UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Desenvolvimento de um Sistema de Partilha de Informações Sobre Ameaças à Cibersegurança

Rafael Nuno de Freitas Pilré

Mestrado em Engenharia Informática

Trabalho de Projeto orientado por:
Professora Doutora Ana Luísa do Carmo Respício

2024

Agradecimentos

Concluir este projeto foi um verdadeiro desafio, uma jornada que começou com incertezas e obstáculos que, à primeira vista, pareciam impossíveis. Desde o início, fui confrontado com dificuldades técnicas e pessoais que puseram à prova a minha determinação. Contudo, foi através desses desafios que cresci e me superei, encontrando força onde pensava que não existia.

Em primeiro lugar, gostaria de expressar a minha profunda gratidão à minha família, que sempre esteve ao meu lado, oferecendo apoio incondicional em cada etapa deste percurso. Quero deixar uma nota especial ao meu avô, que infelizmente já não está entre nós, mas cujo espírito e ensinamentos me acompanharam e inspiraram ao longo de toda esta caminhada. Esta conquista é também dele.

Agradeço de coração à equipa da Direção de Cibersegurança (DCY), que me acolheu e proporcionou um ambiente de trabalho estimulante e colaborativo. O meu sincero agradecimento ao Eng. Ricardo Ramalho, cuja visão crítica e entusiasmo foram cruciais para o desenvolvimento deste projeto. Ao Eng. Jorge Silva, devo um reconhecimento especial pelo acompanhamento constante, pelo suporte técnico e pela disponibilidade inabalável que demonstrou em todos os momentos de necessidade. Um agradecimento também ao Miguel Saldanha e ao Tomás Ferreira, cuja ajuda foi essencial para resolver diversos problemas que surgiram ao longo do caminho. Sem o contributo de cada um de vocês, este trabalho não teria sido possível.

Não posso deixar de agradecer à minha orientadora Ana Respício, cujo apoio foi fundamental. A sua confiança em mim, a serenidade com que abordou cada desafio foram pilares essenciais na conclusão deste trabalho.

Por fim, um agradecimento sincero aos meus amigos, que estiveram presentes nos momentos mais difíceis. O vosso apoio foi crucial e, sem ele, este projeto teria sido ainda mais desafiante. A vossa amizade e incentivo permitiram-me manter o foco e continuar a acreditar que este objetivo era alcançável.

Dedico este trabalho a todos aqueles que acreditaram em mim e me apoiaram incondicionalmente ao longo deste percurso.

Resumo

Procedimentos robustos de cibersegurança são cruciais nos dias de hoje, dadas as crescentes ciberameaças e o ambiente digital em rápida evolução, especialmente para grandes corporações como a Altice Portugal. O mundo está cada vez mais dependente de tecnologias digitais, o que significa que medidas rigorosas de cibersegurança são cada vez mais necessárias. Este projeto reconhece o papel vital que a Threat Intelligence (TI) desempenha no fortalecimento das defesas contra ameaças dia-zero, ao embarcar em uma investigação crítica das preocupações de cibersegurança neste ambiente dinâmico.

Este projeto envolve o desenvolvimento de um sistema centralizado para a recolha e análise de informações sobre ciberameaças tendo como objetivo a criação de um novo sistema central de recolha de fontes de ciberameaças, incluindo ameaças dia-zero com integração no atual MISP da Altice Portugal. Durante o projeto, foram avaliadas as fontes de informação existentes e exploradas novas fontes de vulnerabilidades, *exploits* e *malware*, confrontando os dados com o conhecimento interno da empresa para avaliar a relevância dessas ameaças. As ameaças identificadas como dia-zero são armazenadas no MISP [31], permitindo uma melhor gestão e partilha de informação sobre ciberameaças. Além disso, o projeto visa obter informações que possam comprometer ativos da empresa, integrando indicadores de comprometimento (IOCs) no QRadar e no Graylog, melhorando a capacidade de monitorização e resposta rápida a ameaças.

Adicionalmente, foi realizado um cadastro detalhado de ativos ao nível das aplicações, recolhendo informações sobre as aplicações em execução nos servidores e PCs da empresa.

Com o novo sistema, detetou-se uma antecipação na obtenção de informações sobre ameaças de dia-zero, além de terem sido identificadas ameaças que, mesmo após um mês, ainda não tinham sido devidamente descritas pelas fontes de cruzamento internas.

Palavras-chave: Vulnerabilidades, Threat Intelligence, Dia-Zero, Exploits, Malware

Abstract

Robust cybersecurity procedures are crucial today, given the increasing cyber threats and the rapidly evolving digital environment, especially for large corporations like Altice Portugal. The world is becoming increasingly dependent on digital technologies, which means that stringent cybersecurity measures are more necessary than ever. This project recognizes the vital role that Threat Intelligence (TI) plays in strengthening defenses against zero-day threats, embarking on a critical investigation of cybersecurity concerns in this dynamic environment.

The goal of this project is the creation of a new central system whose goal is to collect cybersecurity threats from several sources, including 0-day-threats integrating all this information in the Altice Portugal MISP. During the project, existing information sources were evaluated and new sources of vulnerabilities, exploits, and malware were explored, comparing the data with the company's internal knowledge to assess the relevance of these threats. Threats identified as zero-day are stored in MISP, allowing for better management and sharing of information about cyber threats. Additionally, the project aims to obtain information that could compromise the company's assets, integrating indicators of compromise (IOCs) into QRadar and Graylog, thereby improving the ability to monitor and respond quickly to threats.

Furthermore, a detailed asset inventory was conducted at the application level, gathering information about applications running on the company's servers and PCs.

With the new system, there was an anticipation in obtaining information about zero-day threats, and threats were identified that, even after a month, had still not been adequately described by internal correlation sources.

Keywords: Vulnerabilities, Malware, Exploits, Threat Intelligence, Zero-Day

Conteúdo

Lista de Figuras **xiv**

Lista de Tabelas **xv**

Lista de Listagens **xvii**

Lista de Abreviaturas **xix**

1 Introdução **1**

1.1 Motivação 2
1.2 Objectivos 2
1.3 Contribuições 3
1.4 Estrutura do documento 3

2 Contexto **5**

2.1 Conceitos gerais 5
2.1.1 Vulnerabilidade 5
2.1.2 *Exploit* 6
2.1.3 *Malware* 6
2.1.4 *Threat Intelligence* 6
2.1.5 *Cyber Threat intelligence Platforms* 7
2.2 Conceitos-chave para o desenvolvimento 7
2.2.1 Vulnerabilidade dia-zero 7
2.2.2 Ciclo de vida de uma vulnerabilidade dia-Zero 8
2.2.3 MISP 9
2.2.4 *IBM Qradar* 11
2.2.5 *Graylog* 12
2.2.6 Fontes de cruzamento 13
2.2.7 Fontes de confrontação interna 13
2.3 Conceitos essenciais para recolha de dados 14
2.3.1 Recolha de dados de vulnerabilidades e *exploit*s 14
2.3.2 Abordagens de deteção e recolha de dados de *malware* 18

2.4 Outros trabalhos relacionados . . . 19
2.4.1 ZERODays . . . 20
2.4.2 ZeroDays v2 . . . 20
2.4.3 ThreatMiner . . . 21
2.4.4 Follow the Blue Bird . . . 22
2.4.5 D-Miner . . . 22

3 Recursos informativos e os seus respetivos conjuntos de dados 23
3.1 Fontes de Informação . . . 23
3.1.1 Fontes de vulnerabilidades . . . 23
3.1.2 Fontes de *exploits* . . . 24
3.1.3 Fontes de *malware* . . . 26
3.2 Fontes de cruzamento . . . 27
3.2.1 Fontes de cruzamento de vulnerabilidades e exploits . . . 27
3.2.2 Fontes de cruzamento de malware . . . 27
3.3 Fontes de confrontação interna . . . 27

4 Desenho do sistema 29
4.1 Proposta da solução . . . 29
4.1.1 Requisitos do sistema . . . 29
4.1.2 Arquitetura do sistema . . . 30
4.2 Implementação . . . 36
4.2.1 Tecnologias e ferramentas utilizadas . . . 36
4.2.2 Motor de recolha . . . 37
4.2.3 Motor de processamento . . . 46
4.2.4 Motor *Threat Intel Manager* . . . 52

5 Resultados e avaliação 55
5.1 Componente de vulnerabilidades . . . 55
5.1.1 Resultados obtidos . . . 55
5.1.2 Avaliação da componente de vulnerabilidades . . . 57
5.2 Componente de exploits . . . 58
5.2.1 Resultados obtidos . . . 58
5.2.2 Avaliação da componente de exploits . . . 59
5.3 Componente de malware . . . 61
5.3.1 Resultados obtidos . . . 61
5.3.2 Avaliação da componente de malware . . . 62

6 Conclusão 65
6.1 Trabalho futuro . . . 66

Bibliografia **70**

A Descrição das regras de expressão RegEx para extração de campos da ZDI **71**

B Descrição dos métodos para limpeza do corpo do exploit da fonte Vulners **73**

Lista de Figuras

2.1 Ciclo de vida de uma vulnerabilidade (adaptado de [29]) 8
2.2 Representação simplificada de um evento no MISP 10
2.3 Arquitetura do sistema do IBM QRadar 12
2.4 Representação das categorias e parâmetros do CVSSv3.1 16
2.5 Estrutura da versão CPE 2.3 17

4.1 Arquitetura do sistema. 31
4.2 Esquema geral do motor de recolha. 32
4.3 Esquema operacional de um módulo de recolha. 33
4.4 Esquema geral do motor de processamento. 34
4.5 Exemplo de normalização dos dados na componente de vulnerabilidades. 35
4.6 Esquema geral do motor TIM. 35
4.7 Fluxograma da recolha de informação da fonte VulDB. 38
4.8 Fluxograma da recolha de informação da fonte ZDI. 40
4.9 Diferenças entre entradas de exploits iguais (appRain CMF 4.0.5) das fontes ZDT, PSS e EDB. 41
4.10 Fluxograma da recolha de informação da fonte Vulners. 42
4.11 Fluxograma da recolha de informação da fonte INQ. 45
4.12 Cruzamento de vulnerabilidades com as FCr. 47
4.13 Evento de uma vulnerabilidade da fonte VulDB no MISP. 48
4.14 Atributos do evento da vulnerabilidade intitulada “Lepton CMS 7.0.0 Languages Place upgrade.php Local”. 49
4.15 Cruzamento de exploits com as FCr. 50
4.16 Atributos do exploit intitulado “CrushFTP Remote Code Execution”. 51
4.17 Atributos de um evento de malware. 52

5.1 Coincidência das entradas nas fontes de informação de vulnerabilidades. 55
5.2 Distribuição da quantidade de entradas recolhidas ao longo de 15 dias. 56
5.3 Quantidade de vulnerabilidades dia-zero recolhidas de cada fonte em relação ao total de ameaças recolhidas. 57
5.4 Coincidência das entradas nas fontes de informação de exploits. 58
5.5 Distribuição da quantidade de entradas recolhidas ao longo de 1 mês. 58

5.6 Quantidade de exploits dia-zero recolhido de cada fonte, em relação ao total de ameaças recolhidas. 59
5.7 Análise do desempenho do motor de processamento na identificação de exploits previamente conhecidos. 60
5.8 Coincidência das entradas nas fontes de informação de malware. 61
5.9 Quantidade de malware dia-zero recolhido de cada fonte, em relação ao total de malware recolhidas. 62

Lista de Tabelas

3.1 Fontes de informação de vulnerabilidades e os seus atributos. 24
3.2 Fontes de informação de *exploits* e os seus atributos. 25
3.3 Fontes de informação de *malware* e os seus atributos. 26

4.1 Estimativa de tempo de processamento com e sem threads 44
4.2 Intervalos do CVSS Score e correspondente nível de severidade. 48
4.3 Eficácia da distância de Levenshtein na identificação de duplicação de exploits. . 51
4.4 Comparação de nomes de software entre SCCM e CrowdStrike. 53
4.5 Tabela de IOCs e tipos de fontes internas. 54

5.1 Quantidade de IOCs Recolhidos 62

Lista de Listagens

B.1 Função para normalização de dados de fonte 73

Lista de Abreviaturas

CVE *Common Vulnerabilities and Exposures*

CVSS *Common Vulnerability Scoring System*

EDB Exploit-DB

FCi Fontes de Confrontação interna

FCr Fontes de Cruzamento

INQ Inquest Labs

IOCs Indicadores de Compromisso

MAB Malware Bazaar.ch

MISP *Malware Information Sharing Platform*

PSS Packet Storm Security

SIEM Security Information and Event Management

TI *Threat Intelligence*

ZDI Zero Day Initiative

ZDT 0Day.Today

Capítulo 1

Introdução

A cibersegurança, na sua essência, constitui um pilar crítico para a operacionalidade e sustentabilidade das empresas no cenário digital contemporâneo. À medida que a tecnologia avança, as organizações enfrentam desafios cada vez mais complexos relacionados à proteção dos seus sistemas de informação. A Altice desempenha um papel central na sociedade, conectando milhões de pessoas e empresas através de infraestruturas tecnológicas críticas que sustentam a vida moderna. Nesse contexto, a cibersegurança surge como um elemento estratégico indispensável para proteger esses sistemas e garantir a continuidade dos seus serviços perante as crescentes ameaças digitais. Estes sistemas, intrinsecamente ligados ao sucesso empresarial, são constantemente visados por agentes mal-intencionados que procuram explorar vulnerabilidades para fins nefastos. A integridade, confidencialidade e disponibilidade desses sistemas não são apenas componentes essenciais da infraestrutura de TI, mas também pilares fundamentais da cibersegurança que asseguram a continuidade dos negócios perante ameaças digitais.

A relevância da cibersegurança é amplificada pela evolução do panorama tecnológico, onde a adoção de novas ferramentas e a interconexão crescente entre sistemas expandem a superfície de ataque das organizações [23]. Este cenário aumenta a exposição a riscos de segurança, exigindo uma atenção meticulosa e estratégias proativas para identificar, prevenir e mitigar potenciais ameaças. A proteção eficaz contra tais ameaças não apenas salvaguarda informações valiosas contra o acesso não autorizado, mas também protege a reputação da empresa e previne perdas financeiras significativas resultantes da interrupção de operações críticas.

Num ambiente empresarial cada vez mais digitalizado e interconectado, a cibersegurança transcende a sua função tradicional de proteção de dados. Ela torna-se um componente estratégico crucial para a inovação, a competitividade e a resiliência organizacional. Perante este contexto, é imperativo que as empresas adotem uma visão holística e integrada da cibersegurança, que englobe não apenas a defesa contra ataques cibernéticos, mas também a capacidade de antecipar, responder e recuperar de incidentes de segurança de maneira eficiente e eficaz.

1.1 Motivação

A transformação digital acelerada, impulsionada pela inovação tecnológica e pela necessidade de adaptação a um mercado globalizado, trouxe consigo um aumento exponencial na quantidade e na diversidade de dispositivos conectados às redes corporativas. Esta evolução não só potencializou a eficiência e a capacidade de inovação das empresas, mas também expandiu significativamente a superfície de ataque, introduzindo vulnerabilidades acrescidas e ampliando o espectro de ciberameaças [38]. A interconexão crescente entre sistemas, dispositivos e plataformas diversificadas transformou a segurança da informação num desafio dinâmico e multifacetado, onde a deteção e a prevenção de ameaças requerem uma vigilância constante e estratégias adaptativas.

A urgência de uma abordagem proativa à cibersegurança foi dramaticamente ressaltada pela pandemia de COVID-19. A crise sanitária global forçou organizações de todos os setores a adotarem o trabalho remoto como uma medida de continuidade operacional, expondo as redes corporativas a um volume sem precedentes de conexões remotas. Este cenário acelerou a digitalização dos processos de trabalho, mas também exacerbou os riscos de segurança, pois a expansão do perímetro de segurança para além das fronteiras físicas tradicionais das empresas facilitou o acesso de atores mal-intencionados a dados sensíveis e infraestruturas críticas. A necessidade de proteger uma força de trabalho dispersa e de garantir a segurança de dados em trânsito tornou-se uma prioridade indiscutível.

Neste contexto, a *Threat Intelligence* (TI) [37] surge como um componente estratégico vital para a cibersegurança. Ao fornecer informações aprofundadas sobre o panorama de ameaças, a TI capacita as organizações a compreenderem não apenas as ameaças iminentes, mas também a anteciparem-se a ataques futuros. A integração de informações detalhadas sobre táticas, técnicas e procedimentos (TTPs) empregados por adversários permite às equipas de segurança aprimorar as suas estratégias defensivas, identificando e mitigando proativamente potenciais ameaças antes que sejam exploradas [36]. A eficácia dessa abordagem é ilustrada pelos projetos ZeroDays [48] e ZeroDays v2 [18] da Altice, que aplicaram esses conceitos na prática.

1.2 Objectivos

O objetivo principal deste projeto é estabelecer um sistema eficaz de partilha de informação sobre ameaças dia-zero dentro da Altice Portugal. Para isso, o trabalho tem os seguintes subobjetivos:

- Criar a plataforma dia-zero da Altice Portugal;
- Apreciar as atuais fontes, ao mesmo tempo que se procura ativamente novas fontes para enriquecer o conhecimento;
- Elaborar processos de forma a identificar novas ameaças que sejam dia-zero para a empresa;
- Desenvolver mecanismos para centralizar a informação relativa a ameaças dia-zero na plataforma MISP;

- Criar processos para identificar o software em execução nos ativos da Altice Portugal.
- Desenvolver processos que ajudem a identificar os ativos afetados pelas ameaças dia-zero;
- Exportar os IOCs do MISP no QRadar SIEM para melhorar a deteção e resposta a ameaças.

Desta forma, este trabalho propõe-se a contribuir para os grandes objetivos estratégicos da Altice:

- Reforçar a postura de segurança da Altice Portugal face às ciberameaças;
- Estabelecer uma prática robusta de partilha de inteligência de ameaças;
- Melhorar a capacidade de resposta da Altice Portugal perante ameaças em constante evolução.

1.3 Contribuições

Este projeto foi criado com o objetivo de melhorar a deteção precoce de ciberameaças, focando-se especialmente em vulnerabilidades, *exploits* e *malware*. As principais realizações deste projeto incluem:

- Criação da atual plataforma dia-zero da Altice;
- Automação da recolha de ameaças a partir de fontes de informação selecionadas;
- Identificação de informações relevantes nos dados recolhidos;
- Processamento da informação, incluindo operações de remoção de duplicados e classificação, e correlação com os cadastros de ativos da empresa, determinando a possível afetação da infraestrutura.
- Armazenamento das ameaças dia-zero no MISP;
- Recolha de informação sobre o software aplicacional dos ativos da empresa, com o objetivo de identificar possíveis ativos comprometidos por ameaças de dia-zero;
- Ingestão dos IOCs no QRadar SIEM para aumentar a riqueza e precisão da análise de segurança.

Em resumo, este projeto visa recolher informações relevantes sobre ameaças de dia-zero à Direção de cibersegurança, permitindo uma ação rápida e reduzindo o tempo entre a publicação dessas ameaças na web e a sua deteção pela equipe do CyberSOC da organização.

1.4 Estrutura do documento

O relatório está organizado em vários capítulos, cada um dos quais serve um propósito específico e contribui para uma compreensão holística do mesmo:

- Capítulo 2 – É dedicado a estabelecer as bases conceituais necessárias para uma melhor compreensão do projeto. Aqui, serão aprofundados os conceitos e teorias essenciais que suportam esta pesquisa, bem como ferramentas e trabalhos já desenvolvidos neste tema;
- Capítulo 3 – Serão apresentadas as características que as fontes de informação devem possuir, as fontes de informação escolhidas para a implementação do projeto, e as fontes utilizadas para cruzamento e confrontação;
- Capítulo 4 - No capítulo 4, será explicada a arquitetura do sistema, bem como serão descritos os aspetos fundamentais da sua implementação;
- Capítulo 5 - No capítulo 5, serão avaliados e analisados os resultados da execução das três componentes do sistema, a componente de vulnerabilidades, *exploits* e *malware*;
- Capítulo 6 - No capítulo 6, será apresentada uma síntese do trabalho realizado, com reflexões sobre os resultados alcançados e sugestões para possíveis melhorias futuras visando aperfeiçoar o sistema.

Capítulo 2

Contexto

Este capítulo apresenta toda a contextualização do presente trabalho, definindo a base para o desenvolvimento do sistema.

2.1 Conceitos gerais

Esta secção estabelece os fundamentos teóricos essenciais para o desenvolvimento e compreensão do projeto em questão. Nesta Secção, serão referidos os conceitos fundamentais que estabelecem a base para a discussão subsequente. É crucial compreender esses elementos, pois eles fornecem o contexto necessário para a abordagem detalhada dos conceitos-chave no desenvolvimento do projeto e na análise de ciberameaças.

2.1.1 Vulnerabilidade

A vulnerabilidade, no contexto da cibersegurança, é uma questão crucial amplamente reconhecida e definida por diversas fontes. De acordo com o *National Institute of Standards and Technology* (NIST) [35], ela é descrita como ”uma fragilidade num sistema de informação, ou nos seus procedimentos de segurança, controlos internos ou implementação, que pode ser potenciada ou despoletada por alguma fonte de ameaça”. Pode manifestar-se como uma fraqueza no hardware ou software do sistema, nas políticas e procedimentos utilizados no sistema e até mesmo nos próprios utilizadores do sistema. As vulnerabilidades foram identificadas devido à compatibilidade e interoperabilidade do hardware e também ao esforço necessário para as corrigir. As vulnerabilidades de software podem ser encontradas em sistemas operativos, software de aplicação e software de controlo, como protocolos de comunicação e controladores de dispositivos. Existem vários fatores que levam a falhas no design de software, incluindo fatores humanos e complexidade do software. As vulnerabilidades técnicas geralmente ocorrem devido a falhas humanas [7].

Nenhum sistema está automaticamente imune a ciberameaças, e as consequências de ignorar os riscos devido à complacência, negligência e incompetência são evidentes. Em 2015, um número sem precedentes de vulnerabilidades foram identificadas como *exploits* de dia-zero que foram utilizados como armas, e os kits de exploração de ataques web estão a adaptar-se e a evoluí-los

mais rapidamente do que nunca. À medida que mais dispositivos estão ligados, as vulnerabilidades serão exploradas [44].

2.1.2 *Exploit*

Um *exploit* é um pedaço de software, dados ou uma sequência de comandos que aproveita uma vulnerabilidade para causar comportamentos não intencionados ou para obter acesso não autorizado a dados sensíveis [47]. Em termos mais simples, podemos comparar um *exploit* a uma chave mestra que é usada para abrir uma porta que deveria estar trancada. A vulnerabilidade representa a fechadura, e o *exploit* é a chave que permite o acesso não autorizado a informações ou sistemas que, de outra forma, estariam protegidos.

No mundo da cibersegurança, os *exploits* são frequentemente usados por atacantes para explorar fraquezas em sistemas computacionais, aplicativos ou redes. Quando uma vulnerabilidade é identificada por atacantes, os *exploits* podem ser criados para tirar proveito dela. Isso pode resultar em ações prejudiciais, como a invasão de sistemas, a divulgação de informações confidenciais ou a interrupção das operações de uma organização.

Por outro lado, os *exploit*s também são usados de forma legítima por profissionais de cibersegurança e investigadores para testar a segurança de sistemas e aplicativos. Eles ajudam a identificar e corrigir vulnerabilidades antes que sejam exploradas por indivíduos mal-intencionados.

2.1.3 *Malware*

O termo ”*Malware*”resulta da combinação das palavras “malicioso” e “software” e é utilizado para designar qualquer software indesejado. O termo foi definido de forma geral por G. McGraw e G. Morrisett como ”qualquer código adicionado, alterado ou removido de um sistema de software com o intuito de causar intencionalmente danos ou subverter a função prevista do sistema”[28].

Malware inclui uma variedade de ciberameaças, como vírus, *trojan*, *worms* e *spyware*, que são desenvolvidos para realizar ações prejudiciais nos sistemas informáticos ou dispositivos dos utilizadores. Estas ações podem incluir o roubo de informações confidenciais, a perturbação do funcionamento normal do sistema ou até mesmo o sequestro de dados, através de técnicas como o *ransomware*.

2.1.4 *Threat Intelligence*

Entende-se por TI a provisão de conhecimento fundamentado em evidências sobre ameaças existentes ou potenciais, e a determinação da eficácia das capacidades de controlo [24]. A TI é uma componente crítica das estratégias modernas na cibersegurança, fornecendo às organizações o conhecimento e os *insights* essenciais necessários para proteger eficazmente os seus ativos digitais. TI gira em torno da recolha, análise e disseminação de informações relacionadas com ciberameaças existentes ou potenciais. O seu principal objetivo é apoiar a tomada de decisões informadas pelas análises de segurança cibernética, melhorar as medidas de proteção e proteger contra várias formas de ciberataques. A TI opera com base na premissa de recolher conhecimento

baseado em evidências de fontes de informação, que se dividem em estruturadas e não estruturadas. As fontes estruturadas são organizadas de acordo com um formato predefinido, geralmente seguindo um modelo ou uma linguagem específica, enquanto as fontes não estruturadas não têm uma organização predefinida ou um formato consistente. Estas fontes podem incluir dados como indicadores de ameaça, padrões de ataque, vulnerabilidades e até mesmo comportamento de agentes de ameaça. A informação recolhida é tipicamente derivada de dados de *open-source*, disponíveis publicamente, e de fontes privadas e proprietárias.

2.1.5 *Cyber Threat intelligence Platforms*

Muitas empresas começaram a depender de *Threat Intelligence Platforms* (TIPs) para superar as limitações e falhas dos sistemas de deteção e monitorização existentes, especialmente o Security Information and Event Management (SIEM) [46]. Estas TIPs são responsáveis por extrair dados estruturados e não estruturados de diversas fontes externas e realizar várias operações complexas, como filtragem, agregação, normalização, deteção, análise e enriquecimento, além de inserir os resultados no SIEM. Contudo, a implementação e utilização destas plataformas ainda estão numa fase inicial e existem várias desvantagens que precisam de ser abordadas, como a avaliação dinâmica da confiança das fontes externas e capacidades avançadas de análise, nas quais ainda é necessária intervenção manual, especialmente para tornar a informação recolhida efetivamente aplicável [41]. Algumas organizações adotaram TIPs *open-source*, sendo as mais utilizadas: MISP (*Malware Information Sharing Platform*) [31] [1], CIF (*Collective Intelligence Framework*) [12], CRITs (*Collaborative Research Into Threats*) [33] e SoltraEdge [27]. O *Malware Information Sharing Platform* (MISP) é o que foi utilizado no projeto (Secção 2.2.3).

2.2 Conceitos-chave para o desenvolvimento

Nesta secção, exploramos conceitos cruciais que desempenham um papel fundamental no desenvolvimento deste projeto. A compreensão aprofundada desses elementos é essencial para uma análise abrangente e uma implementação eficaz.

2.2.1 Vulnerabilidade dia-zero

A definição de uma vulnerabilidade dia-zero refere-se a uma falha de segurança informática que é desconhecida ou não divulgada pelo fabricante ou programador do software ou sistema afetado. No contexto de empresa, uma vulnerabilidade dia-zero refere-se a uma ameaça desconhecida pela organização, ainda sem deteção ou mitigação implementada. O termo dia-zero significa que a vulnerabilidade é descoberta e explorada por atacantes antes que os responsáveis pela segurança tenham tido a oportunidade de corrigi-la. Isso pode ocorrer no mesmo dia em que a vulnerabilidade é identificada, daí o termo [6].

[1] O **MISP** é uma solução de software de código aberto para recolher, armazenar, distribuir e partilhar indicadores de cibersegurança e ameaças relacionadas com incidentes de cibersegurança e análise de malware.

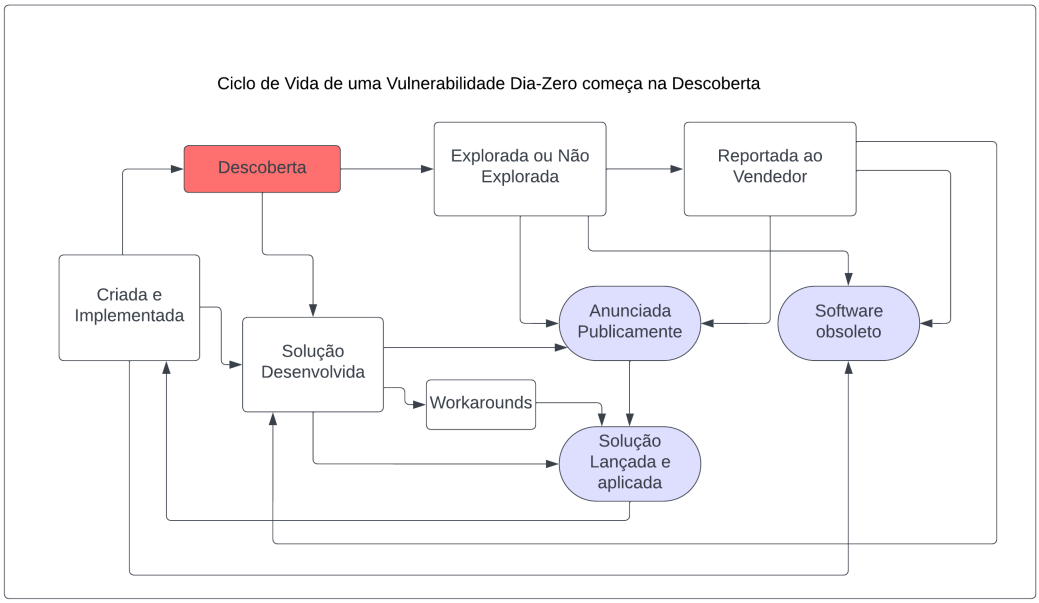


Figura 2.1: Ciclo de vida de uma vulnerabilidade (adaptado de [29])

Essas vulnerabilidades são particularmente perigosas, pois os programadores não têm conhecimento prévio delas e, portanto, não tiveram a oportunidade de lançar um *patch* [2] para proteger os sistemas afetados. Os atacantes podem explorar essas falhas para realizar ataques a sistemas e redes, muitas vezes causando danos significativos.

2.2.2 Ciclo de vida de uma vulnerabilidade dia-Zero

O ciclo de vida de uma vulnerabilidade[29] é ilustrado na Figura 2.1. A caixa vermelha representa o início da vida de uma vulnerabilidade dia-zero, enquanto as caixas roxas representam eventos que encerram essa vida. Uma vulnerabilidade dia-zero existe apenas durante a parte do ciclo de vida que começa com a sua descoberta e termina com a sua correção, descontinuação do software ou divulgação pública. É importante observar que uma vulnerabilidade pode permanecer em várias fases do ciclo de vida ao mesmo tempo, por tempo indeterminado, sem necessariamente mudar de estado.

Uma vulnerabilidade surge quando um programador incorpora no software funções inseguras e falha na validação de dados de entrada. Após essa implementação, a vulnerabilidade pode desaparecer inadvertidamente antes de ser detetada, seja devido a uma correção acidental, ou permanecer oculta durante toda a vida do software. Alternativamente, pode ser descoberta, tornando-se assim uma vulnerabilidade dia-zero.

Depois que a vulnerabilidade é incorporada no software e este é disponibilizado ao público, pode ser identificada por um utilizador comum ou um investigador. Uma vez descoberta, essa

[2] Um **patch** é uma correção de software liberada pelo programador para reparar a vulnerabilidade não conhecida anteriormente ou sem solução até aquele momento.

vulnerabilidade, agora considerada como dia-zero, pode ou não ser explorada. Em alguns casos, o fornecedor pode identificá-la e corrigi-la antes que alguém possa explorá-la novamente.

Além das possibilidades mencionadas anteriormente, é importante notar que, em alguns cenários, pode não existir um *patch* imediato para corrigir uma vulnerabilidade dia-zero. No entanto, durante esse período de vulnerabilidade, os especialistas em segurança podem identificar e recomendar *workarounds* – soluções temporárias que reduzem a exposição ao risco sem corrigir completamente a falha subjacente. Esses *workarounds* podem incluir configurações específicas, restrições de acesso ou outras medidas que dificultem a exploração da vulnerabilidade.

Após a descoberta, uma vulnerabilidade pode ser explorada, dependendo de quem a descobriu e para que fins, podendo optar por mantê-la em segredo para explorar a vulnerabilidade ou vendê-la a terceiros para exploração. A vulnerabilidade também pode ser explorada por vários grupos, permanecendo como uma vulnerabilidade dia-zero até ser relatada ao fornecedor e corrigida, o software tornar-se obsoleto ou se for anunciada publicamente sem notificação ao fornecedor.

Quando uma vulnerabilidade é descoberta, ela torna-se uma vulnerabilidade dia-zero. A divulgação responsável envolve a notificação direta ao fornecedor para que tenha a oportunidade de desenvolver uma correção ou solução antes de ser anunciada publicamente. Após receber a notificação, o fornecedor pode anunciar publicamente a vulnerabilidade antes de desenvolver uma solução, desenvolver e lançar uma correção ou manter a vulnerabilidade em segredo e deixar que a obsolescência resolva o problema.

A divulgação responsável permite ao fornecedor ter uma correção pronta para lançar publicamente. Caso contrário, o fornecedor é notificado quando a vulnerabilidade é divulgada publicamente, o que possibilita a exploração pública enquanto a correção está em desenvolvimento e implementação.

Se a correção lançada e aplicada abordar uma vulnerabilidade dia-zero, a vida dessa termina. Acontece por vezes, uma correção ser lançada e implementada que resolve uma vulnerabilidade que ainda não foi anunciada. Infelizmente, uma nova vulnerabilidade pode ser criada durante o processo de correção.

É possível que uma vulnerabilidade dia-zero nunca seja relatada e, portanto, a sua vida termina quando o software que a possui não está mais em uso.

2.2.3 MISP

O MISP, originalmente criado por militares para compartilhar informações sobre *malware*, evoluiu para um projeto *open-source* que permite a colaboração entre organizações para reunir, compartilhar e correlacionar diversos tipos de ameaças. Essas ameaças incluem Indicadores de Compromisso (IOCs) [30], que são sinais ou artefactos indicativos de uma intrusão cibernética, informações sobre fraudes financeiras e informações de contraterrorismo [51][31]. O MISP é uma plataforma de *Open Source Intelligence* (OSINT) amplamente utilizada por milhares de organizações, incluindo agências como a Organização do Tratado do Atlântico Norte (NATO), ministérios da defesa, comunidades de *Computer Security Incident Response Teams* (CSIRT) e

empresas privadas [31].

No momento, o MISP possui, principalmente, as seguintes funcionalidades: partilha, armazenamento, correlação automática de IoCs, recursos avançados de filtragem e capacidade de exportar e importar dados nos formatos mais comuns, como STIX, OpenIOC, CSV e o formato padronizado do MISP [43][51].

O modelo de dados descreve o formato padrão de descrição para a criação de eventos no MISP. A principal motivação era ter um formato simples e conveniente, que ao mesmo tempo permitisse requisitos mais complexos. Uma vantagem desta abordagem simples é que o utilizador pode decidir por si mesmo o nível de granularidade da informação que deseja partilhar. Por exemplo, um utilizador pode descrever um evento com múltiplos atributos, fornecendo o máximo de informação possível, ou pode apenas colocar o mínimo de informação para um evento. O principal objetivo é contar com um formato de dados viável mínimo e expandi-lo conforme a necessidade de complexidade adicional surgir, em vez de tentar capturar todos os possíveis requisitos futuros antecipadamente. Uma nova entrada no MISP é chamada de objeto de evento. Um evento pode ser definido como um conjunto de características e todos os tipos de descrições para um IoC, incluindo anexos, entre outros. Essas características e informações relevantes são chamadas atributos. Atributos de evento, por exemplo, são data do IoC, nível de ameaça, comentários, organização.

Os atributos podem ser divididos em duas classes distintas, principalmente: categoria e tipo. A principal diferença é que um atributo de categoria contém informações mais gerais, como dados de alvo, atividade de rede, tipo de fraude, entre outros. Num atributo de tipo inclui informações como checksums (md5, sha1), nome do arquivo, nome do host, endereço IP, fonte e destino de email e afins. Além disso, um evento também pode ter *tags*. Uma representação simplificada deste modelo de dados é dada na Figura 2.2.

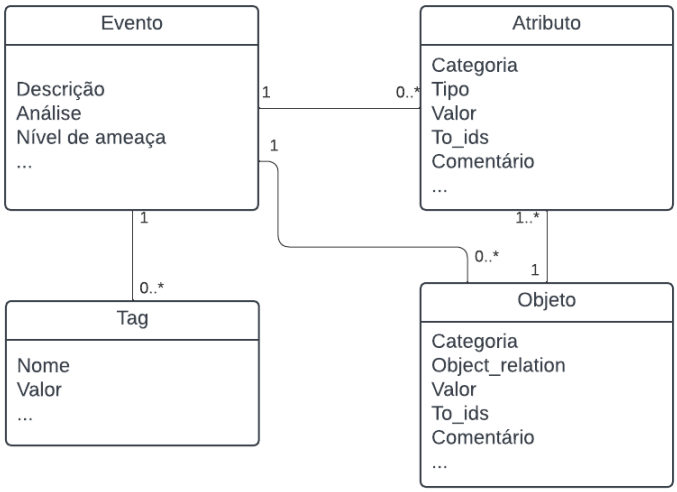


Figura 2.2: Representação simplificada de um evento no MISP

Além dos atributos, os objetos de evento são elementos fundamentais no modelo de dados

do MISP. Cada objeto é uma estrutura que agrupa atributos específicos, representando diferentes entidades ou componentes associados a um evento. Por exemplo, um objeto pode representar um endereço IP, um domínio ou um ficheiro. Essa organização dos dados torna o processo de análise e identificação de padrões muito mais eficiente, facilitando a compreensão dos eventos.

Os objetos fornecem uma estrutura para organizar e descrever de forma abrangente todos os atributos relacionados a um determinado elemento dentro de um evento. Através deles, é possível agregar múltiplos tipos de atributos, tais como endereços IP, nome de um ficheiro, entre outros.

Para melhor se adaptar aos diferentes contextos organizacionais e aos diversos tipos de eventos, o MISP permite ainda a criação de *templates* (modelos) personalizados para os objetos. Estes templates fornecem um padrão reutilizável que os utilizadores podem definir para capturar atributos específicos que são relevantes ao tipo de ameaça ou informação que estão a partilhar. A criação de templates personalizados permite que cada organização crie objetos que satisfaçam as suas necessidades específicas, assegurando que a informação é recolhida e partilhada de forma eficiente e significativa.

2.2.4 *IBM Qradar*

O *IBM QRadar* é um *SIEM (Security Information and Event Management)*, ou seja, é uma plataforma de gestão de segurança de rede que proporciona monitorização e análise de eventos, bem como registo de dados de segurança para efeitos de conformidade, proteção e deteção de incidentes. O seu propósito é auxiliar na gestão de segurança através da recolha, processamento e normalização dos dados provenientes de várias fontes externas, como servidores *Windows*, servidores *Unix*, *firewalls* e servidores *proxy*. Isto é demonstrado na Figura 2.3, onde o *IBM QRadar* é capaz de receber dados destas fontes e passá-los através da sua estrutura de recolha e processamento.

A arquitetura é composta pela consola do *QRadar*, que os analistas utilizam para visualizar, pesquisar e investigar eventos, bem como elaborar relatórios. Os eventos e fluxos recolhidos passam pelos coletores (*Event Collector* e *Flow Collector*) e depois são processados (*Event Processor* e *Flow Processor*) para normalização, análise e geração de ofensas.

É importante explicar o conceito de evento no contexto do *QRadar*, onde um evento representa o registo de uma ação. Uma ofensa alerta para atividade suspeita ou incidente, caso seja detetada atividade suspeita por um ou mais eventos/fluxos. Este processo de determinação do que constitui uma ofensa baseia-se na capacidade do *QRadar* de executar as diferentes regras para testar cada evento e fluxo. Se um evento corresponder às condições de uma regra, cria-se uma ofensa e associa-lhe o evento a ela. É possível verificar a descrição da ofensa, os eventos que a compõem, a origem do registo, a fonte, o alvo, o momento e a forma como ocorreu, bem como as regras que desencadearam os eventos para formar ofensas, e o motivo, através de notas. No caso de uma regra ser ativada, podem ser definidas várias ações a serem executadas, como determinar o seu nível de gravidade, credibilidade e relevância, que influenciarão o cálculo da sua importância, decidir se fará parte de uma ofensa ou não, sendo registado ou descartado, e até mesmo quais

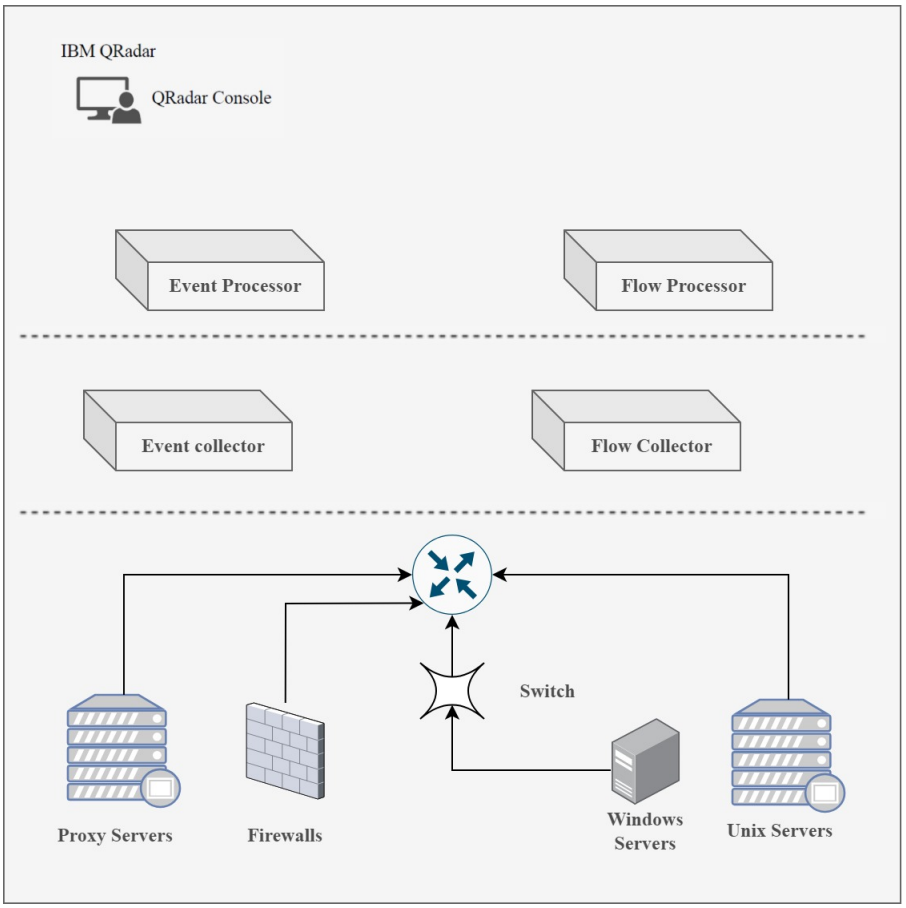


Figura 2.3: Arquitetura do sistema do IBM QRadar

emails e notificações devem ser acionados.

O *IBM QRadar* opera também com base em blocos de construção essenciais, incluindo os *building blocks* e os *reference sets*. O conceito de *building blocks* ajuda na composição de regras, sendo estes uma coleção de testes que não resultam numa resposta ou ação e são usados para fornecer lógica complexa que poderá ser reutilizada na construção de regras. Por sua vez, os *reference sets* são conjuntos de dados que ajudam a enriquecer a análise de eventos, fornecendo informações de contexto relevantes, como listas de endereços IP maliciosos, URLs suspeitos, entre outros.

2.2.5 *Graylog*

O *Graylog* é uma plataforma *open-source* dedicada à gestão de *logs*, permitindo a recolha, indexação e análise de grandes volumes de dados de registo em tempo real. A sua arquitetura escalável e modular torna-o numa solução adequada para empresas de diferentes dimensões e ambientes de TI variados.

O *Graylog* destaca-se pela sua capacidade de centralizar e simplificar a gestão de *logs* provenientes de múltiplas fontes. As suas principais funcionalidades incluem:

- **Recolha de Logs:** O *Graylog* suporta a recolha de *logs* a partir de uma vasta gama de fontes, tais como sistemas operativos, aplicações, dispositivos de rede, entre outros. Utiliza protocolos padrão, como *Syslog* e *GELF* (*Graylog Extended Log Format*), além de suportar outros métodos de integração;

- **Indexação e Armazenamento:** Os *logs* recolhidos são indexados e armazenados em bases de dados como *Elasticsearch*. Esta abordagem permite consultas rápidas e eficientes, facilitando a recuperação de informações relevantes em tempo útil;

- **Análise e Visualização:** O *Graylog* fornece ferramentas avançadas para a análise de *logs*, incluindo um motor de busca robusto, *dashboards* personalizáveis e alertas em tempo real. Estas funcionalidades são cruciais para a deteção de padrões anómalos, monitorização contínua e resposta rápida a incidentes de segurança ou operacionais;

- **Alertas e Notificações:** É possível configurar alertas baseados em condições específicas dos *logs*. Quando um alerta é acionado, o *Graylog* pode enviar notificações através de vários canais, como e-mail, *Slack* ou outras ferramentas de comunicação, assegurando que as equipas relevantes são informadas prontamente.

2.2.6 Fontes de cruzamento

As Fontes de Cruzamento (FCr) nomeadas pelo projeto ZeroDays v2 [18], desempenham um papel interno essencial e serão utilizadas neste projeto com o intuito de avaliar se os dados recolhidos são já do conhecimento da organização. A existência destas fontes de informação é de importância crítica, pois permite-nos determinar quais ameaças são consideradas de dia-zero, isto é, ainda desconhecidas e sem soluções disponíveis, e quais ameaças já são previamente compreendidas e possivelmente geridas.

As FCr podem assumir várias formas, incluindo bases de dados internas, registos históricos, monitorização contínua de sistemas e redes, entre outros. A sua função primordial é fornecer um ponto de referência para avaliar a novidade das ameaças identificadas. Quando se deteta uma nova ameaça, a comparação com as informações armazenadas nas FCr permite determinar se a mesma já foi registada e estudada anteriormente pela organização, ou se é, de facto, uma vulnerabilidade de dia-zero, que exige uma resposta imediata.

2.2.7 Fontes de confrontação interna

Neste projeto, foram recolhidos dados dos ativos da empresa, conhecidos como Fontes de Confrontação interna (FCi) no contexto do projeto ZeroDays v2. Essas fontes de informação contêm dados detalhados sobre o software aplicacional instalado nos sistemas utilizados pela organização, o que é também essencial para avaliar se a empresa está vulnerável a uma ameaça específica.

A qualidade da futura confrontação entre essas ameaças e os ativos internos dependerá da extensão e precisão dos dados disponíveis nos registos internos. A falta de informações suficientes ou precisas pode resultar na geração de alertas redundantes e não fundamentados.

2.3 Conceitos essenciais para recolha de dados

Nesta secção, aborda-se conceitos fundamentais que são imperativos para a recolha eficaz de dados, uma componente vital deste projeto. Explora-se a importância das bases de dados de vulnerabilidades, *exploit*s e *malware*, destacando como estes recursos são fundamentais para compreender e enfrentar desafios relacionados à cibersegurança. Além disso, examina-se técnicas de deteção de *malware*, essenciais para identificar e responder a potenciais ameaças. O entendimento profundo desses conceitos é essencial para a construção de uma estratégia sólida de recolha de dados no âmbito deste trabalho.

2.3.1 Recolha de dados de vulnerabilidades e *exploit*s

A *MITRE Corporation* [33] mantém a lista *Common Vulnerabilities and Exposures* (CVE) [34], uma compilação de vulnerabilidades conhecidas descritas num formato padrão. Um índice global de vulnerabilidades conhecidas simplifica análises complexas, como a deteção de ameaças persistentes avançadas. Portanto, indexar vulnerabilidades conhecidas no CVE tornou-se uma prática padrão para todos os tipos de profissionais de segurança, incluindo fornecedores de software. Cada entrada no CVE possui um ID (CVE-ID), uma breve descrição e a data de criação.

A *National Vulnerability Database* (NVD) do NIST [35] espelha e complementa as entradas do CVE na sua base de dados. A cada hora, a NVD entra em contato com o CVE para obter vulnerabilidades recém-divulgadas. Cada vulnerabilidade indexada na NVD passa por uma análise detalhada, incluindo a atribuição de uma pontuação de impacto com base no *Common Vulnerability Scoring System* - CVSS - (para as versões 2.0 [20] e 3.0 [19]), e links relacionados à vulnerabilidade, como sites de avisos ou discussões técnicas. A NVD usa o CVE-ID em vez de um identificador próprio.

Adicionalmente, muitas outras bases de dados online compilam vulnerabilidades conhecidas e fornecem o uso irrestrito dos seus conteúdos. A informação complementar fornecida por cada base de dados pode variar, mas, em geral, elas oferecem uma descrição, alguma análise das questões de segurança levantadas pelas vulnerabilidades, *exploit*s conhecidos e possíveis correções ou ações de mitigação. Essa diversidade de fontes enriquece a visão global e a compreensão das ameaças de segurança, permitindo que os profissionais se beneficiem de uma variedade de perspetivas e informações.

Além disso, no contexto da cibersegurança, é crucial considerar informações sobre *exploit*s associadas a vulnerabilidades específicas. Bases de dados, como a Exploit-DB (EDB) [17] e a Packet Storm Security (PSS) [42], compilam dados sobre *exploit*s, proporcionando uma compreensão mais aprofundada das técnicas de exploração associadas às vulnerabilidades conhecidas.

Essas fontes adicionais permitem aos profissionais identificar vulnerabilidades e, além disso, compreender como elas podem ser exploradas, auxiliando na implementação de medidas eficazes de mitigação e correção. A utilização de identificadores únicos da ameaça, seja um hash ou qualquer outro tipo de código exclusivo, é de extrema importância nesse cenário. Esses identificadores possibilitam uma correspondência precisa e a recuperação do código do *exploit* associado à vulnerabilidade específica. Disponibilizar esse código não apenas permite uma análise mais aprofundada da ameaça, mas também facilita a adoção de contramedidas específicas, contribuindo para uma resposta mais eficaz diante das ciberameaças. Outros elementos frequentemente incluídos na descrição destas ameaças são igualmente relevantes, como o título da informação, o autor, a explicação detalhada e a data de publicação. Esses campos desempenham um papel crucial ao proporcionar contexto sobre a ameaça ao utilizador final do sistema.

Common Vulnerability Scoring System

O *Common Vulnerability Scoring System* (CVSS) [19] é, nos dias de hoje, o standard para a definição e cálculo da severidade de vulnerabilidades. Este é composto por três grupos de métricas: Base, Temporal e Ambiente, cada um consistindo num conjunto de métricas, como mostrado na Figura 2.4.

Estes grupos de métricas são descritos da seguinte forma:

- **Base**: representa as características intrínsecas e fundamentais de uma vulnerabilidade que são constantes ao longo do tempo e em ambientes de utilizadores;
- **Temporal**: representa as características de uma vulnerabilidade que mudam ao longo do tempo mas não entre ambientes de utilizadores;
- **Ambiente**: representa as características de uma vulnerabilidade que são relevantes e únicas para o ambiente particular de um utilizador.

As métricas de base, imutáveis, avaliam as características intrínsecas da vulnerabilidade, incluindo a facilidade de exploração e o impacto potencial sobre a confidencialidade, integridade e disponibilidade do sistema. Estas dividem-se em métricas de explorabilidade como vetor de ataque, complexidade do ataque, privilégios necessários, interação da vítima e âmbito da vulnerabilidade e métricas de impacto.

As métricas temporais levam em conta a evolução da vulnerabilidade, incluindo o desenvolvimento de *exploits* e o estado de remediação, bem como a confiabilidade das informações sobre a vulnerabilidade. As métricas ambientais personalizam a avaliação ao contexto específico do utilizador, ajustando as métricas de base e incluindo requisitos de impacto conforme a importância da confidencialidade, integridade e disponibilidade para a empresa.

A combinação destas métricas fornece uma pontuação de severidade de 0 a 10, permitindo uma avaliação padronizada de severidade associado a cada vulnerabilidade. Este sistema pro-

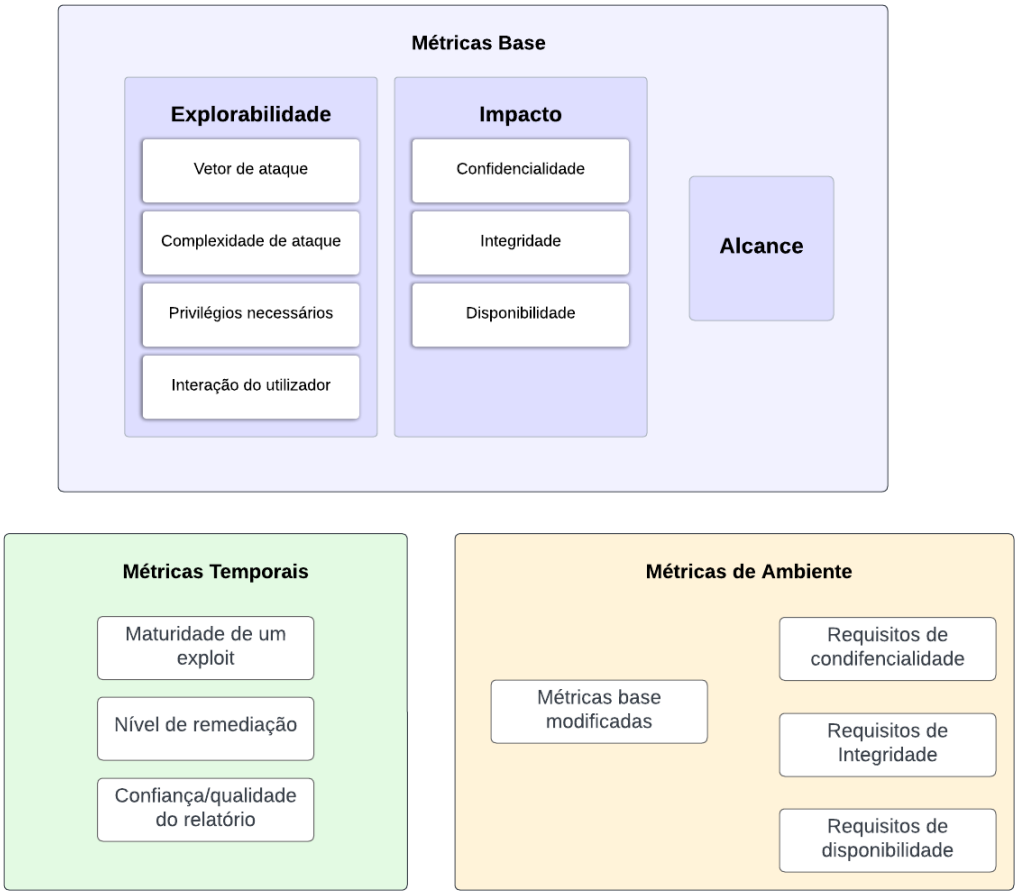


Figura 2.4: Representação das categorias e parâmetros do CVSSv3.1

cura minimizar subavaliações de severidade, fornecendo uma ferramenta útil para gerir e mitigar vulnerabilidades de forma eficaz.

Common Platform Enumeration

O *Common Platform Enumeration* (CPE) [32] é uma outra ferramenta essencial nesse ecossistema. O CPE fornece uma linguagem padronizada para descrever plataformas de software e hardware, permitindo uma identificação consistente das configurações do sistema afetadas por uma vulnerabilidade. O CPE ajuda a melhorar a precisão e a eficácia das análises de segurança, tornando mais fácil para os profissionais correlacionarem informações sobre vulnerabilidades específicas com as plataformas tecnológicas envolvidas.

Originalmente criado pela MITRE e agora sob a gestão do NIST, a versão atual do CPE é a 2.3, que foi introduzida em 2011. Esta versão segue a arquitetura em camadas representado na Figura 2.5. A fundamentação desta arquitetura começa com a camada de nomenclatura, que organiza a estrutura conceitual para a criação de identificadores nomeados de forma adequada (WFN, *Well-*

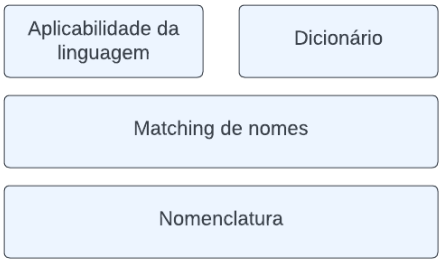


Figura 2.5: Estrutura da versão CPE 2.3

Formed Names), facilitando a vinculação desses identificadores a URIs e strings estruturadas. Tal organização viabiliza a designação de WFNs para categorias de produtos dentro do espectro tecnológico. Cada WFN é formado por conjuntos de atributos e valores seguindo a notação:

WFN: [part=”a”, vendor=”microsoft”, product=”internet-explorer”, version=”8.0.6001”, update=”beta”]

Os atributos fundamentais incluem, mas não se limitam a:

1. **Part**: Identifica a categoria do software, classificando-se em aplicações (a), sistemas operacionais (o) e dispositivos de hardware (h);

2. **Vendor**: Denomina a entidade responsável pela produção do produto;

3. **Product**: Refere-se ao nome ou título do produto;

4. **Version**: Designa o número da versão do produto atribuído pelo produtor, sendo este alfanumérico.

Acima da camada de nomenclatura, existe a metodologia para comparação e associação entre diferentes CPEs, conhecida como *matching* de nomes. Este processo permite a análise e confirmação se dois WFNs de origens distintas pertencem à mesma categoria de produtos, se um constitui um subconjunto do outro ou se são completamente distintos.

No topo da estrutura, encontram-se o dicionário de nomes e a linguagem de aplicabilidade. O dicionário é definido como a norma para construção de repositórios que armazenam WFNs e seus metadados correspondentes. Estes são armazenados em formato de URI ou strings formatadas, com o formato:

URI: cpe:/o:microsoft:windows_vista:6.0:sp2: - professional - x86 -

Expressão formatada: cpe:2.3:o:microsoft:windows_vista:6.0:sp2:-:-:professional:-:x86:-

O NIST mantém o dicionário oficial de CPEs, chamado Official CPE Dictionary, centralizando a pesquisa de identificadores de produtos e eliminando discrepâncias entre diferentes bases de dados. Entidades independentes podem também gerir seus próprios dicionários e submeter versões consolidadas para inclusão no dicionário oficial.

A linguagem de aplicabilidade é utilizada para definir configurações de sistemas ou plataformas mediante a associação lógica de vários identificadores. O atributo *part*, por exemplo, especifica o tipo de produto, facilitando a criação de expressões de aplicabilidade que agrupam WFNs em URIs ou strings formatadas para configurar sistemas por categoria de produto, não por configurações específicas.

Para a divulgação de vulnerabilidades, a correlação de informações de CPE com novas vulnerabilidades permite identificar ativos e tecnologias impactados. Assim, ao aproveitar as funcionalidades de cada camada, facilita-se o *matching* de CPEs ou atributos associados a ativos e sistemas, bem como suas configurações, vinculando automaticamente vulnerabilidades a ativos afetados. Isso contribui significativamente para a gestão de segurança automatizada.

2.3.2 Abordagens de deteção e recolha de dados de *malware*

Os métodos de deteção de *malware* desempenham um papel central na cibersegurança, sendo cruciais para identificar e conter ameaças. Essa importância é destacada pelo papel fundamental desempenhado por plataformas, que, ao empregar esses métodos, expõem publicamente essas amostras, contribuindo significativamente para a comunidade de segurança digital.

A. Métodos de deteção de malware:

As técnicas de deteção de *malware* são utilizadas para identificar, detetar e prevenir que o sistema informático seja infetado, protegendo-o contra perdas de informação. Podem ser categorizadas em deteção baseada em assinaturas, deteção baseada em heurística e *sandbox*.

1. Deteção baseada em assinaturas:

   A deteção baseada em assinaturas utiliza uma assinatura ou sequência de bits para identificar o *malware* e o seu tipo. Programas com um código único são executados quando um ficheiro é aberto ou iniciado no computador. O scanner de *malware* recolhe o código numa base de dados na *cloud*. Esta base de dados contém uma vasta gama de códigos de vírus. Se o código do ficheiro corresponder a uma assinatura na base de dados, ele é identificado como malicioso e rejeitado pelo programa *antimalware* no computador. Após o programa antivírus desmontar o ficheiro infetado e reconhecer o seu padrão e sequência para determinar o seu tipo, o ficheiro é apagado. Esta técnica pode ser estática, dinâmica ou híbrida [45].

2. Deteção baseada em heurística:

   A deteção baseada em heurística é uma abordagem para detetar e distinguir entre o comportamento normal e anormal do sistema para identificar ataques de *malware* conhecidos e

desconhecidos e encontrar uma solução adequada. Regras ou sistemas baseados em pesos determinam o quão arriscada pode ser uma função de um programa. Se essas regras excederem o limite predefinido, são tomadas ações preventivas com base nas configurações do sistema.

3. Deteção de SandBox:

   A *SandBox* [9] é uma célula protegida dentro de um computador criada por programas anti*malware* para conter programas ou código não testados ou não tratados. Impede a infeção por *malware*, pois o programa funciona sem infetar para prejudicar o dispositivo hospedeiro. Dentro da *SandBox*, o ficheiro é observado e analisado para determinar se é prejudicial ou seguro e são realizados testes em programas não verificados que podem conter um vírus ou outro código malicioso, caso o ficheiro seja legítimo, será libertado; se for prejudicial, será rejeitado.

B. Recolha de dados de malware:

Plataformas como a any.run [40] e hybrid analysis [11] oferecem ambientes *SandBox* avançados, complementando as abordagens tradicionais. Essas *SandBoxes* permitem a execução segura de arquivos suspeitos, conectando-se de maneira sinérgica com bases de dados para melhorar a deteção e análise. Além das técnicas de deteção, dados importantes para o tratamento da informação de software malicioso são os seguintes:

- Indicadores de Compromisso (IOC): Informações que podem indicar intrusões, como alterações em arquivos ou conexões a endereços IP específicos;
- Síntese de Arquivo: Um número fixo gerado criptograficamente, útil para identificar variações em arquivos;
- Plataforma Afetada: Na maioria das vezes dispõe apenas do sistema operativo;
- MIME e Tipo de Arquivo: O MIME ajuda a identificar o tipo de arquivo, auxiliando na deteção de software malicioso;
- Outros dados relevantes: estirpe, família do *malware*, identificadores únicos, a data da primeira vez que foi observado.

2.4 Outros trabalhos relacionados

Nesta secção pretende-se abordar projetos relacionados ao projeto em questão com foco nos mecanismos de recolha de ameaças e analisar os mesmos. Estes projetos diferem todos no elemento essencial deste projeto que o faz destacar, que é a partilha de ciberameaças, a sua integração com sistemas de gestão de vulnerabilidades e a sua correlação com os sistemas internos da empresa.

2.4.1 ZERODays

O ZERODays [37], desenvolvido em 2021 na Altice Portugal, destaca-se como uma ferramenta inovadora que automatiza o processo de recolha de informações sobre vulnerabilidades, permitindo posterior integração e análise interna. As fontes de informação utilizadas pelo autor para este projeto foram a VulDB [49] e a Zero Day Initiative (ZDI) [22], e como fonte de atualização foi utilizada a *National Vulnerability Database* (NVD) [35]. Este sistema é composto por três módulos distintos, denominados motores: o motor de informação, o motor de processamento e o motor de notificação.

O motor de informação engloba vários sub-módulos dedicados à recolha sistemática de dados sobre novas vulnerabilidades a cada intervalo de 8 horas. Uma hora após cada operação de recolha, procede-se a uma verificação detalhada para identificar novos dados e, assim, atualizar e enriquecer a informação previamente obtida, através da fonte de atualização.

O motor de processamento é responsável por normalizar as informações recolhidas, tornando-as compatíveis para cruzamento com os dados armazenados no Elasticsearch, provenientes de iterações anteriores, e com a base de dados da Qualys Cloud Platform. As informações resultantes deste processo são inseridas no Elasticsearch, sendo direcionadas para um índice específico caso sejam pertinentes para os ativos da empresa.

Por último, o motor de notificação assume a responsabilidade de alertar os diversos departamentos internos da empresa acerca das vulnerabilidades identificadas.

Os resultados finais do projeto revelam que o ZERODays alcançou com sucesso os seus objetivos de recolha proativa de informações sobre vulnerabilidades. Ao realizar operações em três intervalos diários, o sistema conseguiu reunir e armazenar um número significativo de vulnerabilidades, a maioria das quais era considerada "*0-day*"para a empresa. Contudo, a análise também salientou desafios, como a presença de falsos positivos e a limitação na correspondência de vulnerabilidades com ativos devido a informações restritas nos cadastros. Apesar destes desafios, a implementação de notificações proativas mostrou ser uma ferramenta eficaz para informar sobre novas vulnerabilidades, mesmo que a integração direta com as equipas de ativos afetados seja complexa devido à falta de informações detalhadas.

2.4.2 ZeroDays v2

O projeto ZeroDays v2 [18] na Altice Portugal, realizado em 2022, surge como uma evolução do ZERODays, com o objetivo de melhorar a deteção, análise e notificação de ameaças, como *exploit*s e *malware*. Na definição do processo de monitorização de *exploit*s e *malware*, o autor propôs e implementou uma solução abrangente para a recolha, análise, monitorização e notificação de ameaças, fornecendo informações cruciais à Direção de Cibersegurança sobre potenciais riscos emergentes. O autor baseou-se em fontes confiáveis de *exploit*s, como 0day.today [5], EDB e o PSS. Para a obtenção de dados relacionados a *malware*, foram consultadas fontes incluindo Malware Bazaar.ch (MAB) [8] e a a Inquest Labs (INQ) [25]. Os requisitos fundamentais do ZeroDays v2 foram meticulosamente delineados para garantir uma implementação robusta e efi-

caz. Destacou-se a importância da modularidade, preservando e ampliando os componentes de recolha de informação. A simplicidade foi uma diretriz para evitar erros de implementação e facilitar a manutenção. A confiança nas fontes de informação foi essencial, assim como a produção de dados detalhados e conclusivos. A rapidez e certeza no processo de notificação foram cruciais para garantir a relevância e a eficácia das informações relatadas. Na componente de *exploit*s para lidar com formatações diferentes nas fontes, foi implementado um sistema de comparação de *exploit*s. Após verificar a existência de nova informação, os dados dos *exploit*s são interpretados, uniformizados e cruzados com ferramentas internas para identificar ameaças dia-zero. A confrontação com o cadastro interno permitiu categorizar e determinar sistemas vulneráveis. A triagem e interpretação da informação de software malicioso seguiram a mesma abordagem adotada para *exploit*s, mas considerando apenas as entradas presentes em ambas as fontes para maior fiabilidade. Os resultados obtidos pelo sistema foram altamente positivos, com a componente de *exploit*s reportando sete ameaças dia-zero relevantes para a organização no primeiro mês de atividade. Embora a componente de software malicioso tenha sido analisada por um período mais curto, revelou ameaças já conhecidas internamente. No entanto, esse conhecimento interno permitiu aumentar a visibilidade ao CyberSOC sobre estas ameaças possibilitando uma resposta mais ágil diante de possíveis ataques.

2.4.3 ThreatMiner

O projeto Threat Miner [15] é um motor inovador de análise de texto que utiliza dados da dark web para identificar potenciais ciberameaças. Desenvolvido por uma equipa de investigadores na Universidade da Cidade de Birmingham, o Threat Miner emprega uma combinação de técnicas avançadas de *machine learning* e um repositório de ameaças personalizado para extrair informações acionáveis de fóruns da *dark web*. Essas informações extraídas são então formatadas de uma maneira que pode ser integrada de forma transparente nas TIPs.

A emergência da dark web proporcionou uma plataforma clandestina para atores maliciosos compartilharem *exploit*s, violações e vazamentos de dados, representando um desafio significativo para profissionais de cibersegurança. O Threat Miner enfrenta esse desafio monitorizando proativamente fóruns da *dark web* e extraindo informações relevantes que podem ser usadas para identificar e mitigar potenciais ameaças.

A eficácia do Threat Miner advém da sua capacidade de analisar dados de texto não estruturados, uma tarefa muitas vezes difícil para algoritmos tradicionais de *machine learning* lidarem. O motor utiliza várias técnicas, incluindo processamento de linguagem natural (PLN) e análise sentimental, para extrair informações significativas de publicações na *dark web*.

Um componente-chave do Threat Miner é o seu repositório de ameaças personalizado, que é continuamente atualizado com novas informações sobre ameaças emergentes. Este repositório permite que o motor identifique e classifique com precisão ameaças potenciais, mesmo aquelas que são novas ou obscuras.

As informações extraídas pelo Threat Miner são então formatadas de maneira a poderem ser

facilmente integradas nas plataformas existentes de inteligência de ameaças. Isso permite que analistas de segurança incorporem de forma transparente a inteligência da *dark web* nos seus fluxos de trabalho existentes, proporcionando uma visão mais abrangente do panorama de ameaças.

2.4.4 Follow the Blue Bird

O objetivo do projeto Follow the Blue Bird [10] foi estudar a quantidade e a qualidade dos dados de ameaças publicados no Twitter. Os investigadores recolheram dados de ameaças do Twitter por meio de uma ferramenta de web scraping. A ferramenta recolheu tweets que continham palavras-chave relacionadas a ameaças, como "*malware*", "*ransomware*"e "*phishing*".

Ao analisar os dados, os investigadores identificaram a quantidade de dados de ameaças disponíveis no Twitter, constatando que mais de 20 milhões de tweets relacionados a ameaças foram recolhidos em um período de seis meses. Eles também avaliaram a qualidade desses dados, observando que esta variava, com alguns tweets oferecendo informações úteis sobre ameaças, enquanto outros continham dados imprecisos ou irrelevantes. Além disso, os investigadores identificaram diversos tipos de ameaças nos dados obtidos, sendo as mais comuns *malware*, *ransomware* e *phishing*. Concluíram que o Twitter é uma fonte valiosa de dados de ameaças, capaz de monitorizar tendências, identificar novas ameaças e contribuir para o desenvolvimento de ferramentas de segurança mais eficazes.

Os autores do projeto destacaram a utilidade dos dados do Twitter para melhorar a cibersegurança, identificar novos ataques e tendências de ameaças, além de enfatizar a importância do desenvolvimento de ferramentas analíticas para compreender melhor as ciberameaças. Também ressaltam a necessidade de consciencialização sobre os perigos das ciberameaças e esperam que o projeto Follow the blue bird contribua significativamente para melhorar a cibersegurança.

2.4.5 D-Miner

O D-Miner [26] é uma *framework* projetada para a recolha de dados em mercados de *exploits* dia-zero na *dark web*. Os criadores do D-Miner propõem que a recolha automatizada dessas informações pode contribuir significativamente para a compreensão e análise por parte das entidades interessadas. Para alcançar este objetivo, desenvolveram uma metodologia específica e sugerem uma possível implementação para lidar com o problema identificado.

Experiências realizadas demonstram que o D-Miner é capaz de reunir informações de forma automatizada a partir de diversas fontes, incluindo dois mercados de ameaças na *dark web*. Isso evidencia a eficácia do sistema na captura de dados relevantes.

No entanto, apesar da sua versatilidade, o D-Miner apresenta uma limitação importante: não realiza a integração ou o cruzamento de informações com dados internos da empresa. Como resultado, a informação recolhida não pode ser avaliada ou qualificada com base na sua relevância específica para o contexto da organização.

Capítulo 3

Recursos informativos e os seus respetivos conjuntos de dados

A qualidade do sistema irá refletir-se na qualidade das suas fontes de informação. Deste modo, neste capítulo serão apresentados os dados que as fontes de informação devem conter sobre as ameaças.

3.1 Fontes de Informação

Para a seleção das fontes de informação aplicou-se vários métodos para identificar quais fontes seriam consideradas para o projeto. O primeiro estudo realizado constou na obtenção dos campos de cada uma das fontes para verificar se estas tinham a informação necessária para identificar se um ativo da empresa poderia ficar comprometido. Após esse primeiro estudo, para cada uma das fontes de informação (vulnerabilidades, *exploit*s ou *malware*) fez-se uma recolha manual de amostras de ameaças para entender se a fonte seria importante ou não.

3.1.1 Fontes de vulnerabilidades

As fontes analisadas incluíram a *CVEdetails* [16], *Cybersecurity-help* [14] (ou CSH), *Rapid7* [39], *Open Source Vulnerabilities* (ou OSV) [21], Zero Day Initiative (ZDI), VulDB e NVD.

A Tabela 3.1 apresenta as fontes de informação sobre vulnerabilidades consideradas e os dados que estas podem fornecer. Entre estes dados, destacam-se, como discutido na Secção 2.3.1, o ID do CVE, o CPE para identificar a plataforma afetada e o nível de severidade, conforme o CVSS. É observável que fontes como *Rapid7* e *CVEdetails* não oferecem informações sobre o produto afetado.

No segundo estudo dessas fontes, é crucial compreender que identificar uma fonte como dia-zero para a empresa significa que a vulnerabilidade é nova, ou seja, a vulnerabilidade já é de conhecimento público, mas não está presente nas fontes de cruzamento da empresa ou é uma vulnerabilidade sem CVE-ID atribuído. VulDB, CVEdetails, CSH e Rapid7 são fontes comercializadas. Dado que este projeto não justifica o investimento em mais que uma fonte, foi analisado quais destas seria a fonte mais abrangente em termos de informação sobre novas vulnerabilidades

	CVEdetails	CSH	Rapid7	OSV	ZDI	VULDB	NVD
Identificador da vulnerabilidade				✓	✓	✓	
Descrição	✓	✓	✓	✓	✓	✓	✓
Métrica de severidade	✓	✓	✓		✓	✓	✓
Contramedidas	✓	✓	✓	✓		✓	✓
CVE	✓	✓	✓	✓	✓	✓	✓
CPE ou Plataforma afetada	✓	✓		✓	✓	✓	✓
CVSS	✓	✓	✓		✓	✓	✓
API	✓	✓	✓	✓		✓	✓

Tabela 3.1: Fontes de informação de vulnerabilidades e os seus atributos.

ou vulnerabilidades dia-zero.

Foram escolhidas 30 vulnerabilidades para comparar essas fontes em termos de informação e a data em que foram reportadas. A frequência de publicação na divulgação de uma vulnerabilidade por parte de uma fonte é crucial para a empresa, especialmente no caso de vulnerabilidades dia-zero. Após a conclusão deste estudo, verificou-se que a CVEdetails utiliza exclusivamente informações provenientes da API da NVD, levando à sua exclusão do projeto. A Rapid7 e a CSH, embora em alguns casos reportassem vulnerabilidades mais cedo que outras fontes, por vezes não tinham conhecimento de ameaças. A VulDB continha informações sobre todas as ameaças recolhidas e, em alguns casos, apresentava informações mais cedo. Portanto, decidiu-se considerar a VulDB como a fonte de vulnerabilidades mais abrangente, excluindo a Rapid7 e a CSH, apesar da sua relevância.

Quanto às fontes como a ZDI e a OSV, que se focam mais em reportar vulnerabilidades dia-zero de diferentes vendedores de software, verificou-se se as ameaças que continham eram também publicadas na VulDB. A conclusão da análise foi que a VulDB não possuía registos da maioria dessas ameaças. Por outro lado, embora a OSV disponha de alguma informação, ela é limitada e a sua API ainda está em desenvolvimento, não sendo considerada. Além disso, a estrutura de informação da fonte é invariável, o que não atende às necessidades específicas de extração de informação relevante para o projeto. A NVD, sendo um repositório oficial de vulnerabilidades mantido pelo NIST, é utilizada como repositório de informações sobre tecnologias empregadas na empresa para o conhecimento interno de ameaças. No entanto, não faz sentido utilizá-la como fonte de informações sobre novas ameaças, uma vez que não publica essas informações de forma antecipada. Concluindo, decidiu-se adotar a VulDB e a ZDI como as fontes principais para o projeto, dada a sua abrangência e foco em vulnerabilidades dia-zero.

3.1.2 Fontes de *exploits*

As fontes de *exploits* analisadas incluíram a EDB, PSS, Vulners [50], CXSecurity (ou CXS) [13] e 0Day.Today (ZDT) [1].

A Tabela 3.2 apresenta as fontes de informação relativas a *exploit*s e os dados importantes que devem ser recolhidos. Dos dados gerais, os mais importantes são a plataforma afetada pelo *exploit*, para que, caso não exista um ID de CVE para a vulnerabilidade explorada, seja possível fazer uma

correspondência interna. Esse cruzamento interno pode ser auxiliado também com informações que possam estar presentes no título ou descrição do *exploit*.

O ID do CVE da vulnerabilidade explorada é, destes três, a informação mais importante de obter, pois com este conseguimos obter informação sobre o seu CVSS, um padrão utilizado para avaliar o nível de severidade das vulnerabilidades. Em alternativa, em casos em que a fonte de informação não consegue disponibilizar o ID do CVE da vulnerabilidade explorada, poderá ser útil uma possível métrica de severidade própria, ou seja, a severidade que este *exploit* apresenta, de acordo com a fonte.

	EDB	PSS	Vulners	CXS	ZDT
Identificador único	✓	✓	✓	✓	✓
Título	✓	✓	✓	✓	✓
Autor	✓	✓	✓	✓	✓
Descrição		✓	✓	✓	✓
Data de publicação	✓	✓	✓	✓	✓
Plataforma afetada	✓	✓		✓	✓
CVE	✓	✓	✓	✓	✓
Métrica de severidade	✓				
Código disponível	✓	✓	✓	✓	✓
API			✓		

Tabela 3.2: Fontes de informação de *exploits* e os seus atributos.

No segundo estudo desta fonte de informação, foram recolhidas amostras de ameaças diferentes a partir dos seus ID de CVE. Concluiu-se que a fonte CXS publica, na sua maioria, a mesma informação que a fonte PSS, no entanto, a PSS costuma disponibilizá-la mais cedo. É importante salientar que, embora não seja possível confirmar que a PSS é uma fonte de informação utilizada pela CXS, chegou-se a essa conclusão com base na observação temporal das publicações e portanto esta fonte deixou de ser considerada.

A fonte Vulners foi a única a ser utilizada no projeto, por disponibilizar uma API ao contrário das outras fontes e por normalizar a estrutura das informações provenientes de diversas fontes como PSS, ZDT e EDB. Através da Vulners, é possível aceder a uma compilação de dados que, de outra forma, estariam dispersos e, por vezes, desconhecidos pelas próprias fontes originais. A EDB, apesar de ser contemplada para uso na fase de implementação, apresenta certas limitações, tais como entradas sem uma descrição formatada num campo específico, com a informação a ser incluída diretamente no corpo do *exploit*. A fonte PSS, que também apresenta características distintivas, não fornece explicitamente informações sobre a plataforma afetada, o que requer que tal dado seja inferido a partir do título do *exploit*. Além disso, a PSS fornece um resumo que facilita a atribuição de um identificador único ao ficheiro, publica apenas informações verificadas e disponibiliza um feed RSS que simplifica o processo de recolha de dados.

3.1.3 Fontes de *malware*

Para as fontes de *malware*, já referenciadas anteriormente, as fontes analisadas incluíram Any.Run, INQ, MAB e Hybrid Analysis (ou HAN).

Na Tabela 3.3, são apresentadas quatro fontes de dados e as suas contribuições. Identificar a plataforma afetada é crucial para análises internas (FCi), enquanto a data do primeiro avistamento complementa informações sobre a maturidade do software malicioso e as defesas já existentes. A verificação das fontes de *malware* é imperativa para garantir a confiabilidade das informações. A disponibilidade de IOCs é essencial para a eficácia do SIEM, fornecendo pistas essenciais para a deteção e resposta a ameaças. Detalhes como o tipo de *malware* e sua família são fundamentais para compreender o comportamento malicioso, especialmente em casos como o de *Spyware*, que monitoriza as ações da vítima. O segundo levantamento realizado nestas fontes envolveu a

	Any.run	INQ	MAB	HAN
Plataforma afetada				✓
Tamanho	✓	✓	✓	✓
Síntese	✓	✓	✓	✓
Ficheiro disponível			✓	
Tipo do ficheiro	✓	✓	✓	✓
Data de primeiro avistamento		✓	✓	✓
IOC	✓	✓	✓	✓
Tipo de sofware malicioso	✓	✓	✓	✓
Família do software malicioso	✓	✓	✓	✓
Verifica informação	✓	✓	✓	✓
API	✓	✓	✓	

Tabela 3.3: Fontes de informação de *malware* e os seus atributos.

obtenção dos hashes de diversas amostras maliciosas, seguido pela verificação da sua presença no VirusTotal, uma das principais FCr da empresa. Para cada fonte, foram recolhidas 20 amostras classificadas como maliciosas. Observou-se que nas fontes INQ e MAB, entre duas a três amostras maliciosas não eram reconhecidas pelo VirusTotal, ou, se o eram, nenhum antivírus as identificava como maliciosas. Essa constatação levou à inclusão destas duas fontes no âmbito do projeto.

As fontes Any.Run e HAN apresentaram resultados particularmente interessantes, revelando que, das 20 amostras analisadas, aproximadamente metade das amostras maliciosas recolhidas não eram conhecidas pelo VirusTotal. Ou, caso fossem, não tinham sido detetadas como maliciosas por nenhum antivírus. No entanto, dada a necessidade de otimização de recursos e considerando que o projeto não justifica a utilização de mais do que um orçamento para uma fonte, a Any.Run foi excluída da seleção, embora seja reconhecida a sua relevância. As fontes MAB e INQ, por sua vez, foram integradas no projeto devido aos resultados promissores apresentados.

3.2 Fontes de cruzamento

Conforme mencionado na Secção 2.2.6, o funcionamento do projeto requer a definição e a interação com fontes de cruzamento. Esta etapa é essencial para avaliar o conhecimento interno da empresa acerca de determinadas ameaças. Sem este processo, torna-se impossível determinar se uma ameaça específica é conhecida internamente.

3.2.1 Fontes de cruzamento de vulnerabilidades e exploits

Para avaliar a cobertura interna de vulnerabilidades e *exploits* será utilizada como fonte de cruzamento interna à organização a base de dados do produto Qualys VMDR 2.0. Cada vulnerabilidade identificada é comparada com registos mantidos pelas ferramentas de segurança da empresa. Estes registos incluem vulnerabilidades catalogadas nas bases de dados dessas ferramentas, que podem ser detetadas durante suas operações. A comparação do identificador CVE com registos equivalentes nos sistemas internos permite determinar se a empresa já possui proteção contra a vulnerabilidade em questão. Aquelas para as quais não se encontra correspondência são tratadas como vulnerabilidades de dia zero dentro do contexto empresarial, indicando a ausência de proteção interna.

Na componente de *exploits* além da verificação do CVE, é crucial realizar uma análise mais profunda, que inclui a verificação das métricas de CVSS, a indicação da existência de um *exploit* e a correspondência deste *exploit* com os registos no EDB, entre outros critérios. Esta análise detalhada assegura uma compreensão mais completa da exposição à severidade e da eficácia das medidas de proteção adotadas pela empresa.

3.2.2 Fontes de cruzamento de malware

Tal como é crucial avaliar o conhecimento interno sobre vulnerabilidades e *exploits* para se determinar ameaças dia-zero, é igualmente importante aplicar um processo semelhante ao avaliar *malware*. Contudo, devido à falta de informações específicas sobre o CVE-ID das vulnerabilidades que o *malware* visa explorar nas fontes atuais, não é viável correlacionar diretamente estas informações com a base de dados do Qualys VMDR 2.0. No entanto, a obtenção do hash dos ficheiros de *malware* possibilita o uso da IBM X-Force Exchange e do VirusTotal para determinar a novidade da ameaça.

3.3 Fontes de confrontação interna

A identificação precisa e a gestão de vulnerabilidades, *exploits* e *malware* são fundamentais para a proteção dos ativos informáticos de uma organização. Neste projeto, enfrentou-se o desafio de obter uma visão completa e atualizada do estado dos ativos informáticos, uma vez que o cadastro dos ativos na empresa é gerido através de diversas fontes. Este sistema permite ter uma base de dados consolidada e atualizada dos sistemas, cuja informação é regularmente sincronizada com

um índice do *Elasticsearch*. Este índice, contudo, limita-se a informação sobre os sistemas, sem detalhar as aplicações que estão instaladas em cada um deles.

A ausência de detalhes sobre as aplicações instaladas nos sistemas representa uma lacuna significativa, pois a presença destes e as suas versões podem determinar o nível de exposição a certas ameaças. Para colmatar esta falha, foi desenvolvido um processo que não estava planeado para este projeto, mas que era essencial para a identificação dos ativos afetados por estas ameaças.

Capítulo 4

Desenho do sistema

Após a definição do contexto no Capítulo 2, essencial para entender as diversas tecnologias e conceitos, e a investigação realizada no Capítulo 3, onde as fontes de informação selecionadas foram apreciadas e avaliadas, delineou-se o sistema atual. Este sistema abrange a recolha, análise, processamento e identificação de ativos comprometidos, em resposta ao problema identificado, tendo em conta as limitações, possibilidades e o limite temporal para realização do mesmo.

4.1 Proposta da solução

Nas Secções subsequentes, é apresentada a proposta para o desenho do sistema. Esta solução está condicionada pelos requisitos estabelecidos na Subsecção 4.1.1, os quais delineiam as características essenciais que o projeto deve atender. Em seguida, é descrita a arquitetura dos motores do sistema e como eles se interconectam.

4.1.1 Requisitos do sistema

1. **Modularidade**

    - Garantir e aumentar a modularidade dos componentes de recolha de informação, permitindo a inclusão ou exclusão de fontes sem impactar significativamente o funcionamento do sistema;
    - Manter e melhorar a simplicidade e facilidade de manutenção dos módulos, assegurando a sua independência funcional.

2. **Simplicidade**

    - Priorizar abordagens de design e desenvolvimento simples para minimizar erros de implementação e facilitar a manutenção contínua do sistema.

3. **Confiabilidade**

    - Selecionar fontes de informação altamente confiáveis para garantir a relevância e precisão dos dados recolhidos;

- Reduzir a incidência de falsos positivos, aumentando a confiabilidade do sistema em detectar e tratar ameaças.

4. **Conclusividade**

   - Produzir dados detalhados e conclusivos que permitam decisões e ações eficazes baseadas nas informações processadas.

5. **Rapidez**

   - Os dados devem ser recolhidos com a maior rapidez aquando da sua existência na fonte de forma a que a empresa possa agir sobre a mesma de atempadamente.

4.1.2 Arquitetura do sistema

A arquitetura representada na Figura 4.1 é construída em torno de três componentes principais, cada uma focada numa área específica de ameaças, mantendo um funcionamento lógico uniforme em todas elas:

- **Componente de vulnerabilidades** - Esta componente concentra-se na recolha, processamento e gestão de vulnerabilidades.

- **Componente de exploits** - Esta componente concentra-se na recolha, processamento e gestão de exploits.;

- **Componente de malware** - Esta componente concentra-se na recolha, processamento e gestão de malware.

Cada um desses componentes é integral para a arquitetura, operando independentemente em várias etapas, desde a recolha de dados das fontes pertinentes até o processamento e análise, garantindo uma abordagem segmentada e eficaz para lidar com as ameaças.

No âmbito do desenvolvimento do sistema, a arquitetura, conforme ilustrada na Figura 4.1, foi concebida como uma estrutura composta por duas fases distintas, que se desdobram desde o pré-processamento até ao pós-processamento do MISP de forma cíclica. O sistema é composto por:

- **Motor de recolha** - O motor de recolha interage com fontes de informação para extrair os seus dados, composto por módulos individuais para cada fonte. Falhas num módulo não devem afetar outros, garantindo a continuidade da recolha de informações;

- **Motor de processamento** - O motor de processamento gerencia a informação, incluindo normalização e integração com fontes internas de cruzamento e dados do MISP. Isso visa identificar ameaças desconhecidas para a empresa;

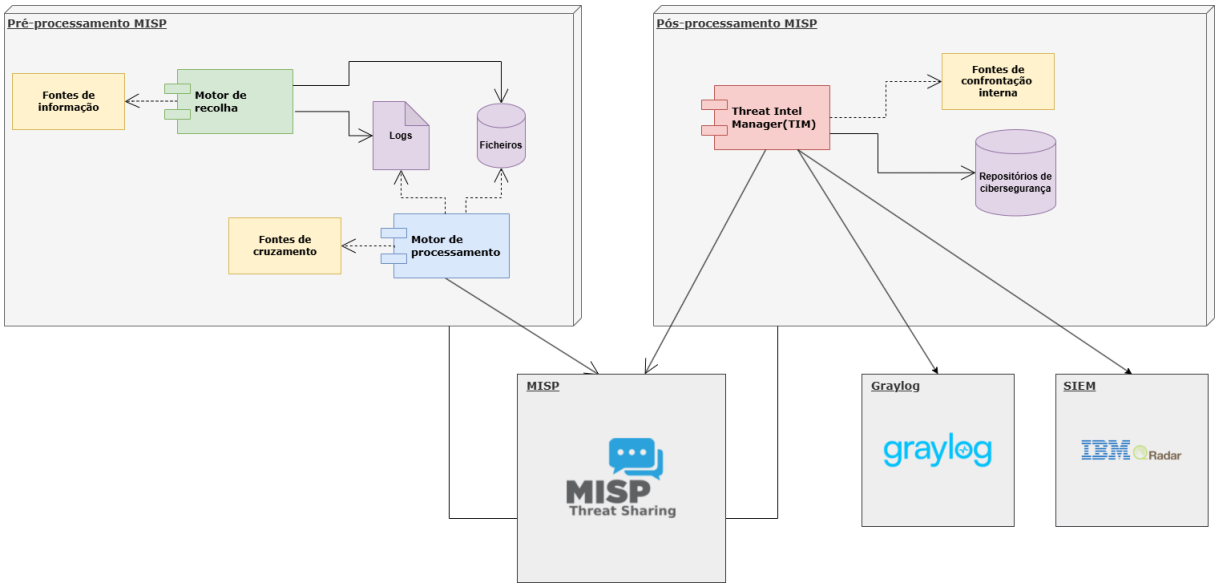


Figura 4.1: Arquitetura do sistema.

- **Motor threat intel manager** - O *threat intel manager* executa dois processos distintos: o primeiro recolhe informações sobre os ativos de diversas fontes, principalmente o software em execução ou instalado. O segundo processo envolve a confrontação das novas informações provenientes do MISP com esse cadastro, incluindo a exportação de IOCs de *malware* para o QRadar SIEM e Graylog.

Arquitetura do motor de recolha

O motor de recolha é formado por vários módulos especializados que operam de forma sequencial, mas independentemente uns dos outros. Isto implica que cada módulo pode falhar autonomamente, sem comprometer o funcionamento dos restantes. Como se observa na Figura 4.2, o fluxo do motor de recolha aplica-se não apenas à componente de vulnerabilidades, mas também às restantes componentes. O mesmo princípio se estende aos outros motores, conforme discutido nesta secção. Contudo, a lógica do processo em cada motor difere. Este processo é cíclico, o que significa que, a cada execução, é possível extrair nova informação. Caso novos dados estejam disponíveis, o ficheiro CSV da fonte correspondente é reescrito com esta nova informação, atualizando-se também o ficheiro de log com um registo dessa atualização. Se não houver novos dados, registar-se-á no ficheiro de log que não ocorreu nenhuma atualização naquele momento específico.

No desenvolvimento dos módulos de recolha, cada módulo interage com dois tipos de ficheiros:

- **Ficheiro de log** - Este ficheiro documenta as ações realizadas pelo módulo, incluindo atualizações efetuadas quando nova informação é recolhida ou, em alternativa, quando não

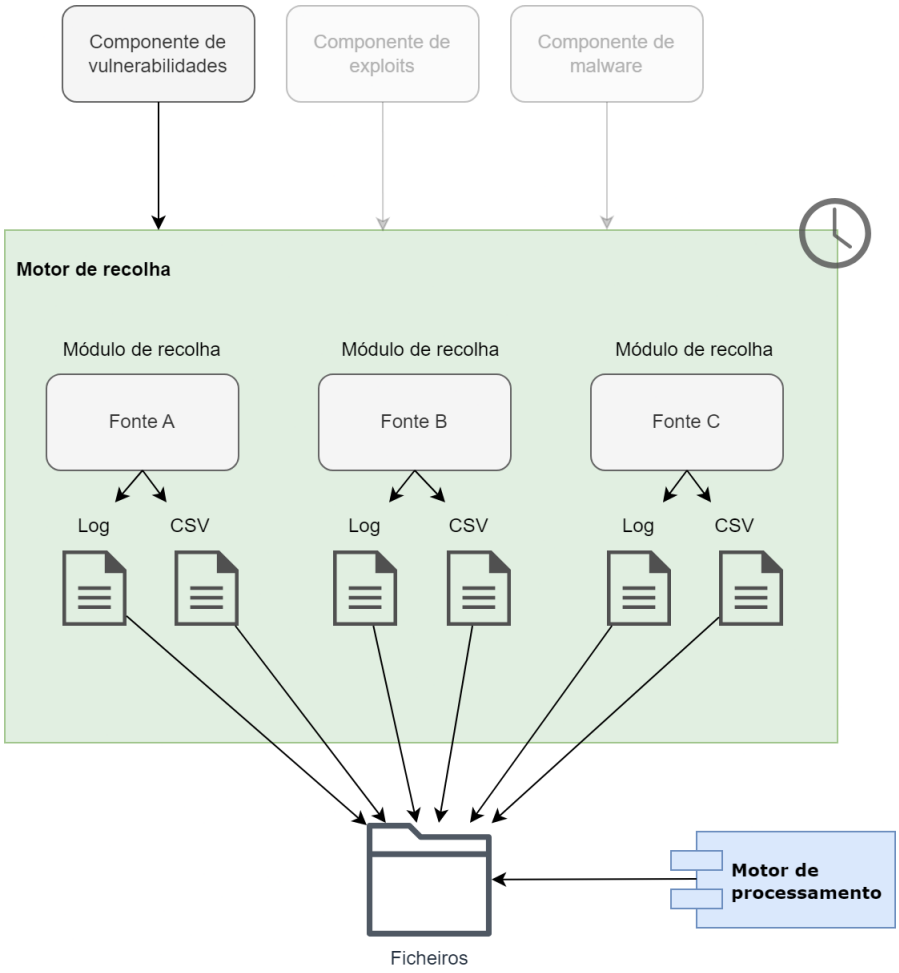


Figura 4.2: Esquema geral do motor de recolha.

há novos dados disponíveis. Este ficheiro regista também eventuais erros que ocorram durante o processo. Além disso, o ficheiro de log é crucial para o motor de processamento, servindo como indicador para determinar quando deve ser ativado o seu processo;

- **Ficheiro CSV (.csv)** - Neste ficheiro é armazenado a informação por parte da fonte em formato *csv* (Comma Separated Values), para que seja utilizado pelo motor de processamento.

O esquema operacional do módulo de recolha segue um padrão consistente que foi adotado para várias fontes, conforme ilustrado na Figura 4.3:

- **Recolha de Informação** - Esta etapa envolve a interação com a fonte de informação para armazenar os dados mais recentes e disponíveis. Durante este processo, também ocorre uma limpeza preliminar dos dados recolhidos, assegurando que a informação seja padronizada;
- **Guardar no ficheiro *csv*** - Caso novos dados sejam obtidos da fonte, estes são guardados num ficheiro *csv*;
- **Atualizar ficheiro de log** - O ficheiro de log é atualizado para refletir as informações mais recentes da última recolha;

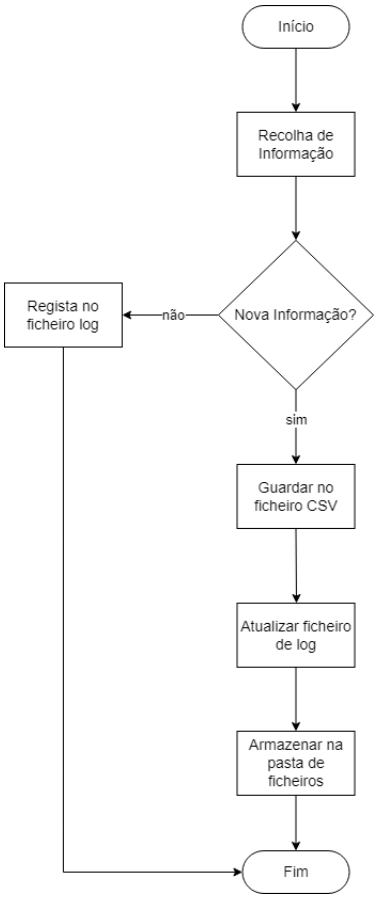


Figura 4.3: Esquema operacional de um módulo de recolha.

- **Armazenar na pasta de ficheiros** - O ficheiro de log e *csv* são guardados na pasta de ficheiros para serem utilizados depois pelo motor de processamento.

Esta metodologia garante a modularidade do sistema e, se for preciso adicionar novas fontes aos módulos de recolha, permite que a manutenção seja realizada de forma escalável.

Arquitetura do motor de processamento

O motor de processamento representado na Figura 4.4, é composto por dois segmentos, o segmento de triagem e análise, e o segmento de cruzamento e normalização.

No segmento de triagem e análise, o processo acede aos ficheiros produzidos pelo motor de recolha de informação e encaminha cada ficheiro para um analisador que está familiarizado com o formato dos dados da respetiva fonte. A presença dos analisadores implica a necessidade de mais alterações no desenvolvimento sempre que se pretenda adicionar uma nova fonte de informação. Contudo, esta abordagem confere maior flexibilidade ao processo, pois não só permite a adição e modificação de informações conforme a origem dos dados recolhidos, mas também possibilita a inclusão de meta informação, como a hora em que a entrada foi analisada. Exemplos desta funcionalidade incluem a atribuição de níveis de fiabilidade às fontes e a normalização de formatos

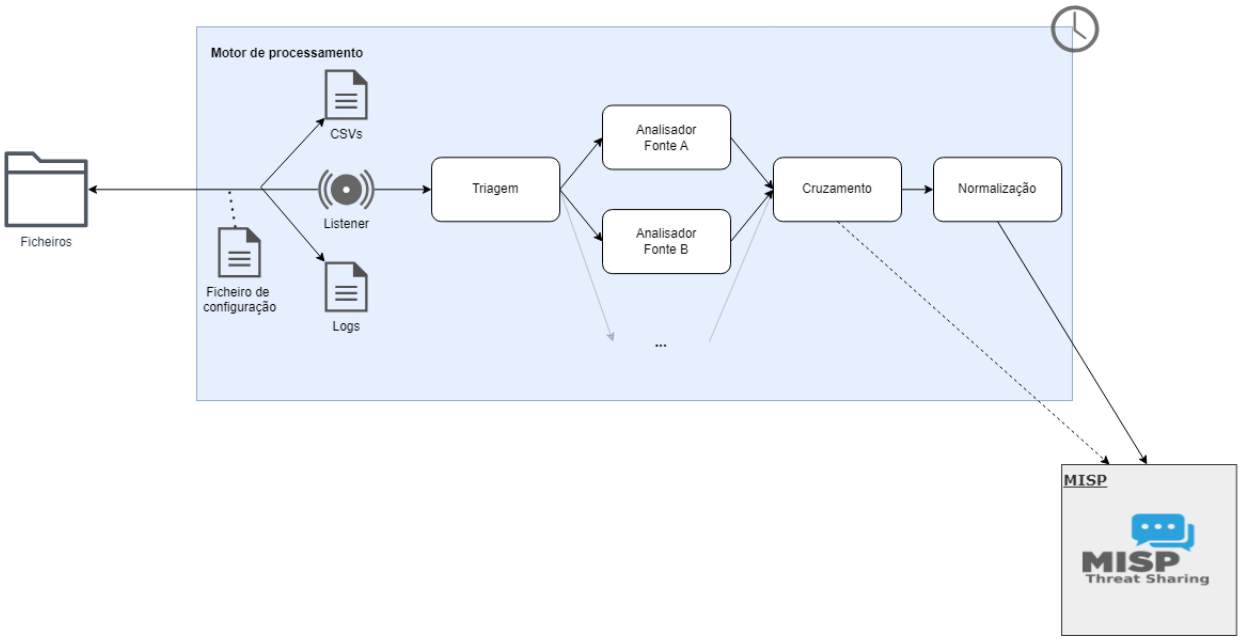


Figura 4.4: Esquema geral do motor de processamento.

de dados distintos, tornando-os compatíveis.

Depois da triagem e análise da informação, os dados entram na segunda parte do motor de processamento, o cruzamento e normalização. Nesta fase, os dados são comparados com as fontes de cruzamento, que irão ajudar a determinar se a ameaça já é conhecida pela organização, mas também com o próprio MISP para não haver duplicados, funcionando assim também como se fosse uma fonte de cruzamento que a longo prazo será mais valiosa. Após o cruzamento, os dados das ameaças dia-zero para a empresa são normalizados de acordo com um template criado especificamente para as necessidades da empresa para cada componente diferente e por fim os dados são armazenados no MISP.

Para otimizar a normalização, procedeu-se à implementação de *templates* de objetos no MISP especificamente concebidos para as componentes de vulnerabilidade, *exploit* e *malware*. Os *templates* foram desenvolvidos considerando a necessidade de uma metodologia consistente que pudesse abranger diversas fontes de informação, as quais frequentemente apresentam estruturas de dados heterogéneas. Como ilustrado na Figura 4.5, a informação proveniente das fontes A e B foi mapeada para um *template* centralizado de vulnerabilidades, que inclui campos essenciais como CVE, CPE, CVSS, plataforma, fornecedor, título, descrição, entre outros. Este mapeamento assegura que independentemente das discrepâncias nas fontes originais, os dados são normalizados e integrados de forma coerente no nosso sistema. O mesmo acontece para a componente de *exploit* e *malware*.

Arquitetura do motor TIM

O motor *Threat Intel Manager* (TIM) é estruturado em dois processos principais. O pri-

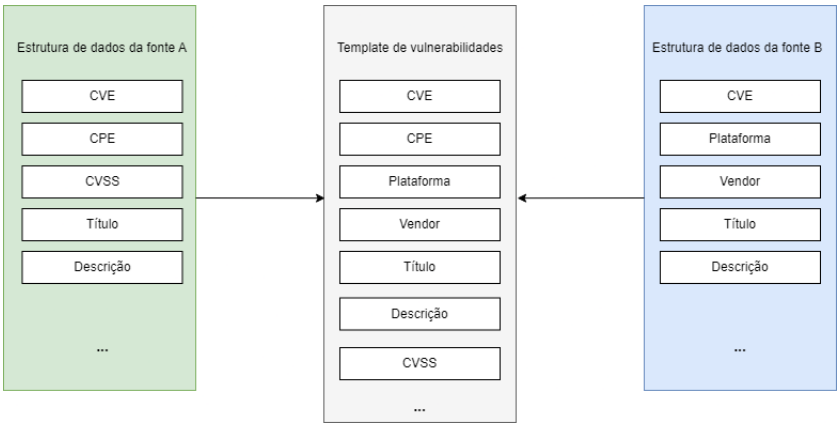


Figura 4.5: Exemplo de normalização dos dados na componente de vulnerabilidades.

meiro processo inicia-se com um componente *listener*, que monitoriza continuamente o MISP para detetar eventos que necessitem de ser processados. Este procedimento envolve a extração dos dados relativamente aos IOCs que são exportados para o Qradar SIEM e Graylog. Isto permitirá no futuro a criação de regras específicas que irão gerir os alertas necessários para a mitigação dos riscos.

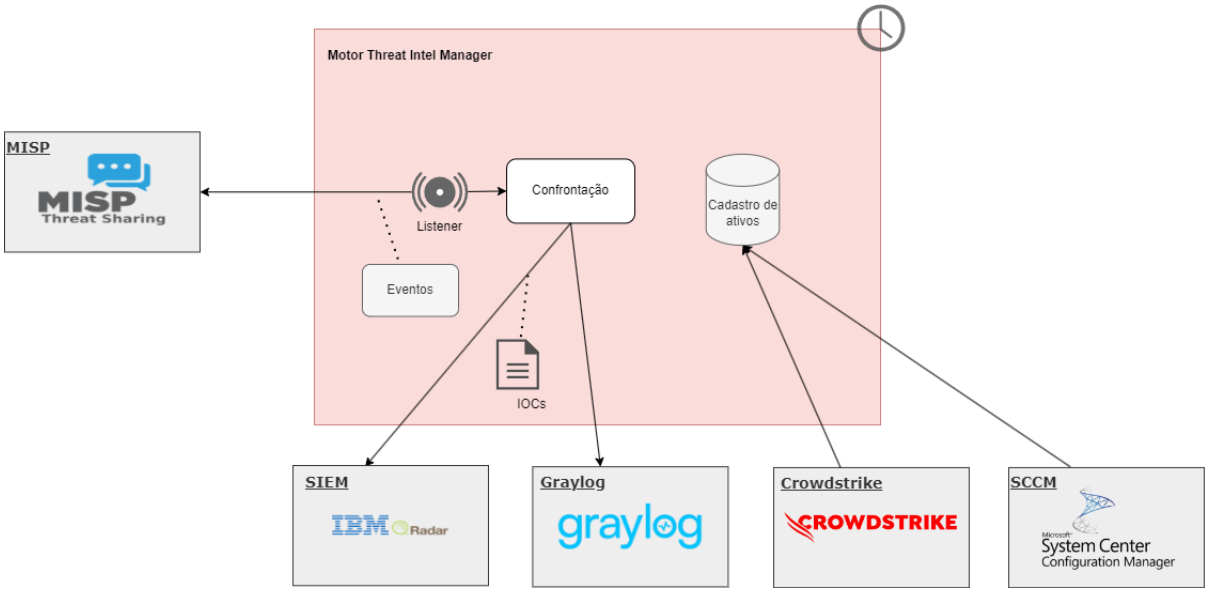


Figura 4.6: Esquema geral do motor TIM.

O segundo processo, que não foi originalmente planeado para este projeto, envolve a agregação de informações sobre os ativos da empresa a partir de diversas fontes externas, incluindo soluções como *CrowdStrike* [2] e SCCM (*System Center Configuration Manager*)[4]. Este processo é vital para enriquecer a base de dados de ativos da empresa com informações atualizadas e diversificadas, potencializando assim a capacidade de resposta a incidentes e a robustez da gestão de ciberameaças.

4.2 Implementação

Com base na arquitetura previamente detalhada, procedeu-se à implementação do sistema. Esta secção abordará as tecnologias e ferramentas empregues na construção do sistema (Subsecção 4.2.1) e detalhará o desenvolvimento dos diferentes motores do sistema (Secções 4.2.2, 4.2.3 e 4.2.4).

4.2.1 Tecnologias e ferramentas utilizadas

Para o armazenamento e visualização de dados, recorreu-se à Elastic Stack [3], que inclui várias tecnologias vantajosas para a manipulação e visualização fácil de grandes quantidades de informação. Dentro do seu ecossistema foram utilizadas as seguintes ferramentas:

- **Kibana**: Facilita a visualização de dados através de uma interface gráfica e permite representá-los em diversos formatos gráficos.
- **Elasticsearch**: Responsável pelo armazenamento e pesquisa de informação, operando através de uma API RESTful.

Para a implementação dos diferentes motores, optou-se por utilizar a linguagem de programação Ruby. A escolha de Ruby permitiu uma maior flexibilidade e eficiência no desenvolvimento dos módulos, dado que Ruby suporta uma vasta gama de bibliotecas reutilizáveis (gemas), facilitando assim a implementação e manutenção dos sistemas:

- **dotenv**: Utilizada para gerir variáveis de ambiente de forma segura;
- ***watir* e *selenium-webdriver***: Facilitam a automação de interações com navegadores web para testes e scraping de dados, com o Watir proporcionando uma interface de alto nível e o Selenium permitindo um controle mais detalhado;
- ***nethttp* e *uri***: Essenciais para realizar requisições HTTP e manipular URIs, respectivamente, facilitando a comunicação com APIs externas.
- ***nokogiri***: Permite parsear e manipular documentos HTML e XML, essencial para o processamento de dados estruturados obtidos de páginas web;
- ***rest-client***: Simplifica a realização de chamadas HTTP para APIs, melhorando a clareza e a manutenção do código de integração com serviços web;
- ***logger***: Permite o rastreamento e a gravação de eventos e erros de forma eficaz;
- ***elasticsearch***: Facilita operações complexas de procura e análise de grandes volumes de dados;
- ***time***: Usada para a manipulação avançada de datas e horas;
- ***digest***: Proporciona a criação de hashes de texto;

- ***singleton***: Aplica o padrão de design Singleton, garantindo que uma classe tenha uma única instância em todo o sistema;
- ***concurrent-ruby***: Oferece suporte a operações paralelas e assíncronas, melhorando significativamente a eficiência e o desempenho em tarefas que podem ser paralelizadas;
- ***text***: Utilizada especificamente para algoritmos de comparação de strings, como a distância de Levenshtein, importante para funções de procura e comparação de texto;
- ***csv, json, yaml***: Suportam o processamento e a serialização/deserialização de dados em formatos CSV, JSON e YAML, respectivamente, facilitando a interoperabilidade de dados.

4.2.2 Motor de recolha

O desenvolvimento do sistema iniciou-se pelo motor de recolha, responsável por recolher as informações das diversas fontes de informação. Nesta secção será explicada a implementação dos módulos de recolha das componentes de vulnerabilidades, *exploits* e *malware*.

Recolha de vulnerabilidades

Na Figura 4.7 foi ilustrado a sequência para a recolha de informação da fonte VulDB. O processo inicia-se com a verificação do ficheiro de registo da última data em que a extração foi realizada. Caso seja a primeira execução, a data é definida para o dia corrente. A partir daí, a tarefa principal é obter novas informações de vulnerabilidades que foram publicadas após a última consulta realizada. Através da API, o programa faz um requisição que solicita dados de vulnerabilidades que foram publicadas após a última execução, utilizando o *timestamp* dessa última execução como filtro, em caso de ser a primeira vez seria as vulnerabilidades do próprio dia. Estes dados são recebidos em formato JSON, o que facilita a manipulação e extração de informações específicas.

O programa verifica se existem novas entradas comparando os identificadores de cada vulnerabilidade recebida com o último registado no ficheiro local. Se identificar novas vulnerabilidades, estas são processadas e armazenadas temporariamente. Em seguida, os novos dados são adicionados a um ficheiro *csv* . Este ficheiro serve como um registo cumulativo de todas as vulnerabilidades identificadas desde a última execução do programa. A atualização é realizada de forma cuidadosa para assegurar que não haja perda de dados, substituindo o ficheiro antigo pelo novo já atualizado.

Finalmente, o programa atualiza o ficheiro de registo com a data e hora da execução atual, garantindo que futuras execuções continuem a partir deste ponto. Também regista a conclusão do processo no ficheiro de log com o *timestamp*, o que permite ao motor de processamento saber quando existe nova informação para ser processada, isto acontece para todos os módulos de recolha das fontes.

Os dados recolhidos da fonte são os seguintes:

- **ID da entrada:** Identificador único da vulnerabilidade;

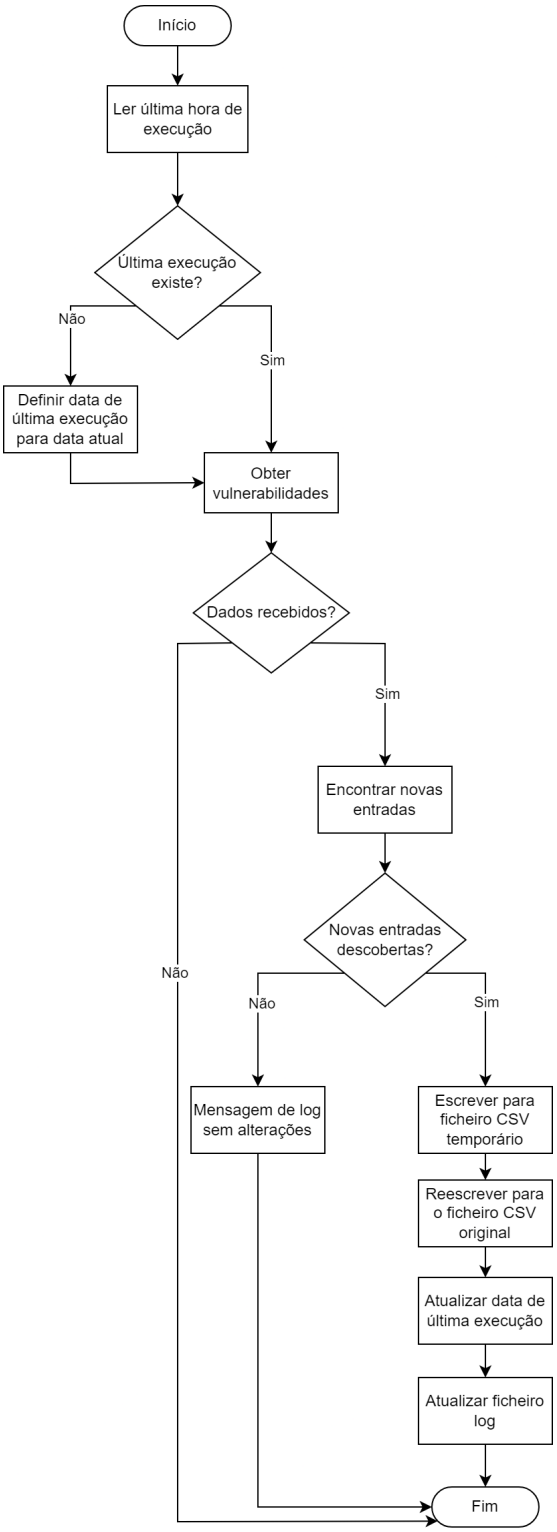


Figura 4.7: Fluxograma da recolha de informação da fonte VulDB.

- **Título da entrada:** Título descritivo da vulnerabilidade;
- **Resumo da entrada:** Resumo da vulnerabilidade;
- **Detalhes das contramedidas da entrada:** Detalhes sobre contramedidas recomendadas;
- **Data e hora de criação da entrada:** Data e hora de criação do registro da vulnerabilidade;
- **Fornecedor do software:** Fornecedor do software afetado;
- **Nome do software:** Nome do software afetado;
- **CPE do software:** Código de identificação do produto conforme o padrão *Common Platform Enumeration*;
- **Pontuações CVSS da Vulnerabilidade:** Para quantificar a severidade da vulnerabilidade (a versão mais atual das pontuações CVSS na VulDB utilizada é a versão 3 de momento);
- **URL do exploit:** URL para o exploit, se disponível;
- **URL do patch de contramedida:** URL para o *patch* de correção, se disponível;
- **URL de workaround:** URL para uma solução alternativa, se disponível;
- **CVE de Origem:** Identificador CVE associado, se disponível.

O procedimento de recolha de dados da fonte ZDI (Figura 4.8) inicia-se com a verificação da existência de um ficheiro *csv* previamente estabelecido, destinado ao armazenamento das informações extraídas. Na ausência deste ficheiro, o programa cria um novo ficheiro, inserindo cabeçalhos.

Uma vez preparado o ficheiro, o programa emprega um navegador automatizado para aceder à página de *advisories* da ZDI. Identificam-se então os links que conduzem às páginas individuais de cada *advisory*. O programa navega para cada página ligada e extrai uma gama de informações detalhadas relativas à vulnerabilidade. Estes dados são armazenados temporariamente, aguardando validação antes de serem efetivamente registados no ficheiro *csv*.

Durante a execução, verifica-se se o identificador da vulnerabilidade (ZDI ID) corresponde ao último recolhido. Se houver discrepância, presume-se que a entrada é nova, adicionando-a ao ficheiro *csv* temporário. Este processo é repetido até que todos os links sejam analisados ou até identificar-se uma correspondência com o último ZDI ID recolhido, sinalizando que as entradas subsequentes já foram previamente recolhidas.

Após a recolha das novas entradas, caso existam novos dados, o ficheiro *csv* temporário é utilizado para atualizar o ficheiro original. Isso garante que somente as informações novas sejam incorporadas. Caso não haja novas entradas, o programa regista uma mensagem no log, indicando a ausência de alterações. O processo é finalizado com a atualização do ficheiro de log, que regista o *timestamp* do processo.

A ZDI possui um conjunto de dados menos detalhado em comparação com a VULDB. A ZDI não fornece, por exemplo, o CPE. No entanto, informações cruciais como o fornecedor afetado, o

produto, a pontuação CVSS e detalhes adicionais são fornecidos e recolhidos através do de *web scraping*. A extração desses dados é realizada com o uso de expressões regulares e métodos de busca específicos, adaptados à estrutura do HTML da página da ZDI.

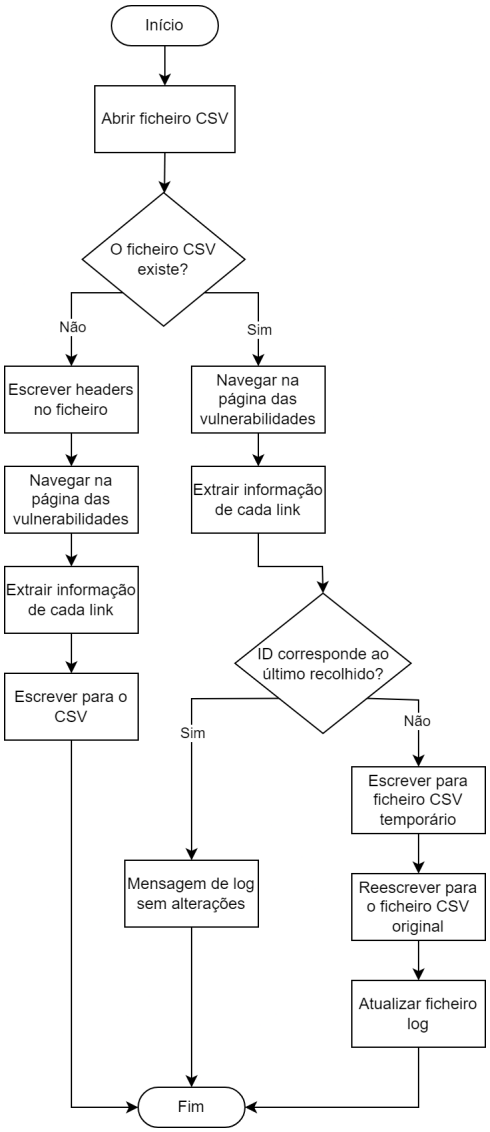


Figura 4.8: Fluxograma da recolha de informação da fonte ZDI.

Para cada campo relevante, o programa utiliza seletores específicos e *regex* para capturar as informações necessárias de forma eficaz que se encontra no Apêndice A.

Recolha de exploits

Para distinguir e identificar *exploits* idênticos, seria essencial que as fontes de informação disponibilizassem uma assinatura do ficheiro contendo o exploit. Contudo, apenas a fonte PSS fornece tal resumo, tornando difícil proceder de forma simplificada noutros casos. A possibilidade de criar uma assinatura a partir do ficheiro completo foi considerada, mas enfrenta limitações práticas

porque, apesar de as fontes fornecerem o mesmo *exploit*, frequentemente alteram o conteúdo do ficheiro adicionando texto adicional. Um exemplo claro é a ZDT, que remove as datas dos *exploits* para as colocar no final do ficheiro.

Na Figura 4.9, observa-se um exemplo deste problema, onde as três fontes de informação possuem o mesmo *exploit*, mas as alterações na formatação e a adição de texto resultam em ficheiros diferenciados.

```
$> diff -u EDB.txt PSS.txt | diffstat -m
PSS.txt | 75 ++!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
1 file changed, 3 insertions(+), 72 modifications(!)

$> diff -u EDB.txt ZDT.txt | diffstat -m
ZDT.txt | 1 -
1 file changed, 1 deletion(-)

$> diff -u PSS.txt ZDT.txt | diffstat -m
ZDT.txt | 75 ---!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
1 file changed, 4 deletions(-), 71 modifications(!)
```

Figura 4.9: Diferenças entre entradas de exploits iguais (appRain CMF 4.0.5) das fontes ZDT, PSS e EDB.

Para abordar este desafio, foi desenvolvido um sistema de comparação de *exploits* parecido ao projeto ZeroDays V2 que inicialmente realiza uma normalização do conteúdo do *exploit*. Este processo de limpeza, representado no Apêndice B, foi meticulosamente adaptado para remover alterações comuns efetuadas por cada uma das fontes, além de eliminar todos os caracteres especiais presentes no corpo do *exploit* que podem levar a erros. Os caracteres removidos incluem todos os invisíveis, espaços e quebras de linha. Além disso, são eliminadas todas as linhas que começam com o caractere ”#”(cardinal), frequentemente usado pelas fontes para indicar comentários. Esta limpeza baseia-se numa expressão regular detalhada no Apêndice A.

Após estas modificações, é possível equiparar dois *exploits* de fontes diferentes através de um resumo SHA-256 comum. Este resumo foi denominado de *Comparison Hash* (CHash), para distingui-lo dos resumos fornecidos pelo PSS.

Na Figura 4.10 são demonstrados os principais passos para a recolha de informação da fonte Vulners. O procedimento inicia-se com a verificação da existência de ficheiros *csv* específicos para cada fonte de dados: ZDI, EDB e PSS. Estes ficheiros são essenciais pois armazenam as informações extraídas separadamente, respeitando a origem específica de cada conjunto de dados. Caso algum dos ficheiros não exista, o programa cria este ficheiro e prepara-o para receber dados, garantindo que o processo de armazenamento esteja alinhado com a fonte correspondente.

Com os ficheiros prontos, o programa lê o último identificador do exploit (ID) registado em cada um para determinar o ponto de partida para novas consultas. Este método evita a duplicação de dados e garante a continuidade da recolha de informações.

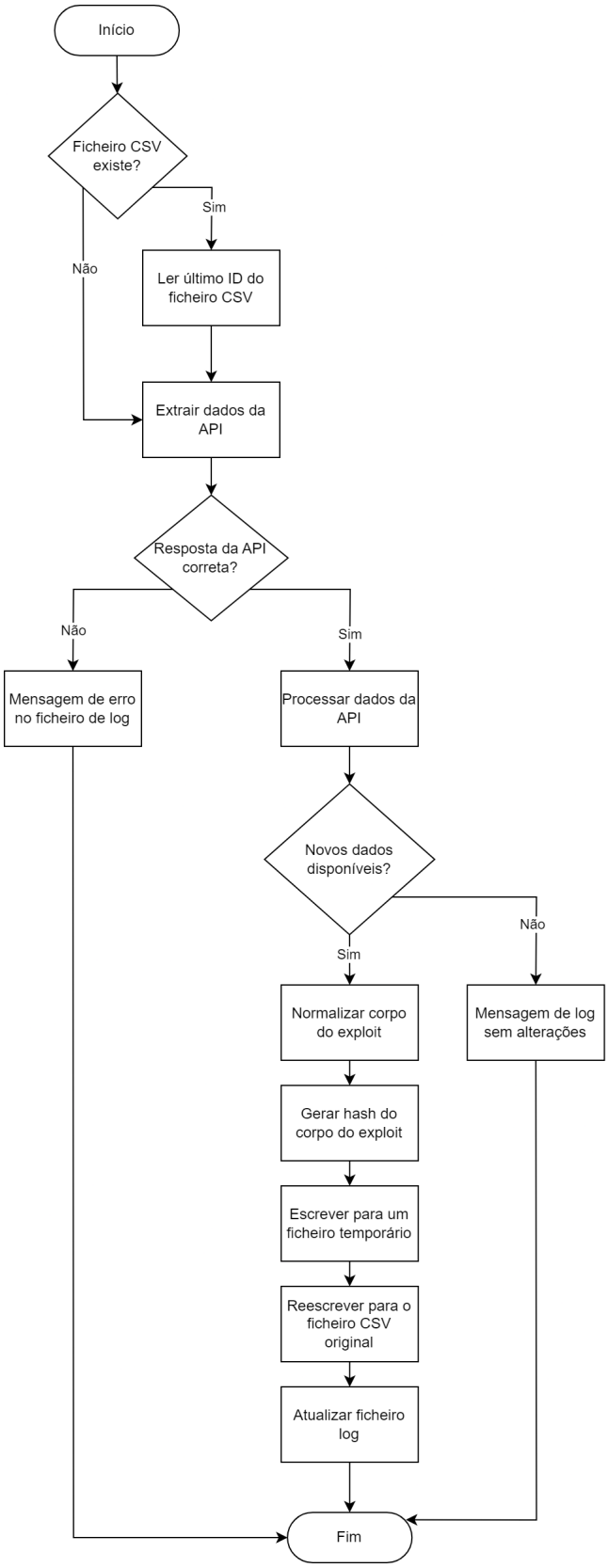


Figura 4.10: Fluxograma da recolha de informação da fonte Vulners.

O programa faz uso da API da Vulners, que serve como um intermediário vital para aceder aos dados das fontes ZDI, EDB e PSS, que por si só não disponibilizam APIs próprias. Através desta API unificada, o programa solicita novas informações de exploits baseando-se nos últimos IDs conhecidos para cada fonte.

Ao receber os dados da API, o programa avalia a correção e integridade das informações. Se a resposta for adequada, inicia-se o processamento dos dados extraídos. Este processamento inclui a normalização do conteúdo dos exploits e a subsequente geração de um hash para cada um como explicado anteriormente.

Novos dados são inicialmente escritos em um ficheiro *csv* temporário correspondente à fonte. Este passo assegura que todas as informações possam ser verificadas e validadas antes de serem definitivamente incorporadas ao ficheiro *csv* principal de cada fonte, tal como foi realizado nas fontes anteriores.

Finalmente, o programa transfere os dados validados do ficheiro temporário para o ficheiro original e atualiza o ficheiro de log. Esta etapa final documenta todas as operações realizadas e quaisquer novos dados adicionados, mantendo um registo detalhado que facilita revisões futuras e manutenção do sistema.

Os dados recolhidos da fonte são os seguintes:

- **ID do exploit:** O identificador único atribuído ao exploit nas diferentes bases de dados;
- **Título do exploit:** Descrição concisa e informativa do exploit;
- **Data de Publicação:** Quando o exploit foi divulgado ou atualizado pela última vez;
- **Descrição do exploit:** Um resumo detalhado do que o exploit faz e como ele opera;
- **CVSS Score:** A pontuação no sistema de pontuação de vulnerabilidades comuns, indicando a severidade do exploit;
- **Código do exploit:** O código que é executado no exploit (apenas utilizado para criar o CHash e plataforma afetada);
- **Plataforma afetada:** O sistema ou aplicação que é alvo do exploit, retirado do código do exploit caso seja possível;
- **CVE:** O identificador CVE associado ao exploit, se disponível;
- **URL de referência:** Link de origem do exploit;
- **CHash:** O hash gerado após a normalização do exploit, usado para comparações eficazes;

Recolha de malware

Na Figura 4.11 são demonstrados os principais passos para a recolha de informação da fonte INQ.

O processo inicia-se com uma requisição à API da INQ para a extração de dados relevantes. Se a API responder corretamente, o programa prossegue para verificar a existência de novos dados. Caso não sejam encontrados dados novos, o sistema regista uma mensagem de *log* indicando que não foram feitas alterações.

Quando novos dados estão disponíveis, o programa verifica se o ficheiro *csv* necessário já existe. Se existir, o último *hash* registado é utilizado como referência para o processamento subsequente. Caso contrário, as amostras são inicialmente filtradas com base em critérios temporais.

O passo seguinte envolve o enriquecimento das amostras com IOCs, que é realizado de maneira eficiente através de concorrência com threads. Este método permite que múltiplas requisições à API sejam processadas em paralelo, acelerando significativamente a recolha e integração dos IOCs. Como demonstrado na Tabela 4.1, a utilização de threads (5 utilizadas) reduz o tempo de processamento de aproximadamente 300 segundos para 50 segundos para 254 entradas, evidenciando a eficácia desta abordagem em otimizar o desempenho do sistema.

Método de Processamento	Entradas	Tempo Estimado (segundos)
Sem Threads	254	$\cong 306$
Com Threads	254	$\cong 55$

Tabela 4.1: Estimativa de tempo de processamento com e sem threads

Após a integração dos IOCs, os dados são escritos num ficheiro temporário para garantir a integridade e precisão antes da sua transferência definitiva para o ficheiro *csv* original. Este ficheiro é então atualizado, e as mudanças são refletidas no ficheiro de *log*.

Os dados recolhidos da INQ são os seguintes:

- **Data de Conclusão da Análise**: Data e hora em que a análise do *malware* foi concluída;
- **Classificação**: Classificação do *malware* conforme definido pela fonte(malicioso, suspeito, desconhecido);
- **Tipo de Ficheiro**: Tipo do ficheiro associado ao *malware*, indicativo do método de infeção ou tecnologia utilizada;
- **Primeira Observação**: Data em que o *malware* foi observado pela primeira vez;
- **Tipo MIME**: Tipo MIME do ficheiro, que oferece uma classificação adicional;
- **SHA256**: Hash sha256 do ficheiro, utilizado como identificador único para o *malware*;
- **Tamanho**: Tamanho do ficheiro;
- **Subcategoria**: Subcategoria do *malware*, fornecendo detalhes específicos sobre a natureza da ameaça;
- **URL da Subcategoria**: URL que proporciona informações adicionais sobre a subcategoria do *malware*;

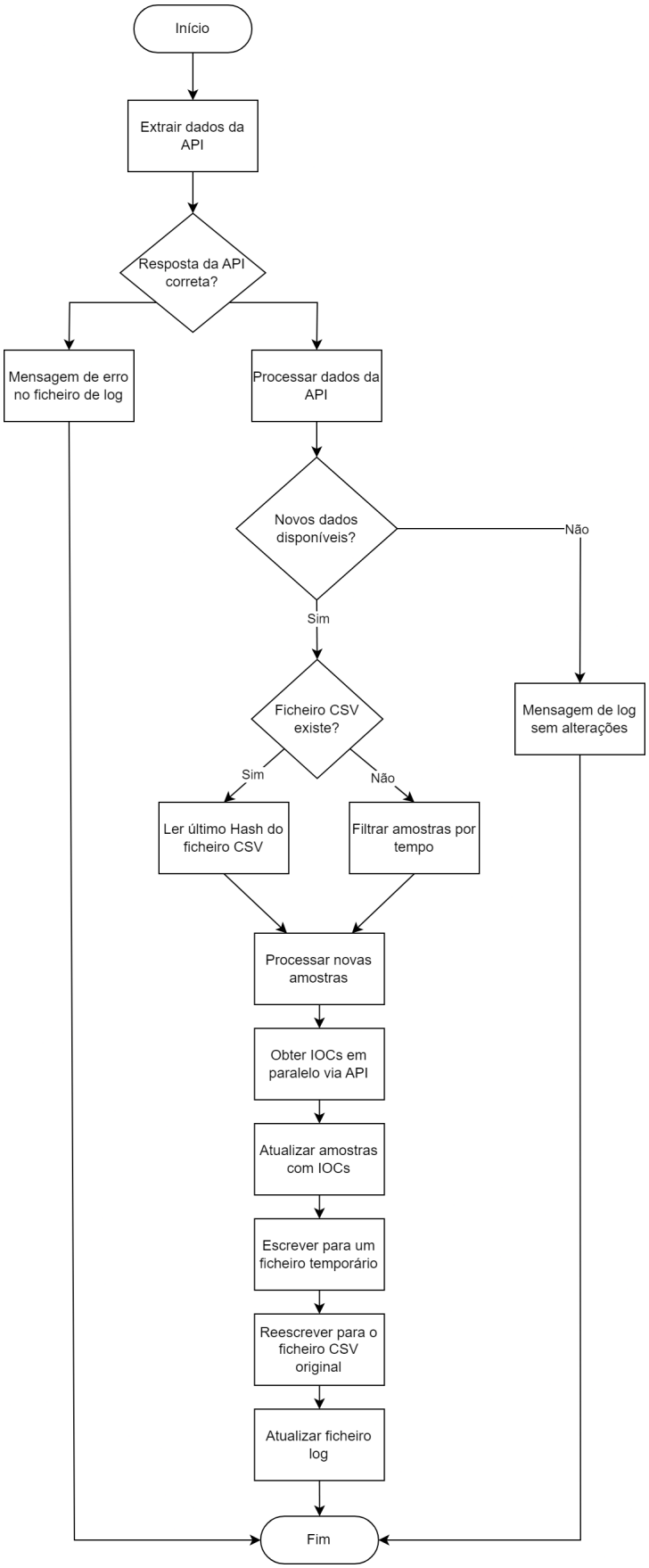


Figura 4.11: Fluxograma da recolha de informação da fonte INQ.

Os IOCs que foram recolhidos em paralelo e adicionados ao ficheiro *csv* incluem:

- **Hashes de Ficheiros**: Identificadores únicos dos ficheiros de *malware*.
- **Nomes de Ficheiros**: Nomes de ficheiros associados ao *malware*, utilizados para detetar infecções.
- **URLs**: Endereços URL maliciosos usados pelo *malware* para comunicação ou infeção.
- **Domínios**: Domínios associados ao *malware* que podem estar envolvidos em atividades maliciosas.
- **IPs**: Endereços IP relacionados ao *malware*.
- **Emails**: Endereços de email usados nas operações do *malware*.
- **Caminhos de Ficheiros**: Localizações específicas de ficheiros no sistema que são usadas pelo *malware*.

O processo de extração e processamento de dados da fonte MAB segue uma metodologia muito similar à aplicada com a INQ, conforme descrito anteriormente. A principal diferença reside da não existência de IOCs. A fonte tem um campo que a fonte anterior não contém, nomeadamente as *tags*. Este campo contém etiquetas que classificam e descrevem as características relevantes de cada amostra de *malware*, como o tipo de ficheiro, o nome específico do *malware*, a sua categoria e possíveis características geográficas associadas.

4.2.3 Motor de processamento

Após o motor de recolha gerar os ficheiros *csv* e de *log*, entra em funcionamento o motor de processamento, que irá monitorizar o ficheiro de log em procura de atualizações. Se houver informação a ser processada e cruzada, este motor entra em ação. Nesta secção será explicado como é feita a triagem e interpretação da informação como também o seu cruzamento com a informação interna para determinar ameaças dia-zero para cada componente, sendo estas armazenadas no MISP.

Triagem e interpretação da informação

No âmbito deste trabalho, foi necessário encontrar um método eficaz para detetar a existência de nova informação disponível dentro dos ficheiros CSV. Inicialmente, foram consideradas várias abordagens, incluindo a verificação da data de modificação do ficheiro, o cálculo de *hashes* para identificar mudanças, guardar o ID da última entrada recolhida num ficheiro à parte e utilizar um ficheiro com uma flag sempre que há uma recolha de informação. No entanto, estas opções foram descartadas por não garantirem a fiabilidade necessária e por não serem completamente eficazes na deteção de informações duplicadas ou de erros.

Para superar estas limitações, decidiu-se implementar uma solução baseada num ficheiro de *log*. Este ficheiro de *log* não só regista quando há uma atualização no ficheiro CSV, como também funciona como um *debugger* para erros. Esta abordagem oferece várias vantagens:

- **Fiabilidade**: O ficheiro de *log* permite monitorizar atualizações de forma precisa e consistente, garantindo que todas as mudanças são registadas.
- **Debugging**: Além de indicar atualizações, o *log* regista erros e eventos importantes, facilitando a identificação e resolução de problemas.
- **Flexibilidade**: Esta solução é mais adaptável a diferentes fontes de dados e a possíveis alterações nos sistemas de identificação das fontes.

Assim, a escolha de um ficheiro de *log* para monitorização e debug representa uma melhoria significativa em relação aos métodos considerados anteriormente, assegurando uma gestão mais eficaz e robusta das atualizações de dados.

Processamento de vulnerabilidades

Após a uniformização da informação, as vulnerabilidades são processadas e cruzadas com as ferramentas de deteção de ameaças internas para determinar se são ameaças dia-zero. Este processamento segue uma sequência de passos específicos para garantir a integridade e a relevância dos dados.

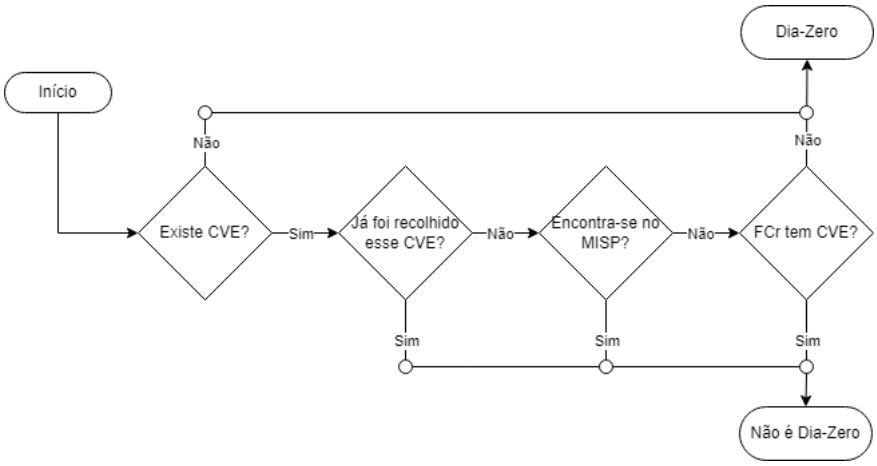


Figura 4.12: Cruzamento de vulnerabilidades com as FCr.

Primeiramente, para cada nova entrada de vulnerabilidade, verifica-se se essa vulnerabilidade possui um identificador CVE. Caso não exista um identificador CVE, a vulnerabilidade é automaticamente considerada como dia-zero, indicando que ainda não possui um reconhecimento formal e, portanto, representa uma ameaça potencial significativa.

Se a vulnerabilidade possuir um identificador CVE, verifica-se se já foi processada anteriormente no ciclo para remover duplicados e evitar redundâncias no sistema. Em seguida, verifica-se

se essa entrada já está presente no MISP, funcionando como um *backlog*.

Após estas verificações iniciais, a vulnerabilidade é confrontada com os registos da base de dados do *Qualys*. Estes registos mantêm uma base de conhecimento sobre vulnerabilidades que já foram identificadas e registadas durante a execução das ferramentas. O cruzamento do identificador CVE com um correspondente nos registos internos permite determinar se a vulnerabilidade já está coberta pelas ferramentas de segurança da empresa.

Se não houver correspondência entre a vulnerabilidade recolhida e os registos internos, a vulnerabilidade é considerada como dia-zero no contexto da empresa e armazenada no MISP.

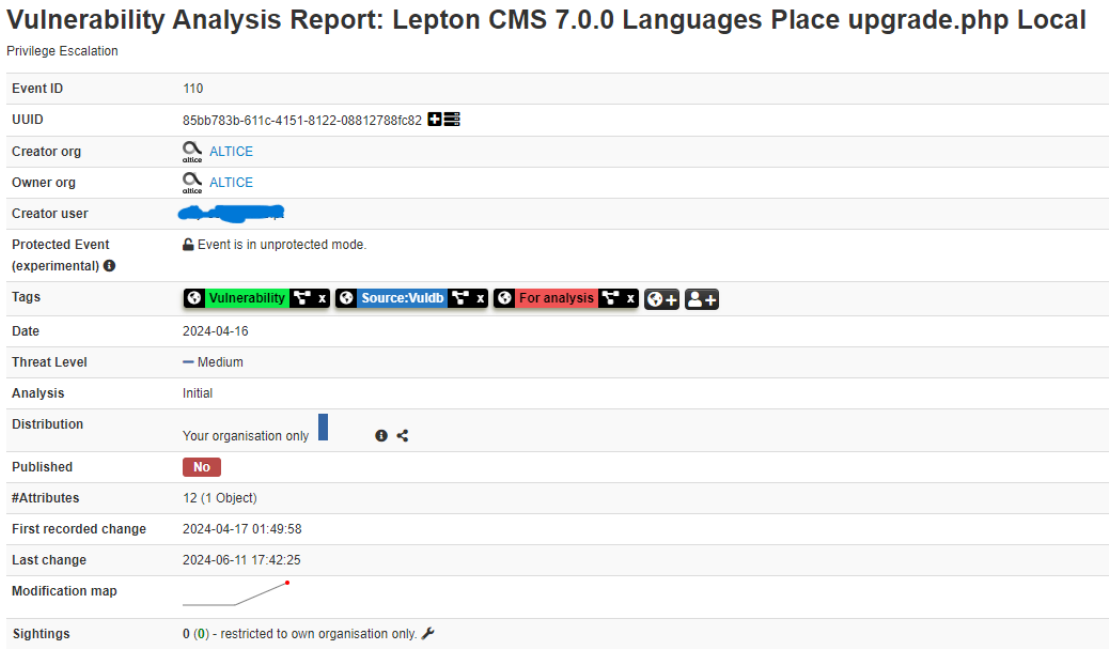


Figura 4.13: Evento de uma vulnerabilidade da fonte VulDB no MISP.

Ao criar uma entrada de vulnerabilidade no MISP (Figura 4.13), são atribuídas etiquetas (*tags*) e um nível de risco (*Threat Level*). As etiquetas criadas para esta componente incluem *"vulnerability"* para facilitar a filtragem pelo tipo de ameaça, *"source:Font"* para filtrar pela fonte, e, caso a vulnerabilidade ainda não tenha sido processada pelo motor de processamento, uma etiqueta temporária denominada "*For analysis*".

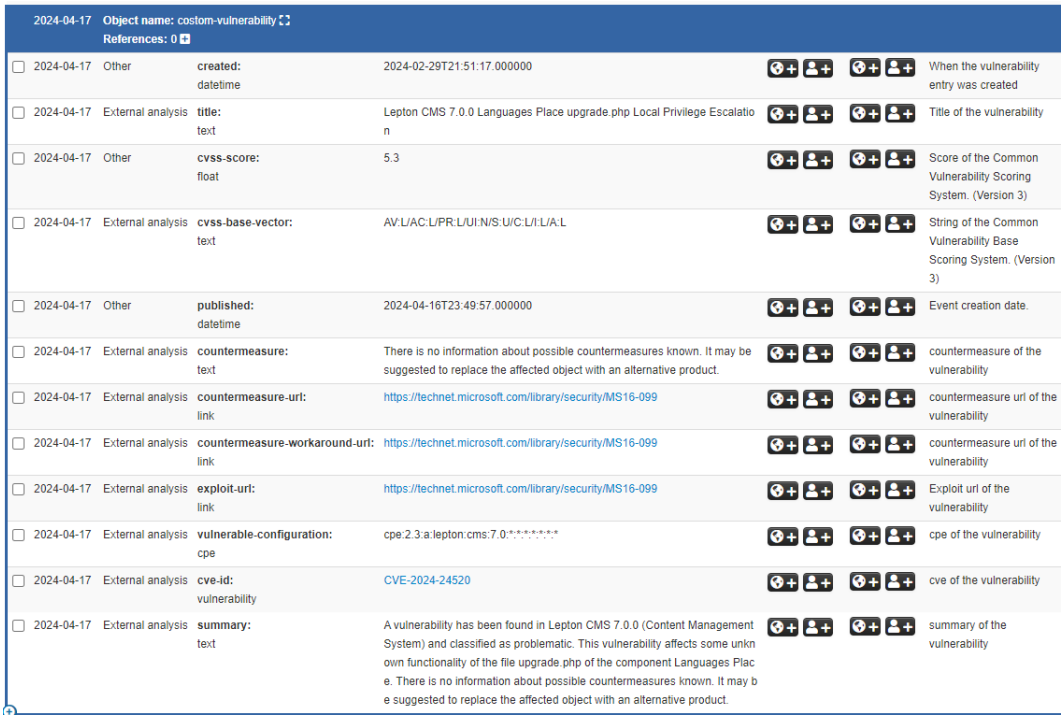
CVSS Score	Nível de severidade
0.0 - 3.9	Baixo (Low)
4.0 - 6.9	Médio (Medium)
7.0 - 10.0	Alto (High)

Tabela 4.2: Intervalos do CVSS Score e correspondente nível de severidade.

O nível de severidade de um evento no MISP pode variar entre baixo (*low*), médio (*medium*) ou alto (*high*). Utilizando o campo CVSS, conseguimos calcular o nível de severidade de uma ameaça através da métrica CVSS Score (Tabela 4.2). Esta informação é meramente informativa

não sendo utilizada de momento em nenhum processo.

Na Figura 4.13, são apresentados os campos que compõem um evento. Na Figura 4.14, podem-se observar os atributos agregados ao mesmo evento que representam a ameaça, de acordo com o *template* criado no MISP para representar uma vulnerabilidade na organização. Embora não sejam exibidos todos os campos do *template* criado para vulnerabilidades, quase todos os campos relevantes estão representados.



2024-04-17	**Object name:** costom-vulnerability **References:** 0					
2024-04-17	Other	**created:** datetime	2024-02-29T21:51:17.000000			When the vulnerability entry was created
2024-04-17	External analysis	**title:** text	Lepton CMS 7.0.0 Languages Place upgrade.php Local Privilege Escalation			Title of the vulnerability
2024-04-17	Other	**cvss-score:** float	5.3			Score of the Common Vulnerability Scoring System. (Version 3)
2024-04-17	External analysis	**cvss-base-vector:** text	AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:L			String of the Common Vulnerability Base Scoring System. (Version 3)
2024-04-17	Other	**published:** datetime	2024-04-16T23:49:57.000000			Event creation date.
2024-04-17	External analysis	**countermeasure:** text	There is no information about possible countermeasures known. It may be suggested to replace the affected object with an alternative product.			countermeasure of the vulnerability
2024-04-17	External analysis	**countermeasure-url:** link	https://technet.microsoft.com/library/security/MS16-099			countermeasure url of the vulnerability
2024-04-17	External analysis	**countermeasure-workaround-url:** link	https://technet.microsoft.com/library/security/MS16-099			countermeasure url of the vulnerability
2024-04-17	External analysis	**exploit-url:** link	https://technet.microsoft.com/library/security/MS16-099			Exploit url of the vulnerability
2024-04-17	External analysis	**vulnerable-configuration:** cpe	cpe:2.3:a:lepton:cms:7.0:*:*:*:*:*:*:*			cpe of the vulnerability
2024-04-17	External analysis	**cve-id:** vulnerability	CVE-2024-24520			cve of the vulnerability
2024-04-17	External analysis	**summary:** text	A vulnerability has been found in Lepton CMS 7.0.0 (Content Management System) and classified as problematic. This vulnerability affects some unknown functionality of the file upgrade.php of the component Languages Place. There is no information about possible countermeasures known. It may be suggested to replace the affected object with an alternative product.			summary of the vulnerability

Figura 4.14: Atributos do evento da vulnerabilidade intitulada “Lepton CMS 7.0.0 Languages Place upgrade.php Local”.

Processamento de exploits

Para o cruzamento de *exploits* (Figura 4.15), para determinar se são ameaças dia-zero, este processo segue uma sequência de passos mais extensa que a das vulnerabilidades.

Para efetuar a correlação de dados com sucesso, é crucial obter dados que identifiquem a ameaça. Considerando os identificadores empregados nas fontes internas, os identificadores CVE das vulnerabilidades exploradas pelo *exploit* são os únicos utilizáveis. Na ausência destes, não é possível confirmar o reconhecimento interno da ameaça.

Portanto, estabeleceram-se os seguintes critérios para determinar se um *exploit* é considerado dia-zero:

- **Consulta ao MISP:** Se o *exploit* não for encontrado no MISP e não foi recolhido anteriormente na mesma iteração com o mesmo CHash, prossegue-se para a análise de CVE;
- **Verificação de iterações anteriores:** Se nenhum *exploit* com o mesmo CHash ou o título do

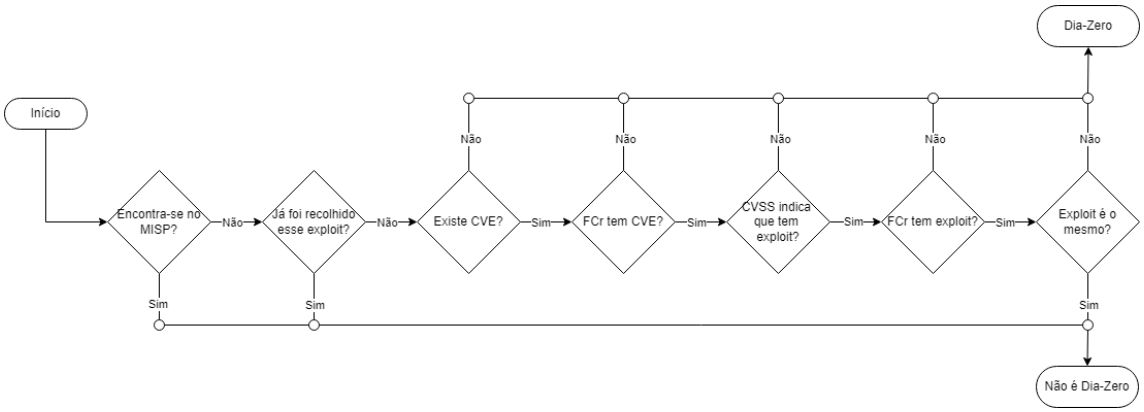


Figura 4.15: Cruzamento de exploits com as FCr.

mesmo não apresenta uma semelhança superior a 70% com outros títulos de *exploits* recolhidos (utilizando a distância de Levenshtein [1]), prossegue-se para a verificação de existência de um CVE, caso contrário não é dia-zero;

- **Existência de CVE:** Se não existir um ID de CVE que identifique a vulnerabilidade explorada, o *exploit* é considerado dia-zero. Caso contrário, verifica-se a presença do CVE na FCr;

- **Verificação de CVE na FCr:** Se a FCr não possuir informação sobre o CVE explorado pelo *exploit*, o mesmo é considerado dia-zero. Se o CVE estiver listado, verifica-se o CVSS;

- **Avaliação de CVSS:** Se o CVSS associado ao CVE não indicar a existência de um *exploit*, o mesmo é considerado dia-zero. Caso contrário, verifica-se se o *exploit* específico está presente na FCr;

- **Verificação de exploit na FCr:** Se não houver nenhum *exploit* listado para o CVE na FCr, o mesmo é considerado dia-zero. Se houver, compara-se o CHash do *exploit* recolhido;

- **Comparação de exploit:** Se não houver uma entrada na lista de *exploits* da FCr com um CHash igual ao recolhido, o mesmo é considerado dia-zero.

A Tabela 4.3 resume os resultados da aplicação do método de distância de Levenshtein para identificar *exploits* duplicados com diferentes limiares de semelhança. Esta análise foi realizada numa amostra de 20 *exploits*, dos quais se identificaram 10 duplicados. Esta análise é feita somente nos *exploits* que foram recolhidos durante a iteração, uma vez que seria demasiada carga computacional fazer esta verificação a todos os registos armazenados no MISP. A análise dos resultados, conforme apresentada na Tabela 4.3, evidencia que diferentes limiares de semelhança têm impactos significativos na identificação de duplicados e na ocorrência de falsos positivos. Com um

[1] A **distância de Levenshtein** é uma métrica utilizada para medir a diferença entre duas sequências de caracteres. Esta distância é definida pelo número mínimo de operações necessárias para transformar uma sequência de caracteres na outra, onde as operações permitidas são a adição, a eliminação ou a substituição de um único carácter.

Limiar de semelhança	Duplicados detetados	Falsos positivos
50%	9	2
60%	8	1
70%	8	0
80%	6	0
90%	3	0
100%	2	0

Tabela 4.3: Eficácia da distância de Levenshtein na identificação de duplicação de exploits.

limiar de 50%, apesar de se identificarem 9 duplicados, registaram-se também 2 falsos positivos, indicando uma suscetibilidade a erros na classificação de *exploits* distintos como duplicados. Este número de falsos positivos diminui progressivamente com o aumento do limiar, sendo que a partir de 70%, não se observam falsos positivos.

Optou-se pelo limiar de 70% como o mais adequado para este processo devido à sua capacidade de identificar para este caso 8 duplicados sem incorrer em falsos positivos apesar de não ter identificado os 2 *exploits* restantes.

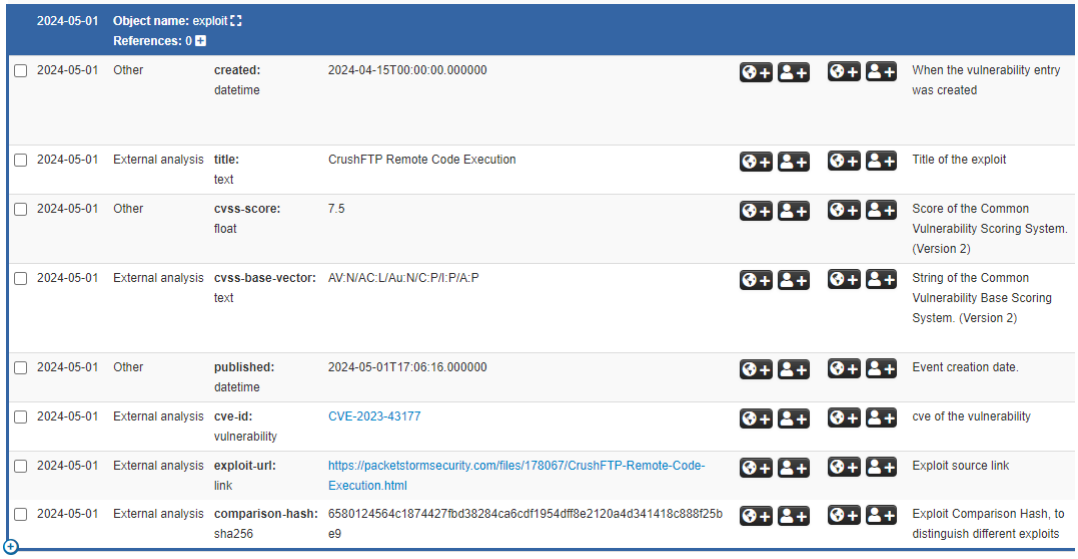


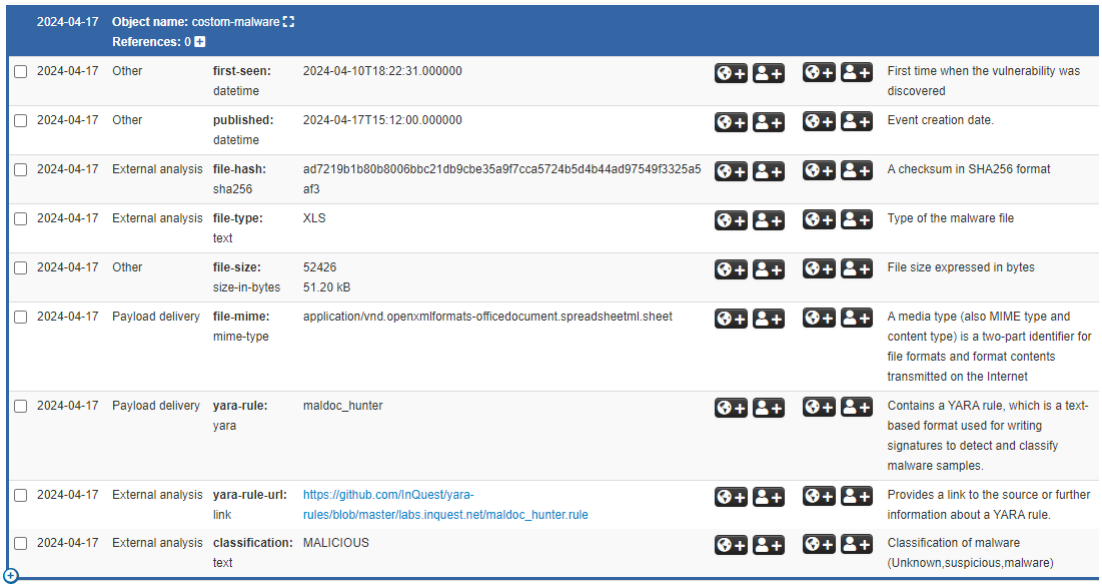
Figura 4.16: Atributos do exploit intitulado “CrushFTP Remote Code Execution”.

Para a criação de um evento de *exploit* no MISP, a lógica segue a mesma que as vulnerabilidades, mudando a *tag* de vulnerabilidade para *exploit*. O nível de severidade é calculado da mesma forma, uma vez que existe informação relativamente ao CVSS. Na Figura 4.16, podem-se observar os atributos que compõem uma ameaça de um *exploit*, de acordo com o template criado no MISP para representar esse tipo de ameaça na organização.

Processamento de malware

Para determinar se um *malware* é uma ameaça dia-zero na organização, o processo é relativamente mais simples do que as restantes componentes. Primeiramente, verifica-se se a ameaça

recolhida existe no MISP através do hash. Caso não haja conhecimento da mesma, as duas Fontes de Cruzamento (FCr) para esta componente são a VirusTotal e a IBM X-Force Exchange, utilizando também o campo do hash como identificador de comparação. Se ambas as fontes indicarem que a ameaça é desconhecida, esta é automaticamente classificada como dia-zero, caso contrário não o será.



2024-04-17	Object name: costom-malware References: 0				
2024-04-17	Other	first-seen: datetime	2024-04-10T18:22:31.000000		First time when the vulnerability was discovered
2024-04-17	Other	published: datetime	2024-04-17T15:12:00.000000		Event creation date.
2024-04-17	External analysis	file-hash: sha256	ad7219b1b80b8006bbc21db9cbe35a9f7cca5724b5d4b44ad97549f3325a5af3		A checksum in SHA256 format
2024-04-17	External analysis	file-type: text	XLS		Type of the malware file
2024-04-17	Other	file-size: size-in-bytes	52426 51.20 kB		File size expressed in bytes
2024-04-17	Payload delivery	file-mime: mime-type	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet		A media type (also MIME type and content type) is a two-part identifier for file formats and format contents transmitted on the Internet
2024-04-17	Payload delivery	yara-rule: yara	maldoc_hunter		Contains a YARA rule, which is a text-based format used for writing signatures to detect and classify malware samples.
2024-04-17	External analysis	yara-rule-url: link	https://github.com/InQuest/yara-rules/blob/master/labs.inquest.net/maldoc_hunter.rule		Provides a link to the source or further information about a YARA rule.
2024-04-17	External analysis	classification: text	MALICIOUS		Classification of malware (Unknown,suspicious,malware)

Figura 4.17: Atributos de um evento de malware.

Na criação de um evento de *malware* no MISP, o nível de severidade tem a sua classificação determinada no campo de informação ”classificação”podendo ser desconhecido (*unknown*), suspeito (*suspect*) ou malicioso (*malicious*). Na Figura 4.17, podem-se observar os atributos que compõem uma ameaça de um malware, de acordo com o template criado no MISP para representar esse tipo de ameaça na organização.

4.2.4 Motor *Threat Intel Manager*

Com a conclusão do pré-processamento do MISP, dá-se início ao pós-processamento, que inclui o motor TIM, encarregue de gerir as ameaças com base na informação que podemos extrair delas para identificar potenciais ameaças internas. Este é também responsável por administrar o registo de ativos, a fim de determinar o software em execução nos dispositivos.

Cadastro de ativos

Neste projeto, emergiu a necessidade de incorporar um processo adicional não previsto inicialmente: a recolha de dados de software dos ativos informáticos da empresa. Optou-se por utilizar duas principais fontes de informação: a plataforma CrowdStrike e o System Center Configuration Manager (SCCM).

A escolha da CrowdStrike deveu-se à sua capacidade de abranger uma gama mais ampla de sistemas operativos como *Windows*, *Linux*, entre outros, proporcionando uma visão mais extensa do

SCCM	CrowdStrike
Office 16 Click-to-Run Licensing Component	365 Apps
Microsoft Visio Viewer 2016	Visio Viewer
Microsoft Visual C++ 2010 x64 Redistributable	Visual C++
Microsoft Visual C++ 2010 x86 Redistributable	Visual C++
Java 7 Update 21	Java 7
Java 7 Update 21 (64-bit)	Java 7
Microsoft Visual C++ 2008 Redistributable - x64	Visual C++
Microsoft Visual C++ 2008 Redistributable - x86	Visual C++

Tabela 4.4: Comparação de nomes de software entre SCCM e CrowdStrike.

ambiente informático da empresa. No entanto, durante a análise dos dados recolhidos, observou-se uma limitação quanto à especificidade das informações apresentadas.

Esta discrepância nos detalhes levantou questões sobre a precisão e utilidade dos dados para fins específicos de análise e gestão de ativos. Assim, decidiu-se limitar, numa fase inicial, a recolha de dados ao SCCM, focando exclusivamente nos servidores e PCs que operam sob o sistema Windows.

Esta abordagem permitiu não só assegurar a precisão dos dados, como também facilitou a gestão e análise subsequente dos mesmos. Alguns exemplos estão representados de forma detalhada na Tabela 4.4, que ilustra as diferenças nas designações e detalhes entre as duas fontes utilizadas.

Gestão de Malware

No contexto da gestão de vulnerabilidades e *exploits*, a informação disponível ao nível do *software*, particularmente no que se refere às vulnerabilidades (CPE), necessita de ser cruzada com a informação dos ativos da empresa. Nos projetos ZeroDays, foi utilizado um algoritmo hardcoded para realizar a comparação entre o *software* de ameaças e a informação de *software* do cadastro de ativos da empresa. No entanto, esta abordagem revelou-se subótima e difícil de manter com o decorrer do tempo.

Para este projeto, a solução ideal seria possuir a informação do software já no formato CPE, o que facilitaria consideravelmente o processo de correlação de dados.

Em relação a *malware*, a confrontação é diferente, pois não existe informação específica sobre o *software* ou sistema operativo que é o alvo. Assim, em vez de cruzar informações de *software*, opta-se por extrair os IOCs de *malware* diretamente para o QRadar SIEM e para o Graylog. Esta funcionalidade é crucial para aumentar a riqueza da informação disponível, permitindo uma correlação mais eficaz das regras e, consequentemente, a deteção de potenciais ameaças.

Na Tabela 4.5 encontram-se os tipos de IOCs que são recolhidos no MISP dos eventos de

IOCs	**Fontes**
Filehashes	antimalware, antispyware
Filenames	antimalware, antispyware, endpoint, configuration manager, cloud service
Urls	firewall, antispyware, antiphishing, load balancer, CDN, cloud service, proxy, web security
Domínios	firewall, antiphishing, load balancer, CDN, proxy, web security
IPs	antimalware, firewall, antispyware, traffic management, honeypot, antiphishing, network security, load balancer, UTM, threat detection, cloud service, proxy, IDS, web security
Emails	antimalware, antispyware

Tabela 4.5: Tabela de IOCs e tipos de fontes internas.

malware. Fez-se uma investigação das fontes no QRadar para verificar que *log sources* continham este tipo de dados, de forma a poder confrontar os mesmos para a criação de regras. Os nomes das *log sources* são dados sensíveis e, portanto, foram substituídos pelas suas funções correspondentes.

Capítulo 5

Resultados e avaliação

Para analisar o desempenho do sistema, este foi executado durante o período de um mês para as componentes de *exploits* e *malware* e 15 dias para a componente de vulnerabilidades. Nesta secção, serão apresentados os resultados obtidos de cada fonte, bem como a análise do cruzamento destes dados com as fontes internas para identificar as ameaças dia-zero. Além disso, foi realizada uma avaliação detalhada das fontes identificadas e dos motores de processamento utilizados.

5.1 Componente de vulnerabilidades

Na componente de vulnerabilidades, o sistema foi configurado para realizar a recolha de dados durante 15 dias(1 de julho a 15 de julho) a partir das fontes ZDI e VulDB. Os dados recolhidos foram armazenados e preparados para análise detalhada, visando identificar e categorizar as vulnerabilidades detetadas durante este período.

5.1.1 Resultados obtidos

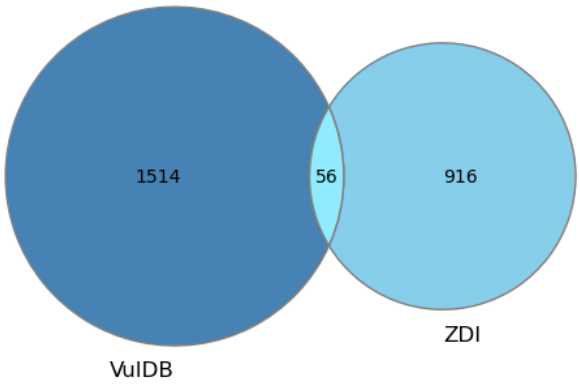


Figura 5.1: Coincidência das entradas nas fontes de informação de vulnerabilidades.

No período estabelecido, foram recolhidas um total de 2430 vulnerabilidades. No entanto, é importante mencionar que, devido a questões fora do ambiente do mestrado, a subscrição da fonte VulDB ainda não foi renovada ao dia de hoje. Assim, para esta fonte, apenas foram recolhi-

das ameaças durante 15 dias, correspondentes à subscrição paga temporária, o que foi suficiente para ser uma amostra representativa do que se pretendia analisar. A versão paga da API permite 400 pedidos por dia, em contraste com os 50 da versão gratuita. Portanto, na análise considerou-se apenas duas semanas de dados para esta fonte, pois seria necessário testar com a versão paga para obter mais informações. Em relação à ZDI, foram obtidos resultados desde o início do ano até à data da análise uma vez disponíveis, o que explica a maior quantidade de resultados comparativamente à VulDB.

Na Figura 5.1, podemos observar que o sistema recolheu 1514 vulnerabilidades da fonte VulDB e 916 da fonte ZDI, havendo uma coincidência de 56 vulnerabilidades entre ambas.

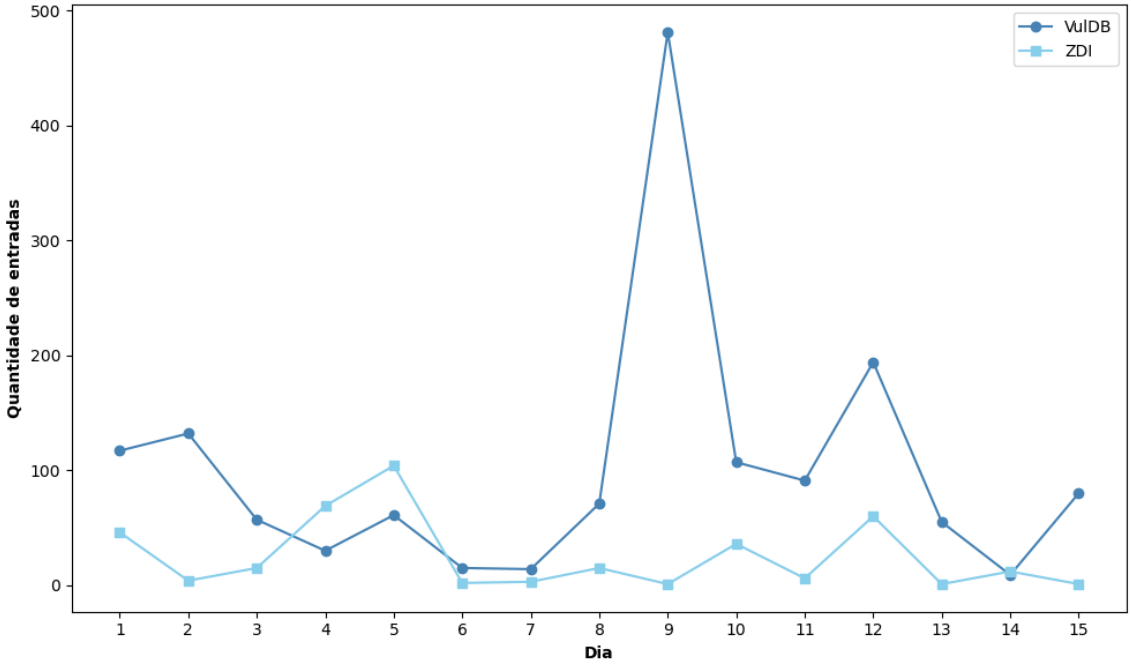


Figura 5.2: Distribuição da quantidade de entradas recolhidas ao longo de 15 dias.

A recolha de dados das duas fontes foi feita duas vezes por dia, às 8h da manhã e às 18h da tarde. Esta frequência foi escolhida para cumprir os limites estabelecidos pela API e para garantir que nenhum dado seja perdido. A recolha às 8h da manhã serve para capturar as informações que surgiram após as 18h do dia anterior, enquanto a recolha às 18h abrange os dados gerados durante o próprio dia. Relativamente à fonte ZDI, manteve-se o mesmo horário de recolha, embora esta fonte seja muito menos frequente que a VulDB. Assim, não houve necessidade de ajustar o horário de recolha.

Durante 15 dias, observou-se (Figura 5.2) que a fonte VulDB é mais frequente do que a ZDI. Notou-se uma diminuição no número de vulnerabilidades publicadas nos dias 6, 7, 13 e 14, que correspondem a fins de semana, com um aumento da frequência nos dias úteis da semana.

Na Figura 5.3, verificou-se que, do total de vulnerabilidades recolhidas de cada fonte, aproximadamente 48% das vulnerabilidades da VulDB foram dia-zero para a empresa, e aproxima-

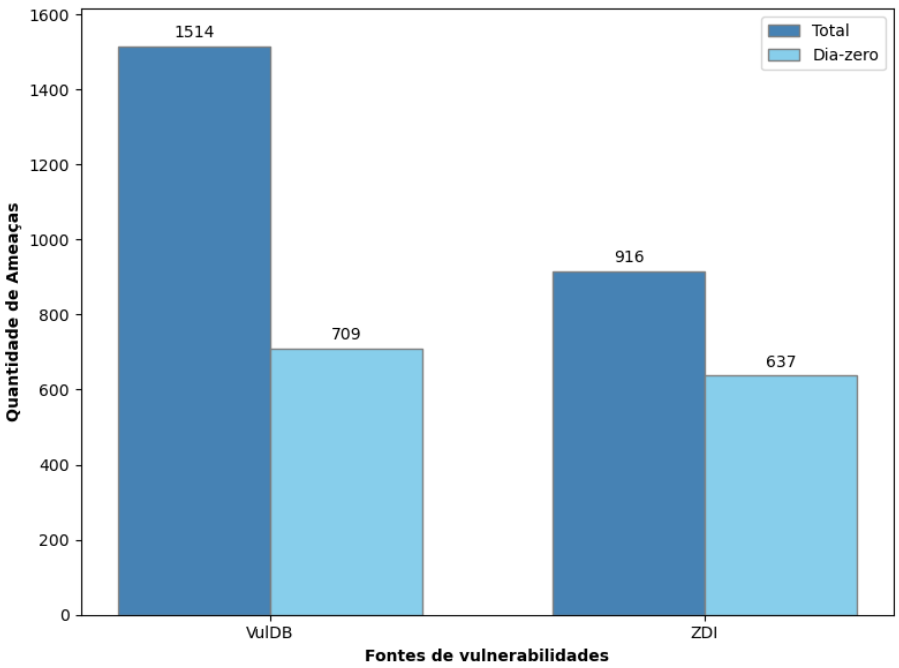


Figura 5.3: Quantidade de vulnerabilidades dia-zero recolhidas de cada fonte em relação ao total de ameaças recolhidas.

damente 70% das ameaças da ZDI foram dia-zero. Também se pode perceber que, mesmo recolhendo vulnerabilidades da ZDI desde o início do ano, muitas entradas foram identificadas como dia-zero no cruzamento com as fontes internas. Isso pode indicar que a riqueza ou a atualização dos dados das FCr não são muito eficazes.

5.1.2 Avaliação da componente de vulnerabilidades

Ao analisar os dados das fontes de informação, conclui-se que, a longo prazo, a fonte VulDB é mais rica em dados. Em um período de duas semanas, esta conseguiu recolher mais vulnerabilidades do que a ZDI em sete meses. É importante notar que a fonte ZDI contém frequentemente entradas sem CVE, o que pode levar a falsos positivos, uma vez que não é possível verificar a duplicidade dessas entradas.

Quanto ao motor de processamento de vulnerabilidades, a lógica de correlação não é tão extensiva quanto a utilizada para *exploits*, por exemplo. No entanto, mesmo após cruzar os dados com fontes internas e com o próprio MISP, foram detetadas 1084 vulnerabilidades já conhecidas previamente.

Com estes resultados, podemos concluir que a fonte VulDB é particularmente rica na partilha de vulnerabilidades, tendo sido possível recolher um grande número de vulnerabilidades dia-zero em apenas duas semanas de dados. Embora a fonte ZDI não seja tão frequente, ela contém várias ameaças que ainda não têm um CVE atribuído, o que contribui para aumentar a riqueza de vulnerabilidades dia-zero do sistema. No geral, ambas as fontes contribuíram para uma maior abrangência

no que diz respeito a vulnerabilidades dia-zero internamente.

5.2 Componente de exploits

Para avaliar a componente de exploits, esta foi executada durante um mês (entre 1 de julho e 31 de julho), e foram recolhidos os dados produzidos para que possam ser analisados.

5.2.1 Resultados obtidos

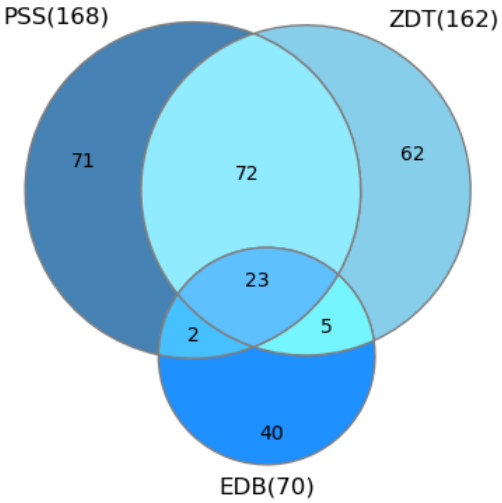


Figura 5.4: Coincidência das entradas nas fontes de informação de exploits.

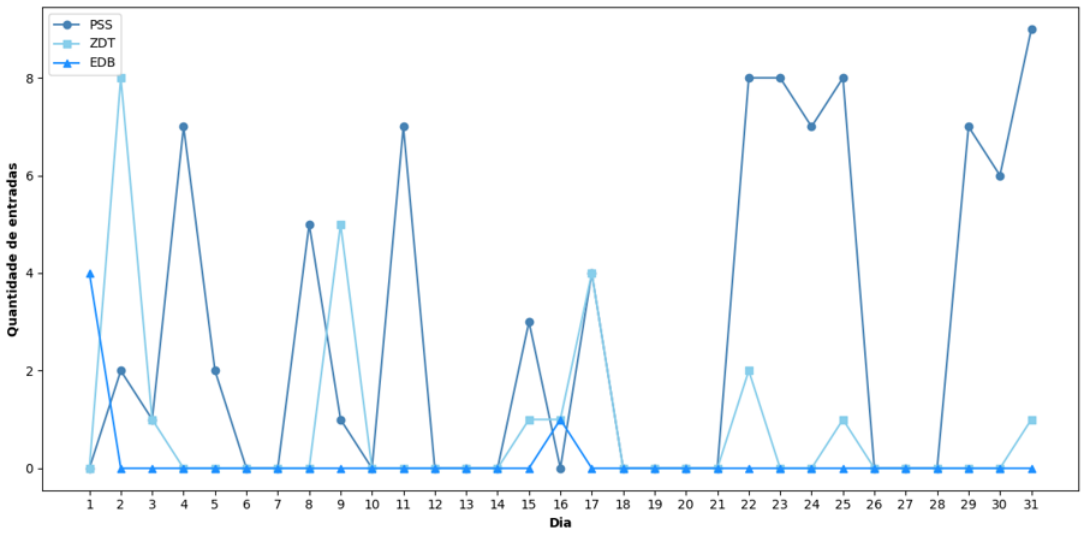


Figura 5.5: Distribuição da quantidade de entradas recolhidas ao longo de 1 mês.

Para avaliar a qualidade de cada uma das fontes, verificou-se a unicidade das entradas, conforme representado na Figura 5.4. Em relação à unicidade das fontes, a fonte PSS apresenta o

maior número de entradas únicas, seguida pela ZDT e pela EDB. A ZDT também partilha um grande número de *exploits* idênticos com a PSS.

Em seguida, analisou-se a recolha de *exploits* exclusivamente publicados durante o mês em questão (Figura 5.5), excluindo-se todas as outras entradas obtidas a partir de aproximadamente dois meses antes, disponibilizadas pela API. Observa-se que a fonte EDB raramente publicou *exploits* durante o mês e que, nos finais de semana, nenhuma das fontes publicou entradas. Após uma inspeção manual, foram verificadas 114 entradas únicas, indicando que o sistema falhou na criação do *hash* de comparação em 20 ocasiões, resultando numa taxa de falhas de aproximadamente 18%. A margem de erro, com um nível de confiança de 95%, é de aproximadamente 7%, o que significa que a taxa de falhas real na população pode variar entre aproximadamente 11% e 25%.

Na Figura 5.6, verificou-se que, do total de ameaças recolhidas de cada fonte, aproximadamente 67% das ameaças da fonte PSS foram classificadas como dia-zero, 60% das ameaças da fonte ZDT foram classificadas da mesma forma, e 71% das ameaças da fonte EDB também foram identificadas como dia-zero. Isto indica que, apesar das variações entre as fontes, a maioria das ameaças são classificadas como dia-zero, com a fonte EDB apresentando a maior proporção.

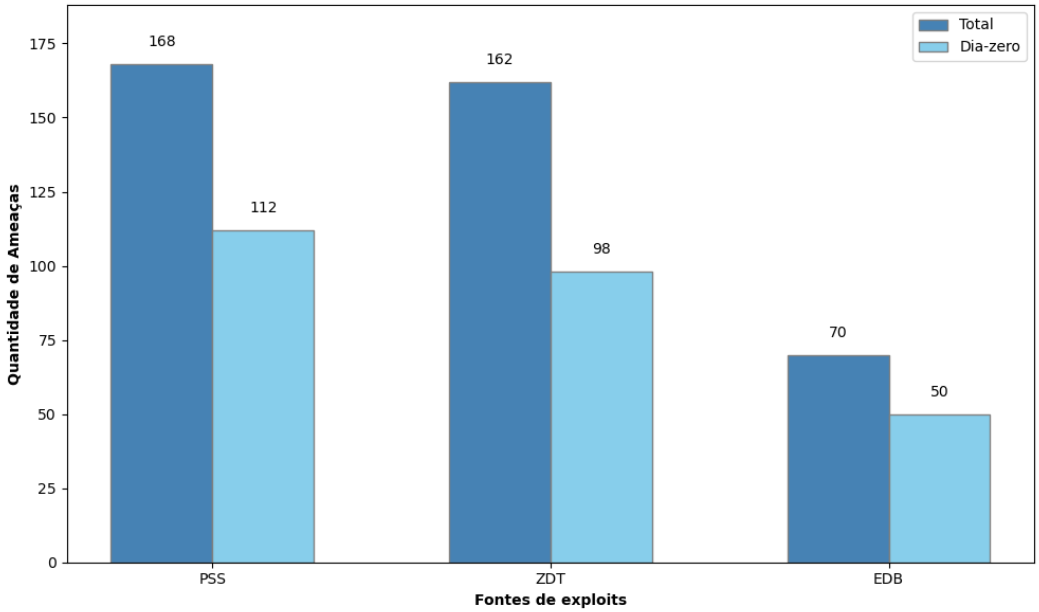


Figura 5.6: Quantidade de exploits dia-zero recolhido de cada fonte, em relação ao total de ameaças recolhidas.

5.2.2 Avaliação da componente de exploits

Dado que a informação destas fontes são recolhidos pela Vulners, o intervalo entre a publicação na fonte original e a sua inclusão na Vulners é extremamente curto. Isso minimiza a preocupação com o tempo necessário para a recolha das ameaças, mesmo com um intervalo reduzido. Conforme explicado no Capítulo 3, o sistema utiliza a Vulners como apartado para as outras fontes através da

sua API para evitar a necessidade de *web scraping*, o que se revelou uma solução excelente para o sistema, prevenindo futuros problemas que possam surgir.

Em relação ao motor de processamento de *exploits*, apresentou um desempenho positivo, não registando falsos positivos. Na prática, isso significa que todas as ameaças identificadas como dia-zero confirmaram-se como tais, com exceção das ameaças previamente mencionadas que não foram corretamente classificadas como duplicadas.

Como ilustrado na Figura 5.7, nenhum dos 114 *exploits* foram classificados como conhecidos internamente. Isso deve-se principalmente a dois fatores: a falta de informação sobre as vulnerabilidades exploradas pelos *exploits* e à reduzida quantidade de vulnerabilidades disponíveis na FCr.

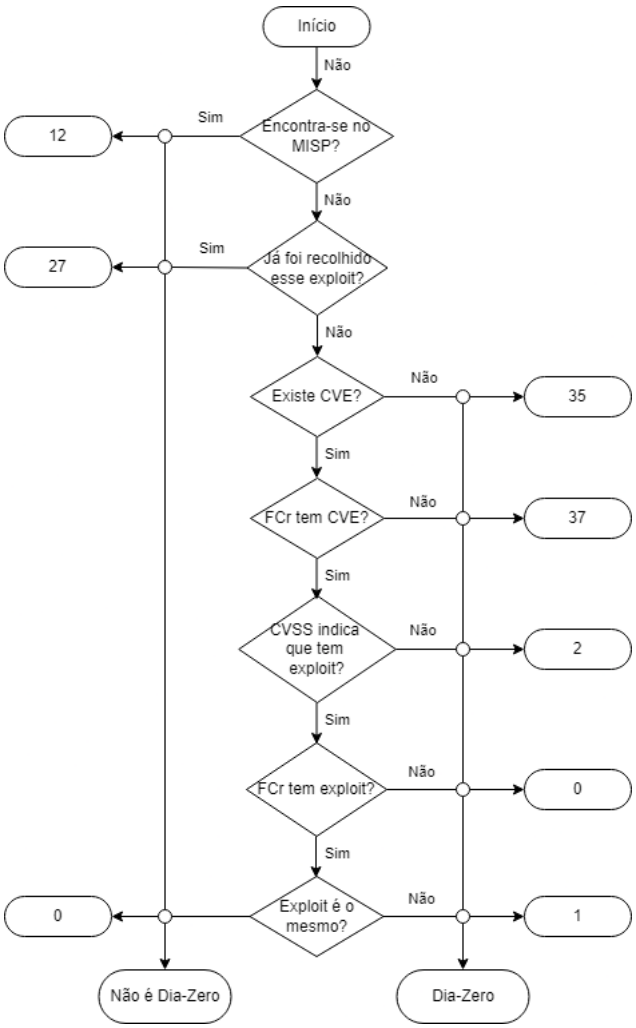


Figura 5.7: Análise do desempenho do motor de processamento na identificação de exploits previamente conhecidos.

5.3 Componente de malware

A componente de *malware* foi executada durante um mês (1 de julho a 31 de julho) e foram extraídos os dados recolhidos, para poder avaliar o seu desempenho.

5.3.1 Resultados obtidos

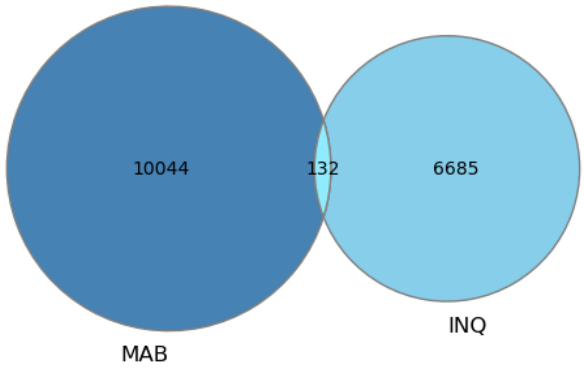


Figura 5.8: Coincidência das entradas nas fontes de informação de malware.

Durante o período estabelecido, foram recolhidas um total de 16.729 entradas como se pode observar na Figura 5.8, das quais 10.044 provenientes da fonte MAB e 6.685 da fonte INQ. Do total, 132 entradas estavam presentes em ambas as fontes.

Na fonte MAB, a solução encontrada para recolher entradas sem perder dados de forma regular foi utilizar a funcionalidade da API que permite recolher as entradas da última hora. Para manter esta regularidade e padrão de recolha, o mesmo procedimento foi aplicado à fonte INQ, utilizando a data de introdução da entrada na fonte. Assim, é possível concluir que, ao contrário de outros componentes, a frequência da recolha de informação não necessita de uma adaptação especial (por exemplo, o processo não beneficiaria de ser executado mais vezes entre as 0h e as 8h do que entre as 8h e as 16h). No entanto, devido à elevada quantidade de informação, é sempre preferível que este processo de recolha de informação ocorra com a maior frequência possível, de modo a manter o sistema atualizado com novas ameaças recém-relatadas. Portanto, o módulo de recolha para este componente é realizado de hora a hora.

Relativamente às entradas declaradas como ”dia-zero”(Figura 5.9), foram recolhidas 4.735 entradas de *malware* dia-zero na MAB e 2.277 entradas dia-zero da fonte INQ. É importante notar que, durante o mês de recolha de dados, a fonte INQ apresentou um comportamento anómalo, uma vez que as entradas estavam a demorar a ser partilhadas na própria fonte no dia específico, acabando por ser expostas apenas dias depois. Alguns problemas com a própria API ocorreram, dificultando a recolha.

Das ameaças dia-zero identificadas, muitas podem já ser conhecidas internamente, uma vez que as FCr utilizadas para este componente têm um limite de 50 pedidos de API por dia cada uma, o que faz com que, deliberadamente, entradas já conhecidas internamente sejam registadas

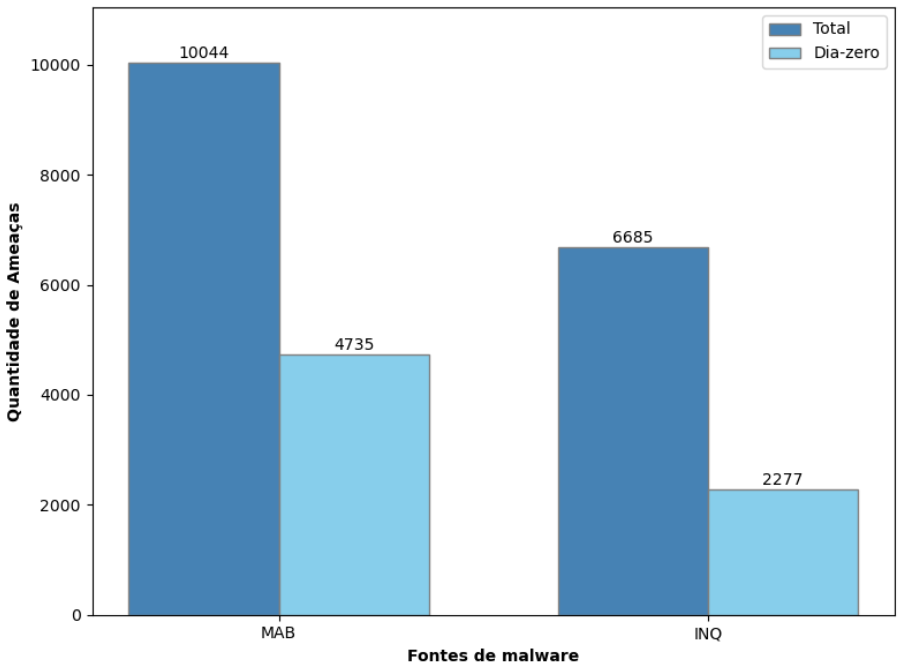


Figura 5.9: Quantidade de malware dia-zero recolhido de cada fonte, em relação ao total de malware recolhidas.

no MISP, contribuindo assim também como uma fonte de conhecimento para reduzir o número de falsos positivos no futuro.

5.3.2 Avaliação da componente de malware

Em relação à análise das fontes de malware, apesar de a MAB ter mais entradas recolhidas do que a INQ, a INQ proporciona-nos informação ao nível de IOCs. Esta riqueza de dados contribuiu para melhorar a eficiência do QRadar SIEM e do Graylog na criação de regras de correlação, facilitando a identificação de padrões suspeitos ou anómalos.

Tipo de IOC	Quantidade
File Hashes	5467
Filenames	953
Emails	184
URLs	360
IPs	106
Domains	570

Tabela 5.1: Quantidade de IOCs Recolhidos

No motor de processamento, ao cruzar os dados com as FCr e com o próprio MISP, identificou-se 5.309 ameaças já com conhecimento interno, o que equivale a aproximadamente 58% do total previamente conhecidos na fonte MAB, e 4.408 ameças na fonte INQ, com cerca de 38% já conhecidos. Isso indica que, em termos percentuais, a fonte INQ é mais eficiente na recolha de ameaças

de dia-zero.

Na Tabela 5.1, podemos verificar o número de IOCs recolhidos por tipo para os sistemas de gestão de segurança e análise de *logs*. É importante notar que esta recolha começou mais tarde do que a recolha e processamento dos dados do sistema, uma vez que a funcionalidade só ficou operacional posteriormente. Isso explica a menor quantidade de dados registados em relação ao número total de ameaças recolhidas durante o mês.

Capítulo 6

Conclusão

Este trabalho teve como objetivo desenvolver um sistema eficaz de recolha, processamento e gestão de informação que permitisse à empresa manter-se atualizada sobre ameaças (vulnerabilidades, exploits e malware) anteriormente desconhecidas internamente e que pudessem comprometer a sua segurança.

O sistema foi desenhado para recolher dados de diversas fontes: para a componente de vulnerabilidades, foram definidas a VulDB e a Zero Day Initiative, com uma calendarização de duas vezes ao dia (8 horas e 16 horas); para a componente de exploits, a Vulners forneceu informações provenientes de 0day.today, Exploit-DB e Packet Storm Security, com recolhas diárias às 18 horas devido ao limite de requisições e à baixa quantidade de *exploits* diários; por fim, para a componente de *malware*, a recolha foi executada de hora em hora, garantindo a captura de dados das fontes InquestLabs e MalwareBazaar.

A arquitetura do sistema integra três motores: o motor de recolha (responsável pela obtenção da informação), o motor de processamento (que cruza os dados com fontes internas e cria ameaças dia-zero no MISP) e o Threat Intel Manager (TIM), que agrega as informações sobre os ativos da empresa e trata da exportação dos IOCs para o QRadar SIEM e Graylog.

Este sistema cumpriu os principais requisitos estabelecidos durante a sua conceção: modularidade, simplicidade, confiabilidade, conclusividade e rapidez. Embora o sistema tenha alcançado os objetivos delineados inicialmente, os resultados apresentam limitações, nomeadamente em relação às fontes de confrontação interna, ou seja, os ativos utilizados para fazer a correspondência entre ameaças e ativos, e à falta de informação detalhada sobre o sistema afetado ou sobre como este é representado.

Os resultados foram positivos, com uma taxa de identificação de ameaças dia-zero superior a 30% em cada fonte, demonstrando que o sistema foi eficaz a aumentar a cobertura de ameaças conhecidas internamente. Além disso, um número significativo de IOCs foi exportado para o QRadar SIEM e Graylog, o que contribuirá para uma deteção e mitigação mais eficazes na proteção e resposta a incidentes.

Em conclusão, este trabalho demonstrou que o sistema desenvolvido permite à empresa manter-se sistematicamente informada, reagindo de forma atempada a ameaças que, até então, eram desconhecidas.

6.1 Trabalho futuro

No futuro o sistema pode ser aperfeiçoado com os seguintes aspetos:

- **Fontes de informação:** A eficácia do sistema está intrinsecamente ligada à quantidade e à qualidade das fontes de informação que contém. Assim, a inclusão de um número crescente de fontes de elevada qualidade será fundamental para o aperfeiçoamento contínuo do sistema. É essencial que essas fontes respeitem os requisitos estabelecidos no sistema.
- **Normalização do sistema afetado:** A categorização das informações dos ativos da empresa segundo a estrutura de CPE facilitaria a confrontação interna, especialmente na identificação e gestão de vulnerabilidades. Este processo de normalização poderá aumentar a eficiência e a precisão na avaliação dos riscos associados.
- **Priorização de CVSS:** Os métodos tradicionais de priorização de vulnerabilidades, como os fornecidos pelo CVSS, muitas vezes não consideram as interdependências complexas e os fatores contextuais que influenciam o risco real que as vulnerabilidades representam em ambientes do mundo real. O desenvolvimento de uma abordagem mais detalhada, que leve em conta estes elementos, poderia proporcionar uma estrutura mais precisa e eficaz para a priorização de vulnerabilidades, melhorando, assim, a gestão das ameaças potenciais na empresa.
- **Identificação de exploits duplicados:** O processo atual de identificação de *exploits* duplicados apresenta limitações, uma vez que ainda há casos de duplicação que não são corretamente identificados. A implementação de um algoritmo de *fuzzy hashing* que permite atribuir um nível de diferença entre o código fonte das entradas poderá melhorar significativamente este processo, permitindo uma identificação mais precisa e eficiente de *exploits* duplicados.

Bibliografia

[1] 0day.today agreement - 0day.today exploit database : vulnerability : 0day : new exploits : buy and sell private exploit : shellcode by 0day today team. https://0day.today/, 2023. (visitado a 11/18/2023).

[2] Crowdstrike: We stop breaches with ai-native cybersecurity. https://www.crowdstrike.com/en-us/, 2024. (visitado a 09/26/2024).

[3] Elk stack: Elasticsearch, kibana, beats e logstash — elastic. https://www.elastic.co/pt/elastic-stack, 2024. (visitado a 05/29/2024).

[4] What is configuration manager? - configuration manager — microsoft learn. https://learn.microsoft.com/en-us/mem/configmgr/core/understand/introduction, 2024. (visitado a 09/26/2024).

[5] 0day Today Team. 0day.today exploit database : vulnerability : 0day : new exploits : buy and sell private exploit : shellcode by 0day today team. https://0day.today/, 2023. (visitado a 11/18/2023).

[6] Lillian Ablon and Andy Bogart. Zero days, thousands of nights. *RAND Corporation, Santa Monica, CA*, 2017.

[7] Mohamed Abomhara and Geir M Køien. Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks. *Journal of Cyber Security and Mobility*, pages 65–88, 2015.

[8] abuse.ch. Malwarebazaar — about. https://bazaar.abuse.ch/about/, 2023. (visitado a 11/18/2023).

[9] Tibra Alsmadi and Nour Alqudah. A survey on malware detection techniques. In *2021 International Conference on Information Technology (ICIT)*, pages 371–376. IEEE, 2021.

[10] Fernando Alves, Ambrose Andongabo, Ilir Gashi, Pedro M Ferreira, and Alysson Bessani. Follow the blue bird: A study on threat data published on twitter. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*, pages 217–236. Springer, 2020.

[11] Crowdstrike. Hybrid analysis. https://www.hybrid-analysis.com/submissions/sandbox/files, 2023. (visitado a 11/28/2023).

[12] CSIRTG. csirtgadgets.com/collective-intelligence-framework. https://csirtgadgets.com/collective-intelligence-framework, 2023. (visitado a 10/17/2023).

[13] CXSecurity. Cxsecurity.com free security list. https://cxsecurity.com/, 2023. (visitado a 11/28/2023).

[14] cybersecurity help. Vulnerability database. https://www.cybersecurity-help.cz/, 2023. (visitado a 11/27/2023).

[15] Nathan Deguara, Junaid Arshad, Anum Paracha, and Muhammad Ajmal Azad. Threat miner-a text analysis engine for threat identification using dark web data. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 3043–3052. IEEE, 2022.

[16] CVE Details. Cve security vulnerability database. security vulnerabilities, exploits, references and more. https://www.cvedetails.com/, 2023. (visitado a 11/27/2023).

[17] Exploit-DB. Exploit database - exploits for penetration testers, researchers, and ethical hackers. https://www.exploit-db.com/, 2023. (visitado a 11/14/2023).

[18] Tomás Ferreira. Repositório da universidade de lisboa: Zerodays v2 sistema de gestão de ameaças de cibersegurança dia-zero (exploits e software malicioso dia-zero). https://repositorio.ul.pt/handle/10451/56844, 2022. (visitado a 10/07/2023).

[19] FIRST. Common vulnerability scoring system version 3.0. https://www.first.org/cvss/v3-0/, 2023. visitado a 11/14/2023.

[20] FIRST. Cvss v2 archive. https://www.first.org/cvss/v2/, 2023. (visitado a 11/14/2023).

[21] Google. "open source vulnerability database". https://osv.dev/, 2023. (visitado a 11/27/2023).

[22] Zero Day Initiative. Published — zero day initiative. https://www.zerodayinitiative.com/advisories/published/, 2023. (visitado a 11/11/2023).

[23] Julian Jang-Jaccard and Surya Nepal. A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences*, 80(5):973–993, 2014. Special Issue on Dependable and Secure Computing.

[24] Halima Kure and Shareeful Islam. Cyber threat intelligence for improving cybersecurity and risk management in critical infrastructure. *Journal of Universal Computer Science*, 25(11):1478–1502, 2019.

[25] InQuest Labs. Inquest labs - dfi - inquest.net. https://labs.inquest.net/dfi, 2023. (visitado a 11/18/2023).

[26] Heather Lawrence, Andrew Hughes, Robert Tonic, and Cliff Zou. D-miner: A framework for mining, searching, visualizing, and alerting on darknet events. In *Proceedings of the 2017 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, October 2017.

[27] Jerome Leonard. Soltra edge – thehive project. https://blog.thehive-project.org/tag/soltra-edge/, 2020. (visitado a 10/17/2023).

[28] Gary McGraw and Greg Morrisett. Attacking malicious code: A report to the infosec research council. *IEEE Software*, 17(5):33–41, 2000.

[29] Miles A McQueen, Trevor A McQueen, Wayne F Boyer, and May R Chaffin. Empirical estimates and observations of 0day vulnerabilities. In *2009 42nd Hawaii international conference on system sciences*, pages 1–12. IEEE, 2009.

[30] Héctor D Menéndez. Malware: the never-ending arm race. *Open Journal of Cybersecurity*, 1(1):1–25, 2021.

[31] MISP. Misp open source threat intelligence platform & open standards for threat information sharing. https://www.misp-project.org/, 2023. (visitado a 10/17/2023).

[32] MITRE. Cpe - common platform enumeration. https://cpe.mitre.org/, 2014. (visitado a 11/14/2023).

[33] MITRE. Crits: Collaborative research into threats. https://crits.github.io/, 2023. (visitado a 10/17/2023).

[34] MITRE. Cve. https://www.cve.org/About/Overview, 2023. (visitado a 11/14/2023).

[35] NIST. Nvd - home. https://nvd.nist.gov/, 2023. (visitado a 11/11/2023).

[36] NIST. tactics, techniques, and procedures (ttp) - glossary — csrc. https://csrc.nist.gov/glossary/term/tactics_techniques_and_procedures, 2023. (visitado a 11/14/2023).

[37] NIST. threat intelligence - glossary — csrc. https://csrc.nist.gov/glossary/term/threat_intelligence, 2023. (visitado a 11/14/2023).

[38] Bernardi Pranggono and Abdullahi Arabo. Covid-19 pandemic cybersecurity issues. *Internet Technology Letters*, 4(2):e247, 2021.

[39] Rapid7. Rapid7. https://www.rapid7.com/db/, 2023. (visitado a 11/27/2023).

[40] ANY RUN. Any.run - interactive online malware sandbox. https://any.run/, 2023. (visitado a 11/28/2023).

[41] Clemens Sauerwein, Christian Sillaber, Andrea Mussmann, and Ruth Breu. Threat intelligence sharing platforms: An exploratory study of software vendors and research perspectives. 2017.

[42] P. S Security. Packet storm. https://packetstormsecurity.com/, 2023. (visitado a 11/14/2023).

[43] Alessandra Silva, João Gondim, and Luis Javier Albuquerque, Robson e Villalba. A methodology to evaluate standards and platforms within cyber threat intelligence. *Future Internet*, 12(6):108, 2020.

[44] Symantec. istr-21-2016-en. https://docs.broadcom.com/doc/istr-21-2016-en, 2016. (visitado a 10/13/2023).

[45] Rabia Tahir. A study on malware and malware detection techniques. *International Journal of Education and Management Engineering*, 8(2):20, 2018.

[46] ThreatConnect. Threat intelligence platforms. everything you've ever wanted to know but didn't know to ask, 2018.

[47] Abi Tyas Tunggal. What is an exploit? — upguard. https://www.upguard.com/blog/exploit, 2023. (visitado a 10/14/2023).

[48] Jorge Vigário. Zerodays: sistema de gestão de ameaças de cibersegurança de diazero. https://dl.acm.org/doi/abs/10.5555/AAI29335466, 2021. (visitado a 10/07/2023).

[49] VulDB.com. Vulnerability database. https://vuldb.com/?, 2023. (visitado a 11/11/2023).

[50] I VULNERS. Vulners, inc. https://vulners.com/. (visitado a 11/28/2023).

[51] Cynthia Wagner, Alexandre Dulaunoy, Gérard Wagener, and Andras Iklody. Misp: The design and implementation of a collaborative threat intelligence sharing platform. In *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*, pages 49–56, 2016.

Apêndice A

Descrição das regras de expressão RegEx para extração de campos da ZDI

```
/(ZDI-\d+-\d+)/)
/(\d+\.\d+),/
/vector=(.+)/
/(\d{2}\/\d{2}\/\d{2})/
```

Explicação:

1. `(/(ZDI-\d+-\d+)/)` - Identifica textos que correspondem ao formato de ID típico da ZDI, que começa com ”ZDI-”seguido de números, um hífen e mais números.
2. `(/(\d+\.\d+),)/` e `(/vector=(.+))/` - A primeira expressão captura números que incluem um ponto decimal, indicando a pontuação CVSS. A segunda regex extrai a parte do vetor CVSS que segue ”vector=”em um URL.
3. `(/(\d2\/\d2\/\d2)/)` - É utilizada para identificar e formatar datas que aparecem dentro dos parágrafos ou textos. Com esta regra conseguimos captar a informação relativamente à contramedida da vulnerabilidade.

Apêndice B

Descrição dos métodos para limpeza do corpo do exploit da fonte Vulners

```
def normalize_source_data(source_data)
  normalized_data = source_data.tr('`', '')
  normalized_data = normalized_data.lines.reject { |line|
    line.strip.empty? || line.strip.start_with?('#■Date:') || line.strip == ''
  }.join
  normalized_data.gsub!(/\r\n?/, "\n")
  normalized_data.gsub!(/\t|\s+/, '■')
  normalized_data.gsub!(/[\x00-\x09\x0b\x0c\x0e-\x1f\x7f]/, '')
  normalized_data.gsub!(/\s+/, "■").strip
  normalized_data.downcase
  normalized_data = normalized_data.strip
end
```

Listing B.1: Função para normalização de dados de fonte

- `source_data.tr('`', '')` - Remove todos os acentos graves da string de entrada, simplificando o processamento de texto.
- `normalized_data.lines.reject {|line| line.strip.empty? || line.strip.start_with?('# Date:') || line.strip == ''}.join` - Filtra e remove linhas que são vazias, que começam com a string '# Date:', ou que contêm apenas aspas duplas vazias.
- `normalized_data.gsub!(/\r\n?/, "\n")` - Converte todas as quebras de linha do estilo Windows para Unix.
- `normalized_data.gsub!(/[\t]+$/, '')` - Remove espaços em branco e tabulações que aparecem no final das linhas.
- `normalized_data.gsub!(/[\x00-\x09\xb\xc\xe-\x1f\x7f]/, '')` - Elimina caracteres de controle que podem interferir no processamento do texto.
- `normalized_data.gsub!(/\s+/, ' ').strip` - Reduz todos os espaços múltiplos para um único espaço e remove espaços do início e do fim do texto.