriquecimento, tanto para a base local (TOR) quanto remota (Virustotal);
- Verificação da visualização dos dados de enriquecimento ao selecionar um fluxo;
- Verificação da correta visualização de arquivos no formato STIX para os fluxos bloqueados pelo Sistema de Aviso, tanto em formato JSON quanto no modo interativo;
- Verificação da correta visualização de arquivos no formato de regras SNORT;
- Verificação da funcionalidade do sistema de Georreferenciamento;
- Verificação da funcionalidade do Status do EAud, com detecção de funcionamento de cada Sistema (análise inicial, enriquecimento e aviso);

Todas as verificações descritas acimas foram consideradas necessárias para que o sistema de visualização esteja de acordo com o previsto na arquitetura.

4.3 RESULTADOS

4.3.1 Desempenho na Detecção de Fluxos e Atualização de Bases

No primeiro cenário, contendo apenas pacotes legítimos, todos os pacotes recebidos no EAud foram percebidos corretamente como legítimos, ou seja, não houve falsos positivos. Os pacotes que não estavam presentes na Base Temporária do EAud (o que requer consulta ao ECol) foram, em média, processados em 0,5 segundos. Os demais pacotes do fluxo (já inseridos na Base Temporária), 95% do total gerado, foram processados em menos de 0,0012 milissegundos cada pelo Sistema de Análise Inicial. Os pacotes "DNS response", tratados localmente para detecção dos possíveis domínios utilizados pelos *rootkits*, foram

Tabela 4.3: Tempo de resposta do sistema (sem atuação do rootkit).

Tipo de Processamento	**Característica do Fluxo**	**Média de Pacotes**	**Tempo de processamento (ms)**	**Tempo de processamento com carga de 50Mbps(ms)**
Sem consulta ao Coletor	Na base Temporária	55.393	0,00110	0,00100
	Resposta DNS	2030	0,00407	0,00734
Com consulta ao Coletor	Fluxos Legítimos	983	516,78	551,05

processados em aproximadamente 0,0041 milissegundos. A Tabela 4.3 demonstra os resultados obtidos com transmissão apenas de tráfego legítimo.

No segundo cenário, contendo apenas fluxos conhecidamente maliciosos, o *rootkit* estava ativo no ECol e foi realizada a transmissão de tráfego ofuscado contendo 100 fluxos de 1 pacote por segundo cada. Todos os pacotes recebidos e tratados no EAud foram percebidos corretamente como suspeitos, ou seja, não houve falsos-negativos. O tempo de resposta foi, em média de 1,71 segundos. O motivo da elevação do tempo de processamento deve-se ao fato do tratamento e atualização da base de conexões suspeitas, comprovando a grande diferença entre a utilização de API do banco de dados relacional PostgreSQL para consultas diretas em memória volátil.

Em seguida, foi realizada a transmissão de tráfego ofuscado contendo 100 fluxos de 10 pacotes por segundo cada. O tempo médio de resposta foi, em média, de 1,67 segundo, muito próximo do teste anterior (com transmissão de 1 pacote por segundo). Entretanto, como o intervalo de recebimento de cada pacote (0,1 segundo) é muito menor do que o tempo de tratamento deste pacote (em média, 1,67 segundos), pacotes foram mantidos em *buffer* pela biblioteca de captura enquanto pacotes anteriores ainda estavam em tratamento. Dessa forma, mesmo que o tempo de transmissão foi relativamente baixo (10 segundos para transmissão dos 100 fluxos), o tempo para que a Base de Conexões fosse completamente atualizada foi muito maior (ao menos 167 segundos).

Considerando o envio de pacotes em taxas maiores que o tempo de processamento de cada pacote, usando a biblioteca *libpcap*, é possível que ocorra uma situação de "*packet buffer timeout*", ou seja, a partir de certo atraso entre o pacote recebido e o efetivo tratamento, o pacote é descartado. Nos testes realizados com *buffer timeout* igual a 1000 milisegundos, transmissão de 10 pacotes por segundo e tempo de processamento de 1,67 segundos por pacote, ocorreram descartes após aproximadamente 85 pacotes. Na situação em que não foi definido *timeout*, não houve descarte de pacotes. Essa situação pode ser indesejável em soluções *inline*, pois, conforme a descrição da biblioteca libpcap, o *buffer timeout* é necessário para que um aplicativo não espere que o *buffer* de captura do sistema operacional seja preenchido antes que os pacotes sejam entregues.

Em transmissão de tráfego ofuscado contendo 500 fluxos de 10 pacotes por segundo cada, o EAud comportou-se de modo similar, com tempo médio de tratamento de cada pacote de 1,68 segundos. A Tabela 4.4 demonstra os resultados obtidos com transmissão de fluxos exclusivamente maliciosos.

Foi possível verificar que, entre os pacotes transmitidos, não houve falsos-negativos (pacotes tratados como legítimos, mesmo sendo maliciosos); entretanto, como a taxa de transmissão foi superior ao do processamento dos pacotes, houve atraso na atualização da Base de Conexões. Uma transmissão de 10

pacotes por segundo, durante 50 segundos, em um cenário de tratamento sequencial (sem processamento paralelo), houve um atraso de 13 minutos para completa atualização da Base de Conexões. Essa situação demonstra a necessidade de processamento paralelo dos pacotes no EAud, principalmente nas situações de transmissão de muitos fluxos maliciosos, nos quais é necessária consulta ao ECol e também atualização de banco de dados relacional (Base de Conexões).

No terceiro cenário contendo adição de fluxos suspeitos entre fluxos legítimos, simulando uma situação mais próxima à real, o *rootkit* estava ativo no ECol e foi realizada transmissão de tráfego ofuscado, correspondente a aproximadamente 2% do tráfego legítimo sem carga (100 fluxos de 10 pacotes cada). No caso dos pacotes que necessitaram consulta ao ECol (não estavam na Base Temporária) e foram considerados suspeitos, ou seja, sem detecção do pacote pelo ECol, o tempo de resposta foi, em média de 1,6 segundos, próximo ao obtido no cenário anterior, onde todos os pacotes eram maliciosos. Não houve alteração significativa no tempo de processamento de pacotes *DNS Response* ou de pacotes presentes na Base Temporária. A Tabela 4.5 demonstra os resultados obtidos com transmissão de fluxos suspeitos em conjunto com o tráfego legítimo.

4.3.2 Desempenho e Consistência do Enriquecimento

As consultas do Sistema de Enriquecimento feitos utilizando a consulta local (base de dados de nós TOR) foram processadas em aproximadamente 42 milissegundos cada, conforme Tabela 4.6. Considerando que é uma base de texto pequena, com elementos separados por vírgulas, era esperada bom desempenho, mesmo com consultas sendo feitas sequencialmente. Considerando a possibilidade de ordenamento da base e utilização de algoritmos de busca mais eficientes, o resultado pode ser melhorado. Entretanto, é possível deduzir que o enriquecimento local será pouco utilizado, pois, em via de regra, a maior parte das bases de dados de consultas são acessadas remotamente.

Os testes de enriquecimento remoto foram realizados em tempo real, através de consultas na API da plataforma Virustotal para obtenção dos dados relativos a análise dos fornecedores de segurança. Como esse teste depende de fatores não controlados pelo EAud, como conexão da internet utilizada, além do tempo de processamento e resposta da API pesquisada, os resultados apresentaram maior variância e, como previsto, tempo de resposta muito superior ao da base local. Em média, foram necessários cerca de 1,2 segundos para consultas do Sistema de Enriquecimento essa base, conforme Tabela 4.6. O número de consultas feitas a base remota foi reduzida, comparada à realizada na base local, em virtude do número de acessos da API por minuto permitida na versão gratuita do Virustotal.

As tabelas de enriquecimento foram corretamente populadas e apresentaram consistência no relaciona-

Tabela 4.4: Tempo de resposta do sistema pra fluxos exclusivamente suspeitos.

Tipo de Processamento	Característica do Fluxo	Média de Pacotes	Taxa (pacotes por segundo)	Tempo de processamento (ms)
Com consulta ao Coletor	Fluxos Suspeitos	100	1 pps	1.742,77
		100	10 pps	1.674,18
		500	10 pps	1.838,37

Tabela 4.5: Tempo de resposta do sistema (com atuação do rootkit).

Tipo de Processamento	Característica do Fluxo	Pacotes Transmitidos	Tempo de processamento (ms)	Tempo de processamento com carga de 50Mbps(ms)
Sem consulta ao Coletor	Na base Temporária	54.058	0,00114	0,00100
	Resposta DNS	1.329	0,0370	0,00730
Com consulta ao Coletor	Fluxos Legítimos	684	536,15	561,26
	Fluxos Suspeitos	100	1.642,64	1.709,62

Tabela 4.6: Ações do Sistema de Enriquecimento.

Tipo de processamento	Descrição da Base de Dados	Número de Consultas	Resposta (ms)
Consulta à base local	Nó pertencente a rede TOR	100	42,18
Consulta à base remota	Plataforma VirusTotal	30	1.241,14

mento com a tabela de fluxos, realizada através de chave primária e estrangeira.

4.3.3 Desempenho do Sistema de Aviso e Geração de Bloqueio e CTI

Todos os testes de detecção de limiares previstos apresentaram resultados corretos, ou seja, o Sistema de Aviso percebeu que os limiares foram atingidos e assinalou os fluxos como bloqueados. Os arquivos necessários para bloqueio e CTI (padrões STIX2.1 e SNOR Rule) também foram gerados corretamente. Os resultados do desempenho do Sistema de Aviso estão demonstrados na Tabela 4.7. Foi possível verificar também que, em situações mais complexas (domínios que são resolvidos para vários IPs, sendo que cada IP apresenta um resultado de enriquecimento diferente), o banco de dados relacional mostrou-se uma escolha correta, pois consultas integrando todas as tabelas podem ser facilmente implementadas.

Tabela 4.7: Ações do Sistema de Aviso.

Tipo de processamento	Número de Consultas	Resposta (ms)
Sem bloqueio (limiares não atingidos)	50	9,31
Com bloqueio (limiares atingidos)	50	89,60

4.3.4 Sistema de Visualização

Todas as verificações descritas nos testes de interface do Sistema de Visualização foram realizadas e separadas em partes para melhor análise dos resultados. Primeiramente, foi verificada a visualização das tabelas de enriquecimento local e remoto, as quais estão apresentadas na Figura 4.1.

As tabelas de enriquecimento contém 8 fluxos não analisados, conforme teste previsto e dados presentes na Base de Conexões (fluxos suspeitos). O número de consultas presentes na tabela de enriquecimento remota (Virustotal) é superior ao do total de fluxos analisados, visto que alguns fluxos possuem domínios atrelados, e que gera, nesses casos, mais de uma consulta por fluxo (IP e domínios). As informações adicionais de domínio podem ser visualizadas ao selecionar um dos fluxos que possuem domínio atrelado, conforme 4.2. Os IPs e domínios apresentados são fictícios.

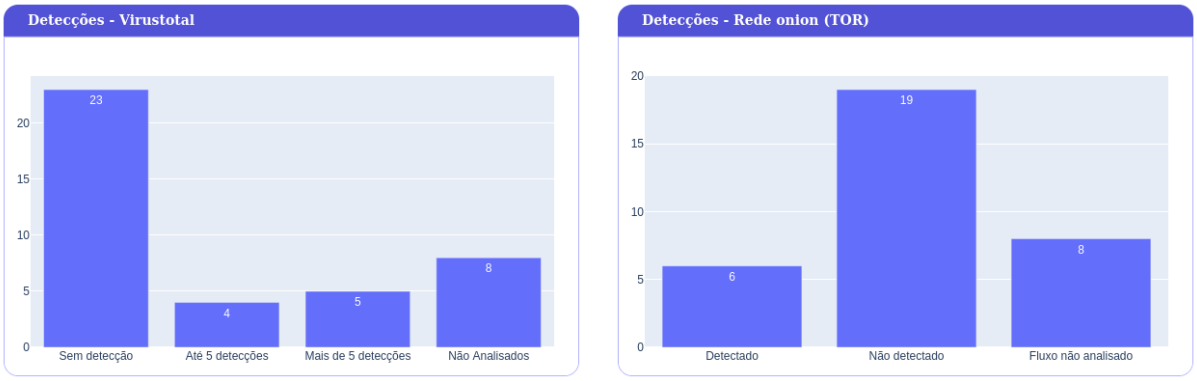


Figura 4.1: Testes de Visualização: Enriquecimento (figura do Autor)

Fluxos Suspeitos - Virustotal

DETECÇÕES MALICIOSAS	DOMINIO	IP	DATA COLETA
0		142.250.219.3	2022-07-18T10:09:58.039197
0	ocsp.pki.goog		2022-07-18T10:09:59.305376

Fluxos Suspeitos - Rede TOR

NOME	FLAGS	VERSÃO	DATA COLETA
Quetzalcoatl	EFGRSDV	Tor 0.4.7.8	2022-07-15T11:26:30.991674

Figura 4.2: Testes de Visualização: Enriquecimento de fluxo (figura do Autor)

Em seguida, foi realizada a análise de fluxo suspeito bloqueado, presente na tabela de fluxos suspeitos, como apresentado na Figura 4.3. O fluxo selecionado, contendo o IP de destino fictício *142.250.219.3*, demonstra o enriquecimento de informações de domínio consultado *ocsp.pki.goog* (também fictício). A partir desse fluxo, é possível verificar a criação de um arquivo de regra no padrão utilizado pelo SNORT, conforme Código 4.1 demonstrando uma regra que especifica qualquer origem (*any*) para o destino bloqueado, tanto para o IP de destino quanto para o nome de domínio utilizado Também é possível verificar a criação do arquivo no padrão STIX2.1, tanto em formato JSON quanto de forma interativa, a qual é apresentada na Figura 4.4, além do georreferenciamento do IP de destino, conforme Figura 4.5.

Código 4.1: Testes de Visualização: Regra SNORT

```
1    drop tcp any any -> 142.250.219.3 any (msg:"IP utilizado por rootkit com
         ofuscacao de trafego no terminal infectado";rev:1)
2    drop udp any any -> any 53 (msg:"Dominio utilizado por rootkit com ofuscacao
         de trafego no terminal infectado"; content:"|04|ocsp|03|pki|04|goog"; rev
         :1)
```

ID	IP DE ORIGEM	IP DE DESTINO	ENRIQUECIMENTO VT	ENRIQUECIMENTO TOR	BLOQUEADO	DOMÍNIOS
45	192.168.190.10	142.250.219.3	true	true	true	ocsp.pki.goog

Figura 4.3: Testes de Visualização: Fluxo bloqueado (figura do Autor)

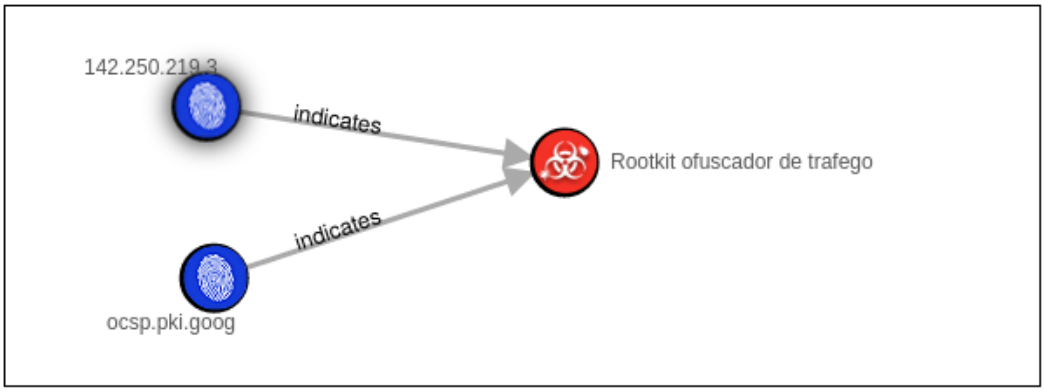


Figura 4.4: Testes de Visualização: STIX Interativo (figura do Autor)

Figura 4.5: Testes de Visualização: georreferenciamento (figura do Autor)

Código 4.2: Testes de Visualização: STIX em formato JSON (parcial)

```
{
    "type": "bundle",
    "id": "bundle--7ee39b39-c9b5-4651-a251-34ec79ef0345",
    "objects": [
        {
            "type": "malware",
            "spec_version": "2.1",
            "id": "malware--eecbc5f6-8aa9-4c55-ab2b-2753f4645fad",
            "created": "2023-04-11T14:53:06.203832Z",
            "modified": "2023-04-11T14:53:06.203832Z",
            "name": "Rootkit ofuscador de trafego",
            "description": "Malware com caracteristica de ofuscacao (...)",
            "malware_types": [
                "backdoor",
                "remote-access-trojan"
            ],
            "is_family": false
        },
        {
            "type": "indicator",
            "spec_version": "2.1",
            "id": "indicator--26d88897-8d7c-4284-aaa6-b341092ce5b7",
            "created": "2023-04-11T14:53:06.184803Z",
            "modified": "2023-04-11T14:53:06.184803Z",
            "name": "142.250.219.3",
            "description": "IPs utilizados por rootkit com ofuscacao (...)",
            "pattern": "[ipv4-addr:value = '142.250.219.3']",
            "pattern_type": "stix",
            "pattern_version": "2.1",
            "valid_from": "2023-04-11T14:53:06.184803Z",
            "labels": [
                "malicious-activity",
                "anonymization"
            ]
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--379ab978-63bb-40f3-9380-7efe2c4254de",
            "created": "2023-04-11T14:53:06.204116Z",
            "modified": "2023-04-11T14:53:06.204116Z",
            "relationship_type": "indicates",
            "source_ref": "indicator--26d88897-8d7c-4284-aaa6-b341092ce5b7",
            "target_ref": "malware--eecbc5f6-8aa9-4c55-ab2b-2753f4645fad"
        },

            (...)
```

5 CONCLUSÕES

Nesse trabalho, foi desenvolvida uma arquitetura capaz de detectar *rootkits* que utilizam técnicas de ofuscação de comunicação no terminal infectado, de forma passiva, escalável, generalizável e adaptável, sem impactos significantes no desempenho da infraestrutura de rede, com enriquecimento de dados de múltiplas fontes e incorporação das informações oriundas de múltiplos terminais, permitindo integração com outros sistemas de defesa previamente existentes, como IDS e *firewalls*. Dessa forma, agrega a organizações dados de CTI e uma nova técnica de monitoramento em tempo real de ações maliciosas.

Os testes de coleta e análise dos fluxos foram considerados satisfatórios para soluções *out-of-band* que dependem de outros fatores, como acessos a bases externas e uso de protocolos de rede, como o TCP. O desempenho do ECol é análoga à apresentada pelo NERD [9]. Tal situação era prevista em virtude de o projeto manter as mesmas características de coletor.

Os tempos de processamento de fluxos legítimos são baixos e não afetaram substancialmente memória e processamento do Sistema Operacional utilizado em laboratório. Pode ocorrer sobrecarga de dados no EAud em situações de tráfego com muitos fluxos novos contendo poucos pacotes cada, pois estes dependem de consulta ao ECol e diminuem o desempenho média da solução. O tempo médio de processamento de um pacote é de aproximadamente 100 milisegundos, considerando a divisão de pacotes executada durante os testes de laboratório (100 pacotes suspeitos e 1.329 consultas DNS em um total de 56.171 pacotes trafegados). No processamento de fluxos suspeitos com consulta ao ECol, o tempo de resposta é acrescido em aproximadamente 300%, em virtude do tratamento e atualização da Base de Conexões Suspeitas. Entretanto, ainda é baixo (inferior a 2 segundos) e, para situações reais, o número de fluxos suspeitos tente a ser insignificante no total de pacotes trafegados pela rede.

No caso de número excessivo de fluxos maliciosos, onde o Elemento Auditor necessita consultar o Coletor e, após detectada suspeição, atualizar a Base de Conexões (pior cenário em tempo de processamento), é possível que a taxa de recebimento de pacotes pelo EAud seja maior que a velocidade de processamento destes pacotes, fazendo com que a informação de fluxos maliciosos seja inserida na Base de Conexões com certo atraso. Para diminuir essa possibilidade, deve-se optar por processamento paralelo de pacotes.

O resultado apresentado nos testes de enriquecimento, que são assíncronos e não impedem o funcionamento da infraestrutura de rede existente, foram considerados satisfatórios, com tempos inferiores a 1,3 segundos, mesmo em situações que requerem uso de API remota. O tempo total entre a detecção do fluxo potencialmente malicioso e a inserção de dados de enriquecimento depende, ainda, do intervalo entre consultas definido na solução. Caso o tempo entre consultas for definido para valores superiores a 1,3 segundos, esse intervalo poderá ser mais impactante para a solução do que o tempo de resposta obtidos em consultas locais ou remotas, considerando-se condições semelhantes às apresentadas nos testes de desempenho. Portanto, o enriquecimento não causa impacto de desempenho significativo para o analista cibernético.

O resultado apresentado nos testes do Sistema de Aviso, que, assim com o Sistema de Enriquecimento, é assíncrono, foram inferiores a 90 milisegundos, mesmo em situações onde é executada criação de regras

de bloqueio e de CTI. Considerando que a arquitetura PARDED foi concebida para que o Sistema de Aviso seja executado em tempos regulares predefinidos superiores a 1 segundo, o tempo de detecção e tratamento obtidos podem ser considerados insignificantes e os resultados considerados satisfatórios.

Todos os testes realizados de visualização e utilização da interface apresentaram resultados corretos, ou seja, o Sistema de Visualização foi capaz de apresentar, de forma gráfica e interativa, as informações de cada sistema presente no Elemento Auditor, conforme previsto na arquitetura PARDED. Não foram detectados testes de desempenho necessários para validação desse sistema em virtude de ser uma interface web e depender de ações manuais do analista de segurança para verificação dos resultados apresentados e para execução de filtros ou ações específicas.

A detecção dos nomes de domínio utilizados pelos fluxos suspeitos, em conjunto com enriquecimento de dados de fontes locais e externas, além da apresentação de forma temporal e visual, adiciona capacidade de obtenção de *insights* por parte de analistas de segurança relacionados ao tráfego considerado suspeito pela solução. A geração de arquivos de CTI, em conjunto com os demais dados de fluxo coletados, como os domínios consultados e o georreferenciamento do destino das comunicações, permite análises de inteligência com maior riqueza de detalhes, mantendo o contexto do objeto da avaliação, evitando visualização de fluxos desnecessários e que poluem os ambientes de análise. Os arquivos de CTI, assim como os arquivos com regras de bloqueio baseado em limiares de detecção, permitem fácil integração com os demais sistemas de defesa presentes em uma infraestrutura de redes de comunicação.

Os resultados obtidos comprovam a possibilidade de uso do PARDED em infraestruturas de rede já existentes, sem impactos significativos. Cerca de 95% dos pacotes foram processados em menos de 0,0012 milissegundos, mesmo em sistemas com carga, sem nenhuma perda de pacotes legítimos. Os dados de enriquecimento e análises de limiares de detecção, por serem executados paralelamente, não possuem nenhuma interferência no fluxo de dados original. A solução demonstrou impacto de operação praticamente nulo em uma infraestrutura existente, considerando tráfego de dados de 50Mpbs.

Ainda, o PARDED demonstrou a viabilidade de adição de inteligência de ameaças e possibilidade de integração com demais sistemas de defesa cibernética. Entretanto, como a detecção é passiva e depende de informações coletadas e tratadas após o início do fluxo de dados analisado, ao menos os pacotes iniciais de fluxos suspeitos não sofrerão bloqueio, mesmo que sejam detectados como maliciosos. Para que as informações de CTI sejam relevantes, é necessária integração de outras bases de consultas e conhecimento da infraestrutura de rede existente. Um grande número de Elementos Coletores, mesmo que não impacte substancialmente no desempenho natural da rede, pode comprometer o Elemento Copiador, caso seja único, e gerar no Elemento Auditor a perda de pacotes ou problemas de *buffer* na captura realizada através da biblioteca `libpcap`. Mesmo que essa ação não produza, a princípio, falsos-positivos, pode gerar tráfego malicioso não detectado.

Trabalhos futuros podem melhorar o projeto de forma a alimentar os sistemas de defesa com novas informações ou tornar a detecção mais eficiente e efetiva para infraestruturas de rede existentes, como:

- Otimizar os valores de limiares utilizados para bloqueio de pacotes: infraestruturas de rede de comunicação são heterogêneas e limiares de detecção customizados para um ambiente podem ser ineficientes em outros. Por exemplo, redes que utilizam a infraestrutura de nós TOR para saída de internet

não podem utilizar essa informação para definir se um fluxo é malicioso.

- Definir estratégias no caso de uso de consultas de domínio com texto cifrado, como *DNS-over-HTTPS*: a obtenção do nome de domínio referente aos fluxos suspeitos é de grande relevância para o enriquecimento dos dados e para definição de limiares de detecção. Caso a informação de DNS não esteja em um pacote padrão do tipo "DNS Response", conforme RFC 1035 [77], o PARDED não conseguirá obter a informação. É necessária uma análise mais aprofundada de métodos alternativos que os *rootkits* possam utilizar para resolver nomes de domínios.

- Integrar técnicas de detecção auxiliares: a adição de novas capacidades para definir como malicioso tráfegos suspeitos eleva a eficiência da solução. Como exemplo, pode-se verificar padrões anômalos no conteúdo de pacotes ou analisar intervalos de transmissão de fluxos detectados como suspeitos.

- Adicionar novas bases de consulta locais e remotas: o enriquecimento eficiente dos fluxos detectados como suspeitos depende da quantidade e qualidade das bases de dados consultadas. É possível adicionar novas bases manualmente ou através motores de resposta automatizados que concentram diversas fontes, como o Cortex [78], que permite a configuração de bases de informação por campos observáveis, como IP, domínio ou url, além de respostas em formato JSON, facilmente tratadas posteriormente pelo Sistema de Enriquecimento do EAud.

- Adicionar novas técnicas de enriquecimento de dados de fluxo de rede: além da adição de novas bases de dados, técnicas adicionais de enriquecimento podem melhorar os resultados apresentados ao analista de segurança cibernética. Por exemplo, histórico de enriquecimentos, reavalidação temporal da relação IP/domínio ou o armazenamento do conteúdo dos fluxos considerados suspeitos para melhorar a análise de TTPs de possíveis atores maliciosos.

- Verificar os *playloads* transmitidos por fluxos suspeitos em bases de *hashs* maliciosos: essa análise é costumeiramente realizada por ferramentas tradicionais de defesa e pode ser valiosa para detecção mais precisa de fluxos maliciosos, aumentando assim a granularidade dos limiares de detecção.

É interessante também verificar o comportamento de grande número de Elementos Coletores em uma infraestrutura real, analisando possíveis gargalos e pontos de melhoria. Seria possível, ainda, elevar o número de informações de enriquecimento através da integração com outras ferramentas de CTI, como o Mitre ATT&CK, e analisar outros protocolos de comunicação que podem ser utilizados para ofuscação de dados, como o UDP ou ICMP.

REFERÊNCIAS BIBLIOGRÁFICAS

1 AV-Test. *Av-atlas, cyber threat situation and warns portal*. 2023. Disponível em: <https://portal.av-atlas.org>. Acesso em: 20 Fev 2023.

2 PETAVRATZI, E. *Transforming data and information into knowledge within the MICA project*. 2017. Disponível em: <https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5b5fefbb8&appId=PPGMS>. Acesso em: 04 abr 2023.

3 United States Government Us Army. *Joint Publication Jp 2-0 Joint Intelligence 22 June 2007*. [S.l.]: Createspace Independent Publishing Platform, 2013.

4 OASIS TC. *OASIS Cyber Threat Intelligence (CTI) TC*. Online: [s.n.], 2023. Disponível em: <https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=cti>. Acesso em: 4 abr 2023.

5 BROMANDER, S.; JØSANG, A.; EIAN, M. Semantic cyberthreat modelling. *STIDS*, p. 74–78, 2016.

6 TELEA, A. C. *Data visualization, Principles and Practice*. 2. ed. Boca Raton, FL: CRC Press, 2014. ISBN 978-1-4665-8527-0.

7 EGELE, M.; SCHOLTE, T.; KIRDA, E.; KRUEGEL, C. A survey on automated dynamic malware-analysis techniques and tools. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 44, n. 2, p. 1–42, 2008.

8 MARQUES, R. S.; EPIPHANIOU, G.; AL-KHATEEB, H.; MAPLE, C.; HAMMOUDEH, M.; De Castro, P. A. L.; DEHGHANTANHA, A.; CHOO, K. K. R. A Flow-based Multi-Agent Data Exfiltration Detection Architecture for Ultra-low Latency Networks. *ACM Transactions on Internet Technology*, v. 21, n. 4, 2021. ISSN 15576051.

9 TERRA, M. B.; GONDIM, J. J. NERD: A Network Exfiltration Rootkit Detector based on a Multi-agent Artificial Immune System. *2021 Workshop on Communication Networks and Power Systems, WCNPS 2021*, 2021.

10 ITU. *Individuals using the Internet*. 2021. Disponível em: <https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx>. Acesso em: 4 Jul 2022.

11 COSTA, H.; NICOLETTI, G.; PISU, M.; Von Rueden, C. Are digital platforms killing the offline star? Platform diffusion and the productivity of traditional firms. *OECD Economics Department Working Papers*, OECD, v. 1, n. 1682, p. 1–27, 2021. Disponível em: <https://doi.org/10.1787/18151973>.

12 European Union Agency for Cybersecurity. *The year in review - ENISA Threat Landscape*. 2020. 1–25 p. Disponível em: <www.enisa.europa.eu/publications/year-in-review/@@download/fullReport>. Acesso em: 4 Abr 2023.

13 MORGAN, S. Cybercrime To Cost The World $10.5 Trillion Annually By 2025. *Cybersecurity Ventures*, p. 1, 2020. Disponível em: <https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/>. Acesso em: 20 Jul 2022.

14 SOWINSKI, D. *The Growing Danger of Data Exfiltration by Third-Party Web Scripts*. 2022. Disponível em: <https://securityintelligence.com/posts/growing-danger-data-exfiltration-third-party-web-scripts/>. Acesso em: 04 Set 2022.

15 BASIN, D. *The cyber security body of knowledge*. University of Bristol, ch. Formal Methods for Security, version..[Online], 2021. Disponível em: <https://www.cybok.org>.

16 ZHENG, Y.; LI, Z.; XU, X.; ZHAO, Q. Dynamic defenses in cyber security: Techniques, methods and challenges. *Digital Communications and Networks*, v. 8, n. 4, p. 422–435, 2022. ISSN 2352-8648. Disponível em: <https://www.sciencedirect.com/science/article/pii/S235286482100047X>.

17 Global Research & Analysis Team. *Advanced Threat predictions for 2021*. 2020. Disponível em: <https://securelist.com/apt-predictions-for-2021/99387/>. Acesso em: 20 Fev 2023.

18 POLAT, G.; SODAH, F. Security issues in IoT: Challenges and countermeasures. *Isaca journal*, p. 1–7, 2019.

19 MADAN, S.; SOFAT, S.; BANSAL, D. Tools and techniques for collection and analysis of internet-of-things malware: A systematic state-of-art review. *Journal of King Saud University - Computer and Information Sciences*, v. 34, n. 10, Part B, p. 9867–9888, 2022. ISSN 1319-1578. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1319157821003621>.

20 TRINKS, M.; GONDIM, J.; ALBUQUERQUE, R. Multi-agent architecture for passive rootkit detection with data enrichment. In: *CSEI: International Conference on Computer Science, Electronics and Industrial Engineering (CSEI)*. Cham: Springer Nature Switzerland, 2023. p. 29–41. ISBN 978-3-031-30592-4.

21 JULIAN, V.; BOTTI, V. Multi-Agent Systems. *Applied Sciences (Switzerland)*, v. 9, n. 7, 2019. ISSN 20763417.

22 WOOLDRIDGE, M. *An introduction to multiagent systems*. England: John wiley & sons, 2009.

23 FERBER, J.; WEISS, G. *Multi-agent systems: an introduction to distributed artificial intelligence*. [S.l.]: Addison-wesley Reading, 1999. v. 1.

24 KNAPP, E. D.; LANGILL, J. T. Exception, anomaly, and threat detection. *Industrial Network Security*, Syngress, Boston, p. 323–350, 2015.

25 ALLEN, M.; CERVO, D. *Multi-Domain Master Data Management*. Waltham, MA, USA: Elsevier, 2015. ISBN 978-0-12-800835-5.

26 ABU, M. S.; SELAMAT, S. R.; ARIFFIN, A.; YUSOF, R. Cyber threat intelligence - issue and challenges. *Indonesian Journal of Electrical Engineering and Computer Science*, v. 10, n. 1, p. 371–379, 2018.

27 KOTSIAS, J.; AHMAD, A.; SCHEEPERS, R. Adopting and integrating cyber-threat intelligence in a commercial organisation. *European Journal of Information Systems*, Taylor & Francis, v. 32, n. 1, p. 35–51, 2023.

28 BRASIL. Lei nº 9.883, de 7 de dezembro de 1999. institui o sistema brasileiro de inteligência, cria a agência brasileira de inteligência - abin, e dá outras providências. *Diário Oficial da República Federativa do Brasil*, Brasília, DF, 1999. Disponível em: <https://www.planalto.gov.br/ccivil_03/leis/l9883.htm>.

29 TOUNSI, W.; RAIS, H. A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & Security*, Elsevier, v. 72, p. 212–233, jan. 2018.

30 LEITE, C.; HARTOG, J.; SANTOS, D. dos; COSTANTE, E. Actionable cyber threat intelligence for automated incident response. In: ____. [S.l.: s.n.], 2023. p. 368–385. ISBN 978-3-031-22294-8.

31 MITRE. *MITRE ATT&CK*. Online: [s.n.], [s.d]. Disponível em: <https://makingsecuritymeasurable.mitre.org/docs/stix-intro-handout.pdf>. Acesso em: 4 abr 2023.

32 MITRE. *Structured Threat Information eXpression - STIX*. Online: [s.n.], 2023. Disponível em: <https://attack.mitre.org/>. Acesso em: 4 abr 2023.

33 U.S. Department of Energy. *Cybersecurity Capability Maturity Model, Version 2.1*. 2022. Disponível em: <https://www.energy.gov/sites/default/files/2022-06/C2M2\%20Version\%202.1\%20June\%202022.pdf>. Acesso em: 29 abr 2023.

34 STILLIONS, R. *The DML model*. 2014. Disponível em: <http://ryanstillions.blogspot.com/2014/04/the-dml-model_21.html>. Acesso em: 01 mar 2023.

35 DINGLEDINE, R.; MATHEWSON, N.; SYVERSON, P. *Tor: The second-generation onion router*. [S.l.], 2004.

36 REED, M.; SYVERSON, P.; GOLDSCHLAG, D. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, v. 16, n. 4, p. 482–494, 1998. ISSN 07338716.

37 Cybersecurity and Infrastructure Security Agency. *Defending Against Malicious Cyber Activity Originating from Tor*. 2020. Disponível em: <https://www.cisa.gov/news-events/cybersecurity-advisories/aa20-183a>. Acesso em: 20 Mar 2023.

38 LACOMBE, E.; RAYNAL, F.; NICOMETTE, V. Rootkit modeling and experiments under linux. *Journal in Computer Virology*, Springer, v. 4, p. 137–157, 2008.

39 SAHU, A.; TARODIA, A.; JAGTAP, S.; GUNDLA, M. R. A survey on rootkit techniques. *International Journal of Innovative Research in Science, Engineering and Technology*, v. 10, p. 4242–4246, 2021.

40 Veracode. *Rootkit*. 2023. Disponível em: <www.veracode.com/security/rootkit>. Acesso em: 05 fev 2023.

41 LEE, H.; SONG, C.; KANG, B. B. Lord of the x86 rings: A portable user mode privilege separation architecture on x86. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. [S.l.: s.n.], 2018. p. 1441–1454.

42 SUBAIRU, S.; ALHASSAN, J.; SANJAY, M. Evaluating capabilities of rootkits tools. International Journal of Advanced Multidisciplinary Research and Studies . . . , 2016.

43 OR-MEIR, O.; NISSIM, N.; ELOVICI, Y.; ROKACH, L. Dynamic malware analysis in the modern era—a state of the art survey. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 52, n. 5, p. 1–48, 2019.

44 THAKKAR, J. *What Is a Rootkit and How Does It Work?* 2020. Disponível em: <www.thesslstore.com/blog/what-is-a-rootkit-and-how-does-it-work>. Acesso em: 05 fev 2023.

45 CHEN, M.; FLORIDI, L.; BORGO, R. What is visualization really for? In: *The Philosophy of Information Quality*. [S.l.]: Springer, 2014. p. 75–93.

46 WILLIAMS, J.; SOCHATS, K.; MORSE, E. Visualization, volume 30, pages 161–207. *Information Today*, 1995.

47 EJAZ, S.; NOOR, U.; RASHID, Z. Visualizing interesting patterns in cyber threat intelligence using machine learning techniques. *Cybernetics and Information Technologies*, v. 22, n. 2, p. 96–113, 2022.

48 NAGY, R.; NÉMETH, K.; PAPP, D.; BUTTYÁN, L. Rootkit detection on embedded iot devices. *Acta Cybernetica*, University of Szeged, v. 25, n. 2, p. 369–400, 2021.

49 RAMANI, R. G.; KUMAR, S. S. Nonvolatile kernel rootkit detection using cross-view clean boot in cloud computing. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, v. 33, n. 3, p. e5239, 2021.

50 ARNOLD, T. M. *A comparative analysis of rootkit detection techniques*. [S.l.]: University of Houston-Clear Lake, 2011.

51 McAfee Inc. *McAfee Rootkit Remover*. Versão 0.8.9.209. Online, 2015. Disponível em: <bleepingcomputer.com/download/mcafee-labs-rootkit-remover>. Acesso em: 25 mar 2023.

52 Malwarebytes. *Malwarebytes Anti-Rootkit Scanner*. Versão 4.5.21.305. Online, 2023. Disponível em: <www.malwarebytes.com/solutions/rootkit-scanner>. Acesso em: 12 fev 2023.

53 European Union Agency for Cybersecurity. *Rootkits*. [s.d]. Disponível em: <www.enisa.europa.eu/topics/incident-response/glossary/rootkits>. Acesso em: 06 fev 2023.

54 Microsoft Corporation. *RootkitRevealer*. Online: [s.n.], 2006. Disponível em: <learn.microsoft.com/en-us/sysinternals/downloads/rootkit-revealer>. Acesso em: 23 mar 2023.

55 HAUGWITZ, H. von. *Advanced Intrusion Detection Environment (AIDE)*. Online: [s.n.], 2023. Disponível em: <aide.github.io/>. Acesso em: 19 mar 2023.

56 Samhain Labs. *The Samhaim File Integrity Host-based Intrusion Detection System*. Online: [s.n.], 2022. Disponível em: <www.la-samhna.de/samhain>. Acesso em: 25 mar 2023.

57 Springer Nature Switzerland. *Springer Link*. 2023. Disponível em: <https://link.springer.com/>. Acesso em: 04 Maio 2023.

58 IEEE. *IEEE Xplore*. 2023. Disponível em: <https://ieeexplore.ieee.org>. Acesso em: 26 Fev 2023.

59 Association for Computing Machinery. *ACM Digital Library*. 2023. Disponível em: <https://dl.acm.org/>. Acesso em: 26 Fev 2023.

60 Google LLC. *Google Acadêmico*. 2023. Disponível em: <https://scholar.google.com>. Acesso em: 6 Fev 2023.

61 JUNNILA, J. *Effectiveness of linux rootkit detection tools*. 68 p. Dissertação (Mestrado em Ciência do Processamento da Informação) — University of Oulu, Oulu, Finlândia, 2020.

62 WANG, X.; ZHANG, J.; ZHANG, A.; REN, J. Tkrd: Trusted kernel rootkit detection for cybersecurity of vms based on machine learning and memory forensic analysis. *Mathematical Biosciences and Engineering*, v. 16, n. 4, p. 2650–2667, 2019.

63 PHAM, D.-P.; MARION, D.; HEUSER, A. Ultra: Ultimate rootkit detection over the air. In: *25th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2022),*. [S.l.: s.n.], 2022.

64 JIANG, X.; LORA, M.; CHATTOPADHYAY, S. Efficient and trusted detection of rootkit in iot devices via offline profiling and online monitoring. In: *Proceedings of the 2020 on Great Lakes Symposium on VLSI*. [S.l.: s.n.], 2020. p. 433–438.

65 MAOUDA, I. *Hunting Rootkits with eBPF: Detecting Linux Syscall Hooking Using Tracee*. 2022. Disponível em: <https://blog.aquasec.com/linux-syscall-hooking-using-tracee>. Acesso em: 12 abr 2023.

66 eBPF. *eBPF Documentation*. 2022. Disponível em: <https://ebpf.io/what-is-ebpf/>. Acesso em: 12 abr 2023.

67 TALUKDER, S.; TALUKDER, Z. A survey on malware detection and analysis tools. *International Journal of Network Security & Its Applications (IJNSA) Vol*, v. 12, 2020.

68 PassMark Software Pty Ltd. *CPU Benchmarks*. 2023. Disponível em: <www.cpubenchmark.net>. Acesso em: 05 fev 2023.

69 PassMark Software Pty Ltd. *About PassMark Software*. 2023. Disponível em: <www.passmark.com/about/index.php>.

70 ROESCH, M. *SNORT Network Intrusion Detection & Prevention System*. Online: [s.n.], 2023. Disponível em: <https://www.snort.org/>. Acesso em: 4 abr 2023.

71 Plotly Technologies Inc. *Dash*. Online: [s.n.], 2023. Disponível em: <plotly.com/dash>. Acesso em: 19 mar 2023.

72 OASIS TC. *CTI STIX Visualization*. Online: [s.n.], 2022. Disponível em: <https://github.com/oasis-open/cti-stix-visualization/>. Acesso em: 4 abr 2023.

73 Maxmind. *GeoLite2 Free Geolocation Data*. 2023. Disponível em: <https://dev.maxmind.com/geoip/geolite2-free-geolocation-data?lang=en>. Acesso em: 24 Fev 2023.

74 SCHäLLIBAUM, J. A. *Nuk3 Gh0st*. 2019. Disponível em: <https://github.com/juanschallibaum/Nuk3Gh0st>. Acesso em: 10 mar 2023.

75 DAN. *Tor Node List*. 2022. Disponível em: <https://www.dan.me.uk>. Acesso em: 12 jul 2022.

76 Chronicle Security. *Virustotal*. 2023. Disponível em: <http://www.virustotal.com>. Acesso em: 15 fev 2023.

77 MOCKAPETRIS, P. *Domain Names - Implementation and Specification*. [S.l.], 1987. Disponível em: <https://datatracker.ietf.org/doc/html/rfc1035>.

78 TheHive Project. *Cortex: Powerful Observable Analysis and Active Response Engine*. 2023. Disponível em: <https://thehive-project.org/#section_cortex>. Acesso em: 6 Jul 2023.

I.1 ANEXO 1 - BASE DE DADOS - TABELA FLUXOS

```
-- PostgreSQL database dump - from database version 13.7 (Debian 13.7-1.pgdg110+1)

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

SET default_tablespace = '';

SET default_table_access_method = heap;

-- Name: fluxosDetectados; Type: TABLE; Schema: public; Owner: datalab

CREATE TABLE public."fluxosDetectados" (
    id integer NOT NULL,
    "ipOrigem" inet,
    "ipDestino" inet,
    "enriqVT" boolean DEFAULT false NOT NULL,
    "enriqTOR" boolean DEFAULT false NOT NULL,
    quantidade smallint DEFAULT 0 NOT NULL,
    bloqueado boolean DEFAULT false NOT NULL,
    chave character varying(30),
    data timestamp without time zone
);

ALTER TABLE public."fluxosDetectados" OWNER TO datalab;

-- Name: fluxosDetectados_id_seq; Type: SEQUENCE; Schema: public; Owner: datalab

CREATE SEQUENCE public."fluxosDetectados_id_seq"
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public."fluxosDetectados_id_seq" OWNER TO datalab;


ALTER SEQUENCE public."fluxosDetectados_id_seq" OWNED BY public."fluxosDetectados".id;

-- Name: fluxosDetectados id; Type: DEFAULT; Schema: public; Owner: datalab

ALTER TABLE ONLY public."fluxosDetectados" ALTER COLUMN id SET DEFAULT
nextval('public."fluxosDetectados_id_seq"'::regclass);

-- Name: fluxosDetectados fluxosDetectados_pkey; Type: CONSTRAINT; Schema: public;
Owner: datalab

ALTER TABLE ONLY public."fluxosDetectados"
    ADD CONSTRAINT "fluxosDetectados_pkey" PRIMARY KEY (id);


```

I.2 ANEXO 2 - BASE DE DADOS - TABELA DOMÍNIOS

```
1  -- PostgreSQL database dump - from database version 13.7 (Debian
   13.7-1.pgdg110+1)
2
3  SET statement_timeout = 0;
4  SET lock_timeout = 0;
5  SET idle_in_transaction_session_timeout = 0;
6  SET client_encoding = 'UTF8';
7  SET standard_conforming_strings = on;
8  SELECT pg_catalog.set_config('search_path', '', false);
9  SET check_function_bodies = false;
10 SET xmloption = content;
11 SET client_min_messages = warning;
12 SET row_security = off;
13
14 SET default_tablespace = '';
15
16 SET default_table_access_method = heap;
17 --
18 -- Name: dominios; Type: TABLE; Schema: public;
19 --
20 CREATE TABLE public.dominios (
21     fk_id integer DEFAULT nextval('public."enriquecimentoVT_id_seq"'::regclass)
   NOT NULL,
22     nome character varying(254),
23     data timestamp without time zone
24 );
25
26 ALTER TABLE public.dominios OWNER TO datalab;
27 --
28
```

I.3 ANEXO 3 - BASE DE DADOS - TABELA ENRIQUECIMENTO (VT)

```
-- PostgreSQL database dump - from database version 13.7 (Debian 13.7-1.pgdg110+1)

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

SET default_tablespace = '';

SET default_table_access_method = heap;

-- Name: enriquecimentoVT; Type: TABLE; Schema: public; Owner: datalab

CREATE TABLE public."enriquecimentoVT" (
    fk_id integer NOT NULL,
    deteccao smallint DEFAULT 0,
    data timestamp without time zone,
    ip inet,
    dominio character varying(254)
);

ALTER TABLE public."enriquecimentoVT" OWNER TO datalab;

-- Name: TABLE "enriquecimentoVT"; Type: COMMENT; Schema: public; Owner: datalab

COMMENT ON TABLE public."enriquecimentoVT" IS 'dados do virustotal';

-- Name: enriquecimentoVT_id_seq; Type: SEQUENCE; Schema: public; Owner: datalab

CREATE SEQUENCE public."enriquecimentoVT_id_seq"
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public."enriquecimentoVT_id_seq" OWNER TO datalab;

-- Name: enriquecimentoVT_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
datalab

ALTER SEQUENCE public."enriquecimentoVT_id_seq" OWNED BY
public."enriquecimentoVT".fk_id;

-- Name: enriquecimentoVT fk_id; Type: DEFAULT; Schema: public; Owner: datalab

ALTER TABLE ONLY public."enriquecimentoVT" ALTER COLUMN fk_id SET DEFAULT
nextval('public."enriquecimentoVT_id_seq"'::regclass);

-- Name: enriquecimentoVT fk_id_fluxo; Type: FK CONSTRAINT; Schema: public; Owner:
datalab

ALTER TABLE ONLY public."enriquecimentoVT"
    ADD CONSTRAINT fk_id_fluxo FOREIGN KEY (fk_id) REFERENCES
public."fluxosDetectados"(id) ON UPDATE CASCADE ON DELETE CASCADE;
```

I.4 ANEXO 4 - CÓDIGO - SISTEMA DE ANÁLISE INICIAL

```
1  //OBS: como o código é muito extenso, será apenas apresentada a extrutura utilizada
2
3  //BIBLIOTECAS
4
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <unistd.h>
8  #include <signal.h>
9  #include <linux/ip.h>
10 #include <linux/tcp.h>
11 #include <linux/udp.h>
12 #include <pthread.h>
13 #include <etcdlib.h>
14 #include <arpa/inet.h>
15 #include <postgresql/libpq-fe.h>
16 #include <string.h>
17 #include <pcap.h>
18
19 //DEFINICOES para calculos e cabecalhos de rede
20
21 #define iphdr(x)    ((struct iphdr *)(x))
22 #define tcphdr(x)   ((struct tcphdr *)(x))
23 #define udphdr(x)   ((struct udphdr *)(x))
24 #define PACKETS_PER_CYCLE 200
25 #define BLOCKED_THRESHOLD 5
26 #define PACKET_LIMIT 10000
27 #define BILLION  1000000000.0
28 #define MILLION  1000000.0
29
30 #define ETHER_HEADER 14
31
32 #define DNS_LIMIT 4200   //maximo de linhas da tabela temporaria de DNS. Acima disso, cor
   sobreescrever
33 #define BASECHAVESLOCAL_LIMIT 2000
34 #define MAX_CHAVE 28
35
36 #define T_A 1 //Ipv4 address
37 #define T_NS 2 //Nameserver
38 #define T_CNAME 5 // canonical name
39 #define T_SOA 6 /* start of authority zone */
40 #define T_PTR 12 /* domain name pointer */
41 #define T_MX 15 //Mail server
42
43 typedef struct blockedCount {
44     __be32 address;
45     uint8_t count;
46 } blockedCount_t;
47
48 static struct nlif_handle* interfaceHandle;
49 static struct nfq_handle *nfqHandle;
50 pthread_mutex_t verdictMutex = PTHREAD_MUTEX_INITIALIZER;
51 pthread_mutex_t blockCountMutex = PTHREAD_MUTEX_INITIALIZER;
52
53 static blockedCount_t *blocked;
54 static uint8_t blockedMaxIndex = 0;
55 //static uint8_t packetIndex = 0;
56 static volatile unsigned int processedPackets = 0;
57 static volatile unsigned int blockedPackets = 0;
58
59 PGconn *conn;
60
61 FILE *out_file;
```

```
62
63  static char base_temporaria[BASECHAVESLOCAL_LIMIT][MAX_CHAVE+1];
64  static short int base_temporaria_indice = 0;
65
66  typedef struct {
67      unsigned char nome[150];
68      __be32 ip;
69  } tuple_dns;
70
71  static tuple_dns lista_dns[DNS_LIMIT];
72  static int lista_dns_last=0;
73  static int lista_dns_first=0;
74
75  static volatile int interrupt = 0;
76  pcap_t *handle;
77
78  /* DNS:
79  -----------------
80      inicios das estruturas e funcoes DNS
81  -----------------
82  */
83  struct DNS_HEADER
84  {
85      unsigned short id; // identification number
86
87      unsigned char rd :1; // recursion desired
88      unsigned char tc :1; // truncated message
89      unsigned char aa :1; // authoritive answer
90      unsigned char opcode :4; // purpose of message
91      unsigned char qr :1; // query/response flag
92
93      unsigned char rcode :4; // response code
94      unsigned char cd :1; // checking disabled
95      unsigned char ad :1; // authenticated data
96      unsigned char z :1; // its z! reserved
97      unsigned char ra :1; // recursion available
98
99      unsigned short q_count; // number of question entries
100     unsigned short ans_count; // number of answer entries
101     unsigned short auth_count; // number of authority entries
102     unsigned short add_count; // number of resource entries
103 };
104
105 struct QUESTION
106 {
107     unsigned short qtype;
108     unsigned short qclass;
109 };
110
111 #pragma pack(push, 1)
112 struct R_DATA
113 {
114     unsigned short type;
115     unsigned short _class;
116     unsigned int ttl;
117     unsigned short data_len;
118 };
119 #pragma pack(pop)
120
121 struct RES_RECORD
122 {
123     unsigned char *name;
```

```
124     struct R_DATA *resource;
125     unsigned char *rdata;
126 };
127 
128 //Structure of a Query
129 typedef struct
130 {
131     unsigned char *name;
132     struct QUESTION *ques;
133 } QUERY;
134 
135 struct RES_IPV4
136 {
137     unsigned int ipv4;
138 };
139 
140 u_char* ReadName(unsigned char* reader,unsigned char* buffer,int* count)
141 {
142     unsigned char *name;
143     unsigned int p=0,jumped=0,offset;
144     int i , j;
145 
146     *count = 1;
147     name = (unsigned char*)malloc(256);
148 
149     name[0]='\0';
150 
151     //read the names in 3www6google3com format
152     while(*reader!=0)
153     {
154         if(*reader>=192)
155         {
156             offset = (*reader)*256 + *(reader+1) - 49152; //49152 = 11000000 00000000 ;)
157             reader = buffer + offset - 1;
158             jumped = 1; //we have jumped to another location so counting wont go up!
159         }
160         else
161         {
162             name[p++]=*reader;
163         }
164 
165         reader = reader+1;
166 
167         if(jumped==0)
168         {
169             *count = *count + 1; //if we havent jumped to another location then we can c
170         }
171     }
172 
173     name[p]='\0'; //string complete
174     if(jumped==1)
175     {
176         *count = *count + 1; //number of steps we actually moved forward in the packet
177     }
178 
179     //now convert 3www6google3com0 to www.google.com - transforma padrao DNS para padrao
180             //    www.
181     for(i=0;i<(int)strlen((const char*)name);i++)
182     {
183         p=name[i];
184         for(j=0;j<(int)p;j++)
185         {
```

```
            name[i]=name[i+1];
            i=i+1;
        }
        name[i]='.';
    }
    name[i-1]='\0'; //remove the last dot
    return name;
}
/*
------------
    fim das estruturas DNS
------------
*/
void stopExecution(int _){
    printf("Received SIGINT!\n");
    interrupt = 1;
    pcap_breakloop(handle);
    pcap_close(handle);
    handle = NULL;
}

/*static void listTuples(void) {
    //printf("==========\nTuple count is %d\n", lista_dns_last-lista_dns_first);
    //for (int i = lista_dns_first; i < lista_dns_last; ++i)
    //    printf("   [%s] -> %d\n", lista_dns_count[i].nome, lista_dns_count[i].ip);
    //puts("==========");
}*/

static void addIPDNS(unsigned char *str, __be32 address) {
    if (sizeof(str)>150){
        strncpy(lista_dns[lista_dns_last].nome, str,49);
        lista_dns[lista_dns_last].nome[49]='\0';
    }else
        strcpy(lista_dns[lista_dns_last].nome, str);

    //printf("Adding '%s', mapped to %d\n", str, val);
    lista_dns[lista_dns_last].ip = address;
    lista_dns_last++;

    if (lista_dns_last>= DNS_LIMIT)  //rotaciona
        lista_dns_last = 0;

    if (lista_dns_last == lista_dns_first){ //o inicial vai ser sobrescrito pelo final. /
        lista_dns_first++;
        if (lista_dns_first>= DNS_LIMIT) //rotaciona
            lista_dns_first = 0;
    }
}

/*static void deleteTuple(char *str) {
    int index = 0;
    while (index < tupleCount) {
        if (strcmp(str, tuple[index].strVal) == 0) break;
        ++index;
    }
    if (index == tupleCount) return;

    printf("Deleting '%s', mapped to %d\n", str, tuple[index].intVal);
    if (index != tupleCount - 1) {
        strcpy(tuple[index].strVal, tuple[tupleCount - 1].strVal);
        tuple[index].intVal = tuple[tupleCount - 1].intVal;
    }
```

```
1  //funçao principal
2
3  int main(int argc, char **argv){
4
5      char dev[] = "enp0s8\0";  //nome da interface
6      char errbuf[PCAP_ERRBUF_SIZE];
7
8      //filtros de pacotes (padrao tcpdump) - nesse exemplo, o Sistema de Analise Inicial
   observa apenas pacotes TCP/80 e UDP/53
9      //por padrao, o sistema analisa qualquer pacote TCP e UDP
10     char filter_string[] = "tcp port 80 or udp src port 53\0";
11     //char filter_string[] = "tcp or udp\0";
12     struct bpf_program fp;
13     /***
14      *    verifica se é posśivel finalizar a execução (CTRL+C). Será necessário para
   finalizar a captura
15     ***/
16     if(signal(SIGINT, stopExecution)!=0){
17         fprintf(stderr, "Unable to catch SIGINT!");
18         exit(3);
19     }
20     /***
21      *     inicia banco de dados
22      ***/
23     conn = PQconnectdb("host=localhost user=xxx password=xxx dbname=xxx");
24     if (PQstatus(conn) == CONNECTION_OK) {
25        printf("Conexão no Banco de dados com efetuada com sucesso.\n");
26     }
27     else{
28       printf("Falha na conexão ao Banco de Dados.\n");
29       PQfinish(conn);
30       exit(4);
31     }
32
33     //para salvar as informacoes de tempo de processamento (modo DEBUG)
34     if (DEBUG_TP)
35         out_file = fopen("auditor.results", "w");
36
37     /***
38      *     inicia sistema de captura de pacotes PCAP
39      ***/
40     handle = pcap_open_live(dev, BUFSIZ, 1, 0, errbuf);
41     if (handle == NULL) {
42         fprintf(stderr, "Couldn't open device %s: %s\n", dev, errbuf);
43         return(2);
44     }
45     if (pcap_compile(handle, &fp, filter_string, 0, PCAP_NETMASK_UNKNOWN) == -1) {
46         fprintf(stderr, "Couldn't parse filter %s: %s\n", filter_string,
   pcap_geterr(handle));
47         return(2);
48     }
49     if (pcap_setfilter(handle, &fp) == -1) {
50         fprintf(stderr, "Couldn't install filter %s: %s\n", filter_string,
   pcap_geterr(handle));
51         return(2);
52     }
53
54     printf("iniciando captura de pacotes.\n");
55     sleep(2);
56
57     pcap_loop(handle,0,process_pkt_sniffer,NULL);
58
59     PQfinish(conn);
```

I.5 ANEXO 5 - CÓDIGO - SISTEMA DE ENRIQUECIMENTO

```
import readTorList
import requests
import sys
import psycopg2
import time
import datetime

TEMPO_RODADA = 120

def buscaTOR(valor,tipo="ip"):

    start_time = datetime.datetime.now()

    resp=readTorList.buscaIP(valor)
    print(valor,resp)

    end_time = datetime.datetime.now()

    time_diff = (end_time - start_time)
    execution_time = time_diff.total_seconds() * 1000

    print("Tempo TOR:",execution_time)

    return resp

#-------------------------------------------------------------------------------------------

def buscaVT(valor, tipo="ip"):

    start_time = datetime.datetime.now()

    resp = {"erro":0}
    headers = {'x-apikey' : '[inserir_a_api_key_aqui]'}

    if tipo == "ip":
        r = requests.get('https://www.virustotal.com/api/v3/ip_addresses/'+valor, header
    else:
        r = requests.get('https://www.virustotal.com/api/v3/domains/'+valor, headers=hea

    if r.ok:
        d = r.json()
        if 'attributes' in d['data']:
            result = d['data']['attributes']['last_analysis_stats']
            for key,data in result.items():
                print(key,"-",data)
            resp['maliciosos']=result['malicious']
            resp['suspeitos']=result['suspicious']
        else:
            print('resposta invalida - sem atributos')
            resp['erro'] = 1

    else:
        if (r.status_code == "404"):
            print("erro da api VirusTotal. Informacao nao econtrada")
            resp['erro'] = 2
        if (r.status_code == "400"):
            print("erro da api VirusTotal. Numero de consultas ultrapassou o permitido p
            resp['erro'] = 3

    end_time = datetime.datetime.now()

    time_diff = (end_time - start_time)
```

```
63      execution_time = time_diff.total_seconds() * 1000
64
65      print("Tempo VT:",execution_time)
66      return resp
67
68  #-------------------------------------------------------------------------------------
69
70  def iniciar_robo():
71      con = psycopg2.connect(host='localhost', database='xxx', user='xxx', password='xxx')
72      cur = con.cursor()
73
74      print("Robo iniciado:")
75
76      while(True):
77
78          list_ids_enriquecidos = []
79
80          cur.execute('select id,"ipDestino","enriqVT","enriqTOR","dominios".nome as domin
    "fluxosDetectados" LEFT JOIN "dominios" ON "fluxosDetectados".id = "dominios".fk_id')
81          recset = cur.fetchall()
82
83          for rec in recset:
84              if rec[2] == False:  #enriqVT nao foi feito, inicia enriquecimento
85                  print(rec[0],"nao enriquecido no Virustotal. Iniciar processo...")
86                  if rec[1] not in list_ids_enriquecidos: #significa que precisa enriquece
87                      resp = buscaVT(rec[1],"ip")
88                      if resp['erro']==0: #sem erro, adicionar na base de dados
89                          print("enriquecer o id",rec[0],", IP:",rec[1])
90                          cur.execute('INSERT INTO "enriquecimentoVT" ("fk_id","deteccao",
    VALUES({},{},\'{}\',NOW())'.format(rec[0],resp['maliciosos'],rec[1]))
91                          con.commit()
92                          list_ids_enriquecidos.append(rec[0]) #esse id foi enriquecido
93                  if rec[4] != None: #significa que existe um dominio a ser enriquecido
94                      resp = buscaVT(rec[4],"dominio")
95                      if resp['erro']==0: #sem erro, adicionar na base de dados
96                          print("enriquecer o id",rec[0],", dominio:",rec[4])
97                          cur.execute('INSERT INTO "enriquecimentoVT" ("fk_id","deteccao",
    "data") VALUES({},{},\'{}\',NOW())'.format(rec[0],resp['maliciosos'],rec[4]))
98                          con.commit()
99                          if rec[1] not in list_ids_enriquecidos:
100                             list_ids_enriquecidos.append(rec[0]) #esse id foi enriquecid
101                 print("esperar 22 segundos:")
102                 time.sleep(22)
103         for ids in list_ids_enriquecidos:
104             cur.execute('UPDATE "fluxosDetectados" SET "enriqVT" = True WHERE id = {}'.f
105             con.commit()
106
107         cur.execute('select id,"ipDestino","enriqTOR" from "fluxosDetectados";')
108         recset = cur.fetchall()
109
110         for rec in recset:
111                 if rec[2] == False:  #enriqTOR nao foi feito, inicia enriquecimento
112                     print(rec[0],"nao enriquecido no Tor. Iniciar processo...")
113                     resp = buscaTOR(rec[1])
114                     if resp['erro']==0: #sem erro, adicionar na base de dados
115                         cur.execute('INSERT INTO "enriquecimentoTOR" ("fk_id","nome", "f
    "data") VALUES({},\'{}\',\'{}\',\'{}\',NOW())'
116
    .format(rec[0],resp['nome'],resp['flags'],resp['versao']))
117                         cur.execute('UPDATE "fluxosDetectados" SET "enriqTOR" = True WHE
    {}'.format(rec[0]))
118                         con.commit()
119                     elif resp['erro']==1: #nao encontrado. Mesmo nao sendo nó TOR, atual
    informando que o enriquecimento foi feito
```

```
120                         cur.execute('UPDATE "fluxosDetectados" SET "enriqTOR" = True WHEI
{}'.format(rec[0]))
121                         con.commit()
122                     time.sleep(1)
123
124         print("Robo finalizado. Aguardando",TEMPO_RODADA,"segundos para nova rodada")
125         time.sleep(TEMPO_RODADA)
126     cur.close()
127     con.close()
128
129 #-------------------------------------------------------------------------------------
130
131 if __name__ == "__main__":
132     inserir = False
133     tipo = "ip"
134
135     if len(sys.argv) < 2:
136         print("necessario informar ip ou dominio")
137         print("python3 enriquecimento.py valor [ip|dominio]")
138         exit(-1)
139     valor = sys.argv[1]
140     if len(sys.argv) > 2:
141         if sys.argv[2] == "dominio":
142             tipo = "dominio"
143         if sys.argv[2] == "robo":
144             tipo = "robo"
145
146     if tipo == "robo":
147         iniciar_robo()
148
149     exit(0)
```

I.6 ANEXO 6 - CÓDIGO - SISTEMA DE AVISO

```
import sys
import psycopg2
import time
import datetime
from stix2.v21 import (Indicator, Malware, Relationship, Bundle)
import stixCreator
import snortCreator

TEMPO_RODADA = 600

#-------------------------------------------------------------------------------------------

def iniciar_robo():
    con = psycopg2.connect(host='localhost', database='xxx', user='xxx', password='xxx')
    cur = con.cursor()
    print("Robo iniciado:")

    while(True):

        list_ids_bloqueados = []

        #-------------------------------------------------------------
        #   Primeira regra:
        #   2 terminais com 2 fluxos não detectados
        #
        cur.execute('SELECT "id","ipDestino","fk_id" FROM "fluxosDetectados" LEFT JOIN "enriquecimentoTOR" ON "id" = "fk_id" WHERE "ipDestino" IN (SELECT "ipDestino" FROM "flu
where "bloqueado"=False and "quantidade">1 group by "ipDestino" HAVING COUNT("ipDestino"
        recset = cur.fetchall()
        for rec in recset:  #se teve resultado, tem novos fluxos para bloquear
            list_ids_bloqueados.append({'id':rec[0], 'ipDest':rec[1], 'tor':rec[2]})
        #print(list_ids_bloqueados)

        #  iniciar criação dos STIX_2.0 e SNORT
        #------------------
        #
        for ids in list_ids_bloqueados:

            indicadores = []
            indicadores_snort = []

            label = ['malicious-activity']

            if ids['tor'] != "NULL":
                label.append('anonymization')

            indicadores.append(stixCreator.add_indicator_ip(ids['ipDest'],label,"IPs uti
rootkit com ofuscação de tráfego no terminal infectado"))
            indicadores_snort.append([ids['ipDest'],ids['id'],"IP utilizado por rootkit
tráfego no terminal infectado","ip"])

            cur.execute('SELECT "nome" FROM "dominios" WHERE fk_id = {};'.format(ids['id
            recset = cur.fetchall()
            for rec in recset:
                indicadores.append(stixCreator.add_indicator_domain(rec[0],label,"Domini
rootkit com ofuscação de tráfego no terminal infectado"))
                indicadores_snort.append([rec[0],ids['id'],ids['id'],"Dominio utilizado
ofuscação de tráfego no terminal infectado","dominio"])

            resultado = stixCreator.cria_stix(indicadores)
            #grava o stix no arquivo
            with open("assets/cti-stix-visualization/stixs.txt",'a') as fw:
                fw.write("\nid{}=`{}`".format(ids['id'],resultado))
```

```
58
59              # pega e grava o snort no arquivo
60              resultado = snortCreator.cria_snortrule(indicadores_snort,"<br>")  #<br> é u
    separador de regra (padrao é '\n')
61              with open("assets/cti-stix-visualization/snorts.txt",'a') as fw:
62                  fw.write("\nid{}=`{}`".format(ids['id'],resultado))
63
64          #  atualizar informação de bloqueio no Banco de Dados
65          #------------------
66          #
67          for ids in list_ids_bloqueados:
68              cur.execute('UPDATE "fluxosDetectados" SET "bloqueado" = True WHERE id =
    {}'.format(ids['id']))
69              con.commit()
70
71          #-------------------------------------------------------------
72          #   Segunda regra:
73          #   1 terminal com 6 fluxos não detectados
74          #
75          list_ids_bloqueados = []
76
77          cur.execute('SELECT "id","ipDestino","fk_id" FROM "fluxosDetectados" LEFT JOIN
    "enriquecimentoTOR" ON "id" = "fk_id" WHERE "bloqueado" = False AND "quantidade">5;')
78          recset = cur.fetchall()
79          for rec in recset:  #se teve resultado, tem novos fluxos para bloquear
80              list_ids_bloqueados.append({'id':rec[0], 'ipDest':rec[1], 'tor':rec[2]})
81
82          for ids in list_ids_bloqueados:
83
84              indicadores = []
85              indicadores_snort = []
86              label = ['malicious-activity']
87
88              if ids['tor'] != "NULL":
89                  label.append('anonymization')
90
91              indicadores.append(stixCreator.add_indicator_ip(ids['ipDest'],label,"IPs uti
    rootkit com ofuscação de tráfego no terminal infectado"))
92              indicadores_snort.append([ids['ipDest'],ids['id'],"IP utilizado por rootkit
    tráfego no terminal infectado","ip"])
93
94              cur.execute('SELECT "nome" FROM "dominios" WHERE fk_id = {};'.format(ids['id
95              recset = cur.fetchall()
96              for rec in recset:
97                  indicadores.append(stixCreator.add_indicator_domain(rec[0],label,"Domini
    rootkit com ofuscação de tráfego no terminal infectado"))
98                  indicadores_snort.append([rec[0],ids['id'],"Dominio utilizado por rootki
    de tráfego no terminal infectado","dominio"])
99
100             #grava o stix no arquivo
101             resultado = stixCreator.cria_stix(indicadores)
102             with open("assets/cti-stix-visualization/stixs.txt",'a') as fw:
103                 fw.write("\nid{}=`{}`".format(ids['id'],resultado))
104
105
106             # pega e grava o snort no arquivo
107             resultado = snortCreator.cria_snortrule(indicadores_snort,"<br>")  #<br> é u
    separador de regra (padrao é '\n')
108             with open("assets/cti-stix-visualization/snorts.txt",'a') as fw:
109                 fw.write("\nid{}=`{}`".format(ids['id'],resultado))
110
111
112         for ids in list_ids_bloqueados:
```

```
113             cur.execute('UPDATE "fluxosDetectados" SET "bloqueado" = True WHERE id =
{}'.format(ids['id']))
114             con.commit()
115 
116         #-------------------------------------------------------------
117         #   Terceira regra:
118         #   1 terminal com 2 fluxos não detectados
119         #       - destino seja TOR;
120 
121         list_ids_bloqueados = []
122 
123         cur.execute('SELECT distinct("id"),"ipDestino" FROM "fluxosDetectados" INNER JOI
"enriquecimentoTOR" ON "id" = "fk_id" WHERE "quantidade">1 AND "bloqueado"=False;')
124         recset = cur.fetchall()
125         for rec in recset:  #se teve resultado, tem novos fluxos para bloquear
126             list_ids_bloqueados.append({'id':rec[0], 'ipDest':rec[1]})
127 
128         for ids in list_ids_bloqueados:
129 
130             indicadores = []
131             indicadores_snort = []
132             label = ['malicious-activity','anonymization']
133 
134             indicadores.append(stixCreator.add_indicator_ip(ids['ipDest'],label,"IPs uti
rootkit com ofuscação de tráfego no terminal infectado"))
135             indicadores_snort.append([ids['ipDest'],ids['id'],"IP utilizado por rootkit
tráfego no terminal infectado","ip"])
136 
137             cur.execute('SELECT "nome" FROM "dominios" WHERE fk_id = {};'.format(ids['id
138             recset = cur.fetchall()
139             for rec in recset:
140                 indicadores.append(stixCreator.add_indicator_domain(rec[0],label,"Domini
rootkit com ofuscação de tráfego no terminal infectado"))
141                 indicadores_snort.append([rec[0],ids['id'],"Dominio utilizado por rootki
de tráfego no terminal infectado","dominio"])
142 
143             resultado = stixCreator.cria_stix(indicadores)
144             #grava o stix no arquivo
145             with open("assets/cti-stix-visualization/stixs.txt",'a') as fw:
146                 fw.write("\nid{}=`{}`".format(ids['id'],resultado))
147 
148             # pega e grava o snort no arquivo
149             resultado = snortCreator.cria_snortrule(indicadores_snort,"<br>")  #<br> é u
separador de regra (padrao é '\n')
150             with open("assets/cti-stix-visualization/snorts.txt",'a') as fw:
151                 fw.write("\nid{}=`{}`".format(ids['id'],resultado))
152 
153 
154         for ids in list_ids_bloqueados:
155             cur.execute('UPDATE "fluxosDetectados" SET "bloqueado" = True WHERE id =
{}'.format(ids['id']))
156             con.commit()
157 
158         #-------------------------------------------------------------
159         #   Quarta regra:
160         #   1 terminal com 2 fluxos não detectados
161         #       - IP com 4 detecoes pelo VT
162 
163         list_ids_bloqueados = []
164 
165         cur.execute('SELECT distinct("id"),"ipDestino" FROM "fluxosDetectados" INNER JOI
"enriquecimentoVT" ON "id" = "fk_id" WHERE "quantidade">1 AND "bloqueado"=False AND "det
166         recset = cur.fetchall()
167         for rec in recset:  #se teve resultado, tem novos fluxos para bloquear
```

```
168             list_ids_bloqueados.append({'id':rec[0], 'ipDest':rec[1]})
169
170         for ids in list_ids_bloqueados:
171
172             indicadores = []
173             indicadores_snort = []
174             label = ['malicious-activity']
175
176             indicadores.append(stixCreator.add_indicator_ip(ids['ipDest'],label,"IPs uti
rootkit com ofuscação de tráfego no terminal infectado"))
177             indicadores_snort.append([ids['ipDest'],ids['id'],"IP utilizado por rootkit
tráfego no terminal infectado","ip"])
178
179             cur.execute('SELECT "nome" FROM "dominios" WHERE fk_id = {};'.format(ids['id
180             recset = cur.fetchall()
181             for rec in recset:
182                 indicadores.append(stixCreator.add_indicator_domain(rec[0],label,"Dominio
rootkit com ofuscação de tráfego no terminal infectado"))
183                 indicadores_snort.append([rec[0],ids['id'],"Dominio utilizado por rootki
de tráfego no terminal infectado","dominio"])
184
185             resultado = stixCreator.cria_stix(indicadores)
186             #grava o stix no arquivo
187             with open("assets/cti-stix-visualization/stixs.txt",'a') as fw:
188                 fw.write("\nid{}=`{}`".format(ids['id'],resultado))
189
190
191             # pega e grava o snort no arquivo
192             resultado = snortCreator.cria_snortrule(indicadores_snort,"<br>")  #<br> é u
separador de regra (padrao é '\n')
193             with open("assets/cti-stix-visualization/snorts.txt",'a') as fw:
194                 fw.write("\nid{}=`{}`".format(ids['id'],resultado))
195
196
197         for ids in list_ids_bloqueados:
198             cur.execute('UPDATE "fluxosDetectados" SET "bloqueado" = True WHERE id =
{}'.format(ids['id']))
199             con.commit()
200
201         print("Robo finalizado. Aguardando",TEMPO_RODADA,"segundos para nova rodada")
202         time.sleep(TEMPO_RODADA)
203     cur.close()
204     con.close()
205
206 #-------------------------------------------------------------------------------------
207
208 if __name__ == "__main__":
209     inserir = False
210     tipo = "ip"
211
212     if len(sys.argv) < 2:
213         print("necessario informar se deve iniciar o robô ou busca por fluxo")
214         print("python3 enriquecimento.py (fluxo|robo) [id do fluxo]")
215         exit(-1)
216
217     if len(sys.argv) > 2:
218         valor = sys.argv[1]
219
220     if sys.argv[1] == "fluxo":
221         tipo = "fluxo"
222     elif sys.argv[1] == "robo":
223         tipo = "robo"
224
```

```
225     if tipo == "robo":
226         start_time = datetime.datetime.now()
227         iniciar_robo()
228         end_time = datetime.datetime.now()
229         time_diff = (end_time - start_time)
230         execution_time = time_diff.total_seconds() * 1000
231         print("Tempo Aviso:",execution_time)
232 
233     else:
234         print("analise de fluxo - testes")
235     exit(0)
```