

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Mateus Herrera Gobetti Borges

**Desenvolvimento de um sistema de geração de
alertas no contexto de ameaças cibernéticas**

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Mateus Herrera Gobetti Borges

**Desenvolvimento de um sistema de geração de alertas no
contexto de ameaças cibernéticas**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, como parte dos requi-
sitos exigidos para a obtenção título de Ba-
charel em Sistemas de Informação.

Orientador: Rodrigo Sanches Miani

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2025

Mateus Herrera Gobetti Borges

Desenvolvimento de um sistema de geração de alertas no contexto de ameaças cibernéticas

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Trabalho aprovado. Uberlândia, Brasil, 28 de abril de 2025:

Rodrigo Sanches Miani
Orientador

Maria Adriana Vidigal de Lima

Paulo Henrique Ribeiro Gabriel

Uberlândia, Brasil
2025

Dedico este trabalho aos meus pais, Maria Lúcia e Geraldo, por todo amor, exemplo e dedicação ao longo da minha jornada, e à minha tia Ezilda, pelo apoio incansável e incentivo constante.

“Não há fatos eternos, como não há verdades absolutas.”

Friedrich Nietzsche

Resumo

Este trabalho tem o objetivo de desenvolver uma funcionalidade que possibilita notificações de alertas de postagens relevantes para o contexto de cibersegurança. As postagens utilizadas para a identificação e o disparo de alertas foram coletadas e classificadas pela pesquisa de identificação de ameaças de segurança, usando mineração de dados de redes sociais e da *darkweb*. Essa pesquisa, financiada pela empresa *DataRisk* e conduzida pelo Professor Doutor Rodrigo Sanches Miani, utiliza *crawlers*, APIs e modelos de aprendizado de máquina e processamento de linguagem natural para coletar e classificar postagens relevantes ao contexto de cibersegurança. Como resultado, foi desenvolvida uma API para o gerenciamento de perfis parametrizáveis de alerta, juntamente de uma rotina que analisa cada perfil, identifica as postagens relevantes e gera notificações por e-mail. Além disso, o conteúdo das postagens encontradas, são armazenadas e disponibilizadas pela própria API de perfis.

Palavras-chave: Alertas, cibersegurança, ameaças.

Lista de ilustrações

Figura 1 – Funcionamento da coleta e processamento das postagens	19
Figura 2 – Funcionamento do processamento e visualização das postagens	20
Figura 3 – Visão geral do funcionamento do gerador de alertas	21
Figura 4 – Modelagem do banco de dados para parametrização	22
Figura 5 – Fluxograma do algoritmo para identificação e notificação de alertas . .	24
Figura 6 – Configuração da API <i>Django</i> no <i>Docker</i>	25
Figura 7 – Configuração do <i>PostgreSQL</i> no <i>Docker</i>	25
Figura 8 – Configuração do <i>NGINX</i> no <i>Docker</i>	26
Figura 9 – Requisição para criação de perfil de alerta	27
Figura 10 – Resposta bem sucedida para criação de perfil de alerta	27
Figura 11 – Resposta para atualização de data do perfil de alerta	28
Figura 12 – Requisição para busca de alertas que devem ser executados no dia atual	28
Figura 13 – Requisição feita para salvar uma postagem alertada por um perfil . . .	29
Figura 14 – Resposta para criação de postagem alertada	29
Figura 15 – E-mail recebido para alerta 1 da primeira rodada	32
Figura 16 – E-mail recebido para alerta 3 da segunda rodada	33
Figura 17 – E-mail recebido para alerta 2 da terceira rodada	33
Figura 18 – E-mail recebido para alerta 4 da quarta rodada	33
Figura 19 – Lista dos e-mails enviados para cada alerta na sétima rodada	33

Lista de tabelas

Tabela 1	–	Perfis de alerta criados para testes <i>end-to-end</i>	30
Tabela 2	–	<i>Keywords</i> e perfis associados a cada um	30
Tabela 3	–	Perfis de alertas que são executados em cada rodada	31
Tabela 4	–	Quantidades de postagens alertadas por rodada	31
Tabela 5	–	Relação de Postagens alertadas por palavra-chave e alerta	32

Lista de abreviaturas e siglas

DDoS	Ataque Distribuído de Negação de Serviço
CTI	Inteligência de Ameaças Cibernéticas
API	Interface de Programação de Aplicações
TI	Tecnologia da Informação
PLN	Processamento de Linguagem Natural
ML	Aprendizado de Máquina
RI	Recuperação de Informação
TF-IDF	<i>Term Frequency-Inverse Document Frequency</i>
TF	<i>Term Frequency</i>
IDF	<i>Inverse Term Frequency</i>
ILN	Interpretação de Linguagem Natural
GLN	Geração de Linguagem Natural
API REST	Interface de Programação de Aplicações baseada em Transferência de Estado Representacional
DRF	<i>Django REST Framework</i>
JWT	<i>JSON Web Token</i>
HTTP	Protocolo de Transferência de Hipertexto
JSON	<i>JavaScript Object Notation</i>
IoCs	Indicadores de Comprometimento
TTPs	Técnicas, Táticas e Procedimentos
TCC	Trabalho de Conclusão de Curso
SGBD	Sistema Gerenciador de Banco de Dados
IA	Inteligência Artificial

Sumário

1	INTRODUÇÃO	10
2	REVISÃO BIBLIOGRÁFICA	12
2.1	Inteligência de Ameaças Cibernéticas	12
2.2	Recuperação da Informação	12
2.3	Processamento de Linguagem Natural	13
2.4	API baseada em Transferência de Estado Representacional	14
2.5	<i>Django REST</i>	14
2.6	<i>Docker e Docker Compose</i>	15
2.7	Testes <i>end-to-end</i>	15
3	TRABALHOS RELACIONADOS	17
4	DESENVOLVIMENTO	19
4.1	Sistema de identificação de ameaças de segurança	19
4.2	Gerador de Alertas	20
4.2.1	API para parametrização	21
4.2.2	Identificação de alertas	23
4.2.3	<i>Docker e Docker-compose</i>	25
4.3	Resultados obtidos	26
4.3.1	Criação de perfil de alerta	26
4.3.2	Atualização de data de execução do perfil de alerta	27
4.3.3	Buscar perfis de alerta que devem ser executados no dia atual	28
4.3.4	Registro de <i>post</i>	29
4.3.5	Testes <i>end-to-end</i>	29
5	CONCLUSÃO	34
	REFERÊNCIAS	35

1 Introdução

Desde a popularização e a expansão da Internet para o público geral, usuários mal-intencionados buscam brechas de segurança para realização de ataques das maneiras mais diversas. As técnicas utilizadas pelos atacantes, muitas vezes, são disseminadas em fóruns, tanto na *surface* quanto na *dark web*, voltados para o tema de ataques cibernéticos, ou mesmo em redes sociais populares, como o X e o Reddit.

Um caso em que atacantes causaram prejuízo, na prática, foi o da Universidade de Utah, que após sofrer um ataque de *ransomware*, no qual foram roubadas informações pessoais de alunos, teve um prejuízo de US\$ 457,000. Outro caso que fica evidente a importância de se ter um sistema de detecção de ataques é o da Amazon que, em 2020, divulgou a tentativa de um ataque de *distributed denial of service* (DDoS) com volume de 2,3 *terabytes* por segundo ([RAHMAN; HEZAVEH; WILLIAMS, 2023](#)).

A Inteligência de Ameaças Cibernéticas (CTI) mostrou-se uma solução possível para enfrentar o aumento do número e da complexidade dos eventos de segurança. Diversas empresas buscam ferramentas para a análise e prevenção de ameaças encontradas em ambientes cibernéticos ([ABU et al., 2018](#)).

Uma maneira de evitar ou reagir da melhor maneira possível é entender como os ataques são feitos, quais ferramentas foram utilizadas e quais as vulnerabilidades os atacantes exploraram. Segundo [Sun et al. \(2023\)](#), algumas técnicas utilizadas para a prevenção de tais ameaças são, por exemplo, a identificação de vazamentos de vulnerabilidades e o levantamento de perfis de *hackers*. Há, também, um enorme número de estudos feitos para prevenção das ameaças e novas técnicas surgindo a cada ano.

Considerando esse contexto, a pesquisa, financiada pela empresa *DataRisk* e conduzida pelo Professor Doutor Rodrigo Sanches Miani, implementou códigos e modelos para coleta e classificação de postagens relacionadas a cibersegurança. A coleta acontece através de *crawlers* e chamada de serviços de redes sociais para obtenção e armazenamento de postagens de usuários que podem conter informações relevantes para o contexto de segurança na Internet. Já para a classificação, foram treinados modelos de processamento de linguagem natural que classificam os textos coletados de acordo com a presença informações relevantes para cibersegurança, de forma que, quanto mais uma postagem apresenta tais dados, maior é a sua relevância.

No contexto da pesquisa apresentada no parágrafo anterior, este trabalho tem o objetivo de implementar a parte que faz a notificação de postagens classificadas que podem ser de interesse do usuário. Para melhor atender a esse requisito, o código desenvolvido neste trabalho torna a identificação das postagens personalizável, ou seja, com base em

informações fornecidas pelo usuário, e com base nas postagens classificadas pela pesquisa, é que são identificadas as postagens que devem ser notificadas.

A coleta de informações para personalizar a notificação de postagens relevantes é chamada de perfil de alerta. Nesses perfis são armazenados os dados que devem ser considerados para identificar um *post* como relevante. Junto a isso, há a rotina que faz o cruzamento de informações entre os perfis e os textos classificados, bem como a identificação e envio dos alertas.

Para realização da criação de perfis de alertas, foi implementada uma Interface de Programação de Aplicações (API), utilizando o *framework Django REST*. Nessa API, há *endpoints* para a criação, a alteração, a recuperação e a desativação de cada alerta, além de serviços que facilitam a manipulação dos parâmetros para a implementação da lógica de geração de alertas.

Implementado em *Python*, o código responsável pela identificação e notificação de textos relevantes, faz a busca pelos perfis de alertas ativos, bem como as postagens classificados desde o seu último processamento, e aplica a lógica para coleta de quais *posts* devem ser notificados. Cada perfil, também possui endereços de e-mail que devem ser notificados sobre a quantidade de postagens encontradas. A visualização do conteúdo dos alertas, por sua vez, acontece por meio de uma interface *web*.

Este trabalho é composto por cinco capítulos, incluindo esta introdução. O segundo capítulo apresenta a revisão bibliográfica, com os principais conceitos e ferramentas utilizadas. O Capítulo 3 trata dos trabalhos relacionados, destacando pesquisas que foram usadas como inspiração. O quarto capítulo descreve o desenvolvimento, demonstrando as funcionalidades implementadas, suas características e os testes realizados. Por fim, o quinto capítulo conclui com a análise dos resultados obtidos e sugestões para futuras contribuições relacionadas a este trabalho.

2 Revisão Bibliográfica

Este capítulo apresenta as ferramentas e os conceitos utilizados para o desenvolvimento do código de identificação e notificação de *posts* relevantes e da API que faz o gerenciamento dos perfis de alertas e dos resultados encontrados.

2.1 Inteligência de Ameaças Cibernéticas

A *Cyber Threat Intelligence* pode ser usada como meio de proteção contra ciberataques, devido ao seu caráter antecipatório (RAHMAN; HEZAVEH; WILLIAMS, 2023). Segundo Conti, Dargahi e Dehghantanha (2018), essa abordagem surge como uma forma de conhecer métodos de ataques cibernéticos e formular estratégias para prevenção de tais ameaças. Já para McMillan (2013), trata-se de um conhecimento empírico composto por diversos fatores que indicam a existência de perigos emergentes à cibersegurança, auxiliando na tomada de decisão.

Considerando as mudanças e as disseminações de táticas de ciberataques, o CTI pode ser um mecanismo eficiente para equipes de Tecnologia da Informação (TI) planejarem meios de lidar com tais ataques (RAHMAN; HEZAVEH; WILLIAMS, 2023). Os principais conceitos utilizados para pesquisas na área são o Processamento de Linguagem Natural (PLN) e Aprendizado de Máquina (ML), com isso, possibilitando a automação da obtenção e da classificação de informações relevantes (LIAO et al., 2016; LEE et al., 2017; ZHU; DUMITRAS, 2018).

Tounsi e Rais (2018) separa o CTI em algumas subdivisões, inteligência estratégica de ameaças, inteligência operacional de ameaças, inteligência tática de ameaças e inteligência técnica de ameaças. Algumas vantagens do uso do CTI são a elaboração de defesas previamente identificadas, o conhecimento de perfis de atacantes, o compartilhamento de informações e a realização de pesquisas em cibersegurança (RAHMAN; HEZAVEH; WILLIAMS, 2023).

2.2 Recuperação da Informação

Segundo Baeza-Yates e Ribeiro-Neto (2013), a Recuperação da Informação (RI) estuda meios de representar, de armazenar, de organizar e de acessar informações. A necessidade de se ter mecanismos de acesso rápido à informação se torna cada vez mais necessário conforme os humanos produzem e armazenam conhecimento e dados que são usados para os mais diversos fins. Antes da computação, já existiam sistemas que faziam

o que os mecanismos de busca, como Google, DuckDuckGo, Bing, entre outros, fazem, porém, eram manuais e lentos, como o exemplo das bibliotecas.

Dos meios de RI, o mais comum para indexação de textos é o *Term Frequency-Inverse Document Frequency* (TF-IDF) que proporciona, além de uma análise estatística, muitos meios para adaptação dos sistemas de RI, conforme a necessidade das buscas (AIZAWA, 2003). Essa métrica, proporciona, por exemplo, uma maior precisão e retomada de dados relevantes a uma busca, o que também facilita a classificação de textos de acordo com modelos matemáticos.

Para a formação de um TF-IDF, é necessário a multiplicação de outros dois termos o *Term Frequency* (TF), que pode ser entendido como a quantidade de vezes que um termo aparece em um texto e o *Inverse Document Frequency* (IDF), que é a importância de um certo termo em uma base de documentos considerando a quantidade de ocorrências observadas (AIZAWA, 2003). Com o TF se analisa a quantidade que um certo termo aparece em um texto, já o IDF mostra qual a importância desses termos conforme a quantidade de vezes que ele aparece na base de busca, ou seja, quanto mais um termos está presente em uma base de documentos, mais comum ele é, portanto, menor é o seu score, entretanto, quanto menos ele aparece, maior é a sua relevância, pois é mais excludente.

2.3 Processamento de Linguagem Natural

O Processamento de Linguagem Natural pretende entender e produzir linguagem humana, via processamento computacional, podendo ser dividido em duas subáreas: Interpretação de Linguagem Natural (ILN) e Geração de Linguagem Natural (GLN) (CASELI; NUNES, 2023). A relação PLN com RI é bem comum, uma vez que processam linguagem natural em forma de texto (CASELI; NUNES, 2023), é interessante evidenciar a importância dessa relação para CTI, uma vez que não somente é necessário buscar possíveis ameaças em uma base de informações conhecidas, quanto interpretar e classificar textos que são possíveis candidatos.

No contexto de CTI, nota-se que o essencial de PLN a se considerar é o ILN, tendo em vista o seu caráter mais interpretativo e classificatório do que está sendo entrado como informação, que, no caso, seriam textos com possíveis ameaças cibernéticas. O PLN, tal como mecanismos de ML, são importantes para o processo de extração de informações relevantes para CTI (RAHMAN; HEZAVEH; WILLIAMS, 2023). O Processamento de Linguagem Natural possibilita a análise semântica de postagens na Internet de forma rápida e automática, assim é possível identificar textos relevantes para determinados assuntos em uma pesquisa.

2.4 API baseada em Transferência de Estado Representacional

Uma API se trata da comunicação entre dois *softwares*, tal como as interfaces gráficas são o meio para a comunicação entre humanos e máquinas. As API's são maneiras práticas e simples para comunicação entre sistemas distribuídos. Mais especificamente, as *Web API's* são um importante recurso para servir dados através da rede (BIEHL, 2015).

Conforme escrito por Soni e Ranga (2019), uma Interface de Programação de Aplicações baseada em Transferência de Estado Representacional (API REST), é um tipo API que utiliza transferência de recursos através de requisições dos clientes, que recebem as respostas do servidor, ou seja, todos os dados necessário para determinado serviço devem ser passados na chamada da API, significando que não há armazenamento de estados.

Para a implementação da API que foi desenvolvida, foram utilizados os conceitos de API REST, pois esses facilitam a organização e a distribuição de recursos, tanto para integração com a interface *web*, que faz parte da pesquisa que originou este trabalho, quanto para a utilização dos dados pelo código que é responsável pela lógica de geração e armazenamento dos alertas.

2.5 Django REST

O *Django REST Framework* (DRF) é uma biblioteca da linguagem *Python* que, junto à biblioteca *web* Django, permite a criação de API's REST, de maneira rápida e fácil, além disso, ainda há todas as possibilidades de manipulação e tratamento de dados fornecidos pela própria linguagem, como o *Pandas*.

Segundo Rubio e Rubio (2017), algumas vantagens em se utilizar o DRF, e não apenas *Python/Django* para a criação de serviços REST, são:

- A biblioteca cria telas administrativas, com descrições dos serviços e exemplo de requisições e respostas, além de mostrar a modelagem dos dados no próprio navegador.
- Fornece algumas das principais técnicas de autenticação, como OAuth2, *Basic* (utilizada neste trabalho), *JSON Web Token* (JWT), entre outras.
- Possui serializadores para relacionamentos complexos de dados, que atendem perfeitamente as possibilidades de modelagem do *Django* puro.

Tendo em vista o exposto, a parametrização dos perfis e o gerenciamento dos dados para a funcionalidade desenvolvida neste trabalho de conclusão de curso utilizará o DRF,

dessa maneira tornando o processo mais rápido, eficaz e escalável. Essa escolha também facilitará o entendimento do código da API, bem como sua modelagem dos dados.

2.6 *Docker e Docker Compose*

Docker é uma plataforma que permite rodar soluções de maneira simples e rápida, facilitando integrações entre partes de um mesmo sistema. Antes de sua popularização, a preparação para um ambiente de *software*, envolviam diversas tarefas, como, gerenciamento de dependências, máquinas virtuais, gerenciamento de sistemas operacionais, entre outros. Com essa plataforma, tais atividades ficam centralizadas e reutilizáveis, por meio de apenas um arquivo de configuração, o *Dockerfile* (MIELL; SAYERS, 2019).

São muitos os casos em que, para tornar uma aplicação operacional, é necessário a criação de múltiplos *Dockers* que se comunicam entre si. O *Docker Compose* é a ferramenta que possibilita o gerenciamento dos *Dockerfiles* envolvidos em um sistema, por meio de configurações e de comandos pré-escritos em um arquivo *YAML*. Com essa abordagem é possível, também, fazer com que os serviços se comuniquem entre si (JANGLA, 2018).

No contexto deste trabalho há, por exemplo, três serviços, 'api', 'nginx' e 'postgres', respectivamente, o código *Django REST*, o *webserver* e o banco de dados. Cada um desses possui seu arquivo de configuração, um *Dockerfile*, com uma imagem disponibilizada no *DockerHub*, plataforma oficial do *Docker*, e as instruções iniciais para preparação dos ambientes. Por exemplo, para configurar um ambiente simple para criação de um *Nginx*, pode-se usar um *Dockerfile* que executará os primeiros comandos, para remover arquivos padrões do *Nginx*, copiar os que serão utilizados para o *container* e expor as portas necessárias.

Já no arquivo *docker-compose*, é feita a definição e a conexão entre os serviços, com base nos *Dockerfiles*, e são descritas configurações como nomes internos para serviços, quais portas estão expostas em quais serviços, volumes de persistência de dados, o comando para iniciar o serviço, o *Dockerfile* utilizado para iniciar o *container*, etc. Essa correlação resulta em uma solução isolada e funcional das partes necessárias para o funcionamento de um sistema, tornando a sua disponibilização rápida e estável, sendo feita de forma automática pelo *Docker*.

2.7 Testes *end-to-end*

Testes *end-to-end* são realizados pela execução total de um sistema, do início ao fim, de diversas maneiras diferentes com o objetivo de validar as funcionalidades que devem ser atendidas por ele (PRESSMAN; MAXIM, 2021). Segundo Figueiredo et al. (2023), esse tipo de teste fornece um entendimento geral de como é o funcionamento total

do sistema e, ao analisar os resultados, é possível verificar o estado de desenvolvimento, podendo ser uma ferramenta eficiente para decidir o futuro do sistema.

Neste trabalho esses testes foram utilizados e serão apresentados ao final da seção de desenvolvimento. Eles foram utilizados para verificar o cumprimento dos objetivos propostos por esse TCC, com diferentes parâmetros e com dados reais classificados pela ferramenta desenvolvida na pesquisa de identificação de ameaças cibernéticas.

3 Trabalhos Relacionados

Este capítulo apresenta os estudos e as soluções comerciais que, de alguma maneira, inspiram e se relacionam com este, principalmente, no âmbito de Inteligência de Ameaças Cibernéticas.

Este trabalho tem como ponto de partida o trabalho realizado pela pesquisa de identificação de ameaças de segurança cibernética, usando mineração de dados de redes sociais e da darkweb. Os *crawlers* utilizados para coleta de textos dos foruns da Internet, estão descritos no trabalho de [Labaki \(2023\)](#). Já uma descrição mais detalhada dos modelos de PLN e ML utilizados e outros mecanismos como o *TF-IDF* e a estrutura das postagens classificadas, foram descritas no trabalho de [Filho \(2024\)](#). Além disso, ainda no escopo dessa pesquisa, [Reis \(2024\)](#) faz a análise das postagens já classificadas quanto a sua temática, utilizando técnicas de ML não supervisionado.

No artigo de [Rahman, Hezaveh e Williams \(2023\)](#), é apresentado um estudo sobre maneiras de implementar mecanismos de CTI. Na pesquisa, além do levantamento de 84 trabalhos relevantes para o tema, são apresentados pontos importantes para a realização de implementações de modelos de CTI, incluindo a identificação de fontes para obtenção de dados, técnicas para a classificação de textos e um *pipeline* de seis passos para extração de informações. O artigo também destaca 11 propósitos para tais extrações, com ênfase na classificação de texto, na extração de Indicadores de Comprometimento (IoCs) e Técnicas, Táticas e Procedimentos (TTPs).

Este Trabalho de Conclusão de Curso (TCC), se propõe a implementar um dos 11 propósitos citados no artigo desta seção. O item em questão é *Cyberthreat alert generation*, que, segundo [Rahman, Hezaveh e Williams \(2023\)](#), tem a função notificar as partes interessadas sobre indicadores das informações que estão sendo analisadas. Para realizar esta tarefa, este TCC parte dos dados coletados e pré-classificados pela pesquisa de identificação de ameaças de segurança usando mineração de dados de redes sociais e *darkweb*.

Já na pesquisa conduzida por [Sceller et al. \(2017\)](#), em que foi realizada a coleta, classificação e absorção de potenciais postagens que possuíam ameaças cibernéticas no *Twitter*, é possível constatar uma forma eficaz de CTI. Com a coleta dos dados, via a API da rede social e com o auxílio de uma coleção de palavras relevantes para o tema de cibersegurança, o algoritmo de detecção de eventos classificou os *tweets* coletados, armazenou-os e os exibiu para os analistas.

Na pesquisa apresentada nessa seção, não há uma funcionalidade como a desenvolvida neste TCC, uma vez que não é apresentado um sistema para geração de alertas

dos *tweets* já classificados, mas o que acontece é a estruturação e apresentação dos dados para os usuários. Contudo, seria possível implementar soluções semelhantes entre o *Event Detection* e a exibição para os analistas, funcionando como um notificador extra de postagens que possuem características que podem ser interessantes aos usuários.

A plataforma *ZeroFox* ¹, que fornece monitoramento em tempo real na *Dark Web*, por meio de Inteligência Artificial (IA), possui uma funcionalidade semelhante à implementada neste trabalho, se diferenciando, justamente nos pontos citados, os alertas dessa plataforma são gerados no momento da classificação e identificados com auxílio de IA. Já a definição previa dos parâmetros desejados nos alertas são o ponto de convergência.

O *CloudSEK XVigil* ² é uma plataforma, voltada para organizações, que analisa discursos em redes sociais e na *surface* e *dark web* com potenciais ameaças cibernéticas. O sistema também oferece um gerador de alertas, de acordo com a severidade da ameaça (baixa, média e alta) e com os grupos de análise (palavra mencionada, vazamento de dados, etc.).

Assim como no *ZeroFox*, o *CloudSEK* se diferencia deste trabalho pela utilização de IA para identificação de alertas e pelas notificações no momento do armazenamento dos textos classificados. Como semelhança, novamente a definição previa dos parâmetros para serem gerados um alerta.

¹ <<https://www.zerofox.com/products/dark-web-monitoring/>>

² <<https://www.cloudsek.com/cyber-threats/hacker-discussions-monitoring>>

4 Desenvolvimento

Para o desenvolvimento da funcionalidade de geração de alertas, primeiro está descrito o contexto de desenvolvimento, bem como os diagramas que demonstram o funcionamento do sistema e, por fim, os resultados da implementação.

4.1 Sistema de identificação de ameaças de segurança

A pesquisa de identificação de ameaças de segurança foi construído com o objeto de fornecer análises, por meio de mecanismos de recuperação de informação e modelos de PLN e ML, e também ser capaz de gerar alertas ao identificar o processamento de *posts* relevantes para o contexto de segurança.

A primeira parte da pesquisa acontece com a coleta, pré-processamento e análise dos dados, conforme demonstrado na Figura 1. Nesse ponto, foram coletadas diversas postagens em fóruns da *Dark web* e em redes sociais, por meio de consultas a APIs oficiais e de *crawlers* que retiravam diretamente das páginas *web*. Em seguida, com os dados armazenados, em um *PostgreSQL*, foram feitos o pré-processamento e o ranqueamento, com base em seus conteúdos.

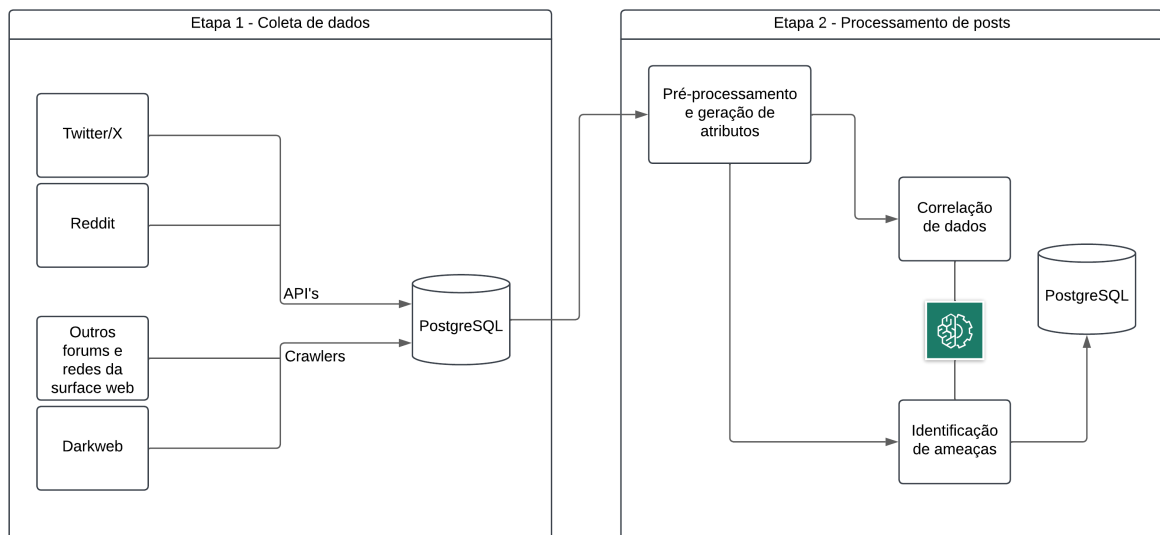


Figura 1 – Funcionamento da coleta e processamento das postagens (Fonte: Do autor)

Na Etapa 2, conforme descrito por Filho (2024), como resultado da classificação dos *posts*, é gerada a coluna "probabilidade". Tal coluna é um valor entre 0 e 1 que indica

a relevância daquele texto quanto a existência de conteúdo que contém algum tipo de ameaça. Já a coluna "relevância", indica a interpretação dessa probabilidade.

Com os dados coletados, processados e classificados, é possível implementar visualizações e tratativas com os resultados obtidos, como demonstrado na Figura 2. Nessa etapa, foram feitos *dashboards* e o gerador de alertas, que estarão disponíveis na interface *web*. Este site consumirá a API de gerenciamento de perfis de alertas, que são usados para notificar o usuário de acordo com o que foi configurado.

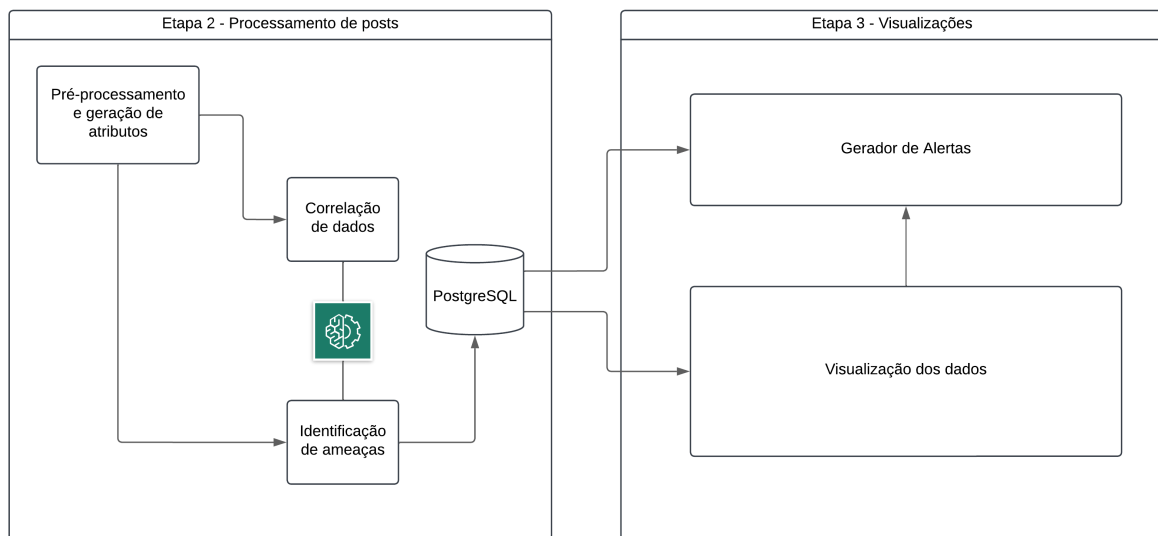


Figura 2 – Funcionamento do processamento e visualização das postagens (Fonte: Do autor)

4.2 Gerador de Alertas

Para o desenvolvimento do gerador de alertas, foram feitos um *script* para identificar as postagens relevantes para cada perfil ativo, notificando o usuário por e-mail e salvando as postagens, que poderão ser visualizadas na interface *web*, e uma API que fará o gerenciamento dos perfis de alertas, isto é, o usuário pode escolher se gostaria de ser notificado de postagens que possuem determinadas palavras, coletados de determinadas fontes e relevância acima de certa porcentagem, que faz paralelo a coluna "probabilidade" descrita na Seção 4.1. Uma visão geral da funcionalidade de alertas pode ser vista na Figura 3.

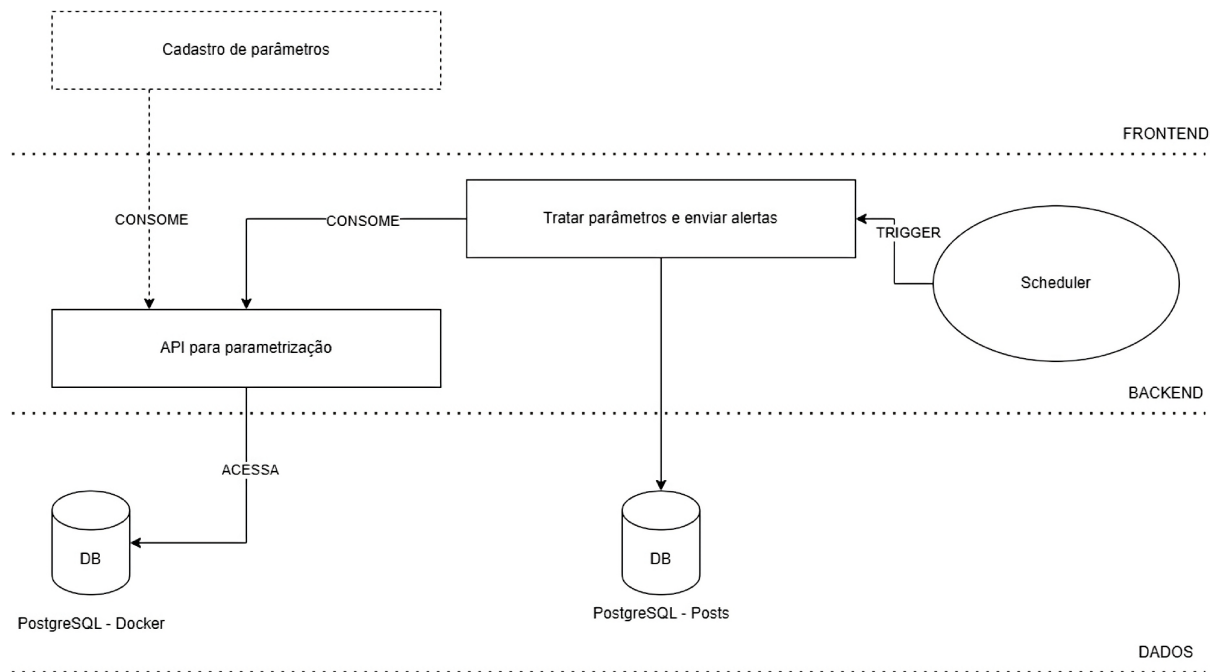


Figura 3 – Visão geral do funcionamento do gerador de alertas (Fonte: Do autor)

4.2.1 API para parametrização

A API para parametrização é o gerenciador de dados, por meio dela são criados os perfis de alertas, que parametrizam quais postagens devem ser notificadas ao usuário que os criou. Para o seu funcionamento, é necessário a estruturação e a persistência dos dados dos perfis gerados. Para isso, a modelagem dos dados foi desenhada conforme a Figura 4. Algo importante de se notar é que as tabelas foram geradas a partir dos *models* do *Django*, então o estado exato do banco de dados é transparente para o desenvolvedor, embora possa ser acessado via Sistema Gerenciador de Banco de Dados (SGBD).

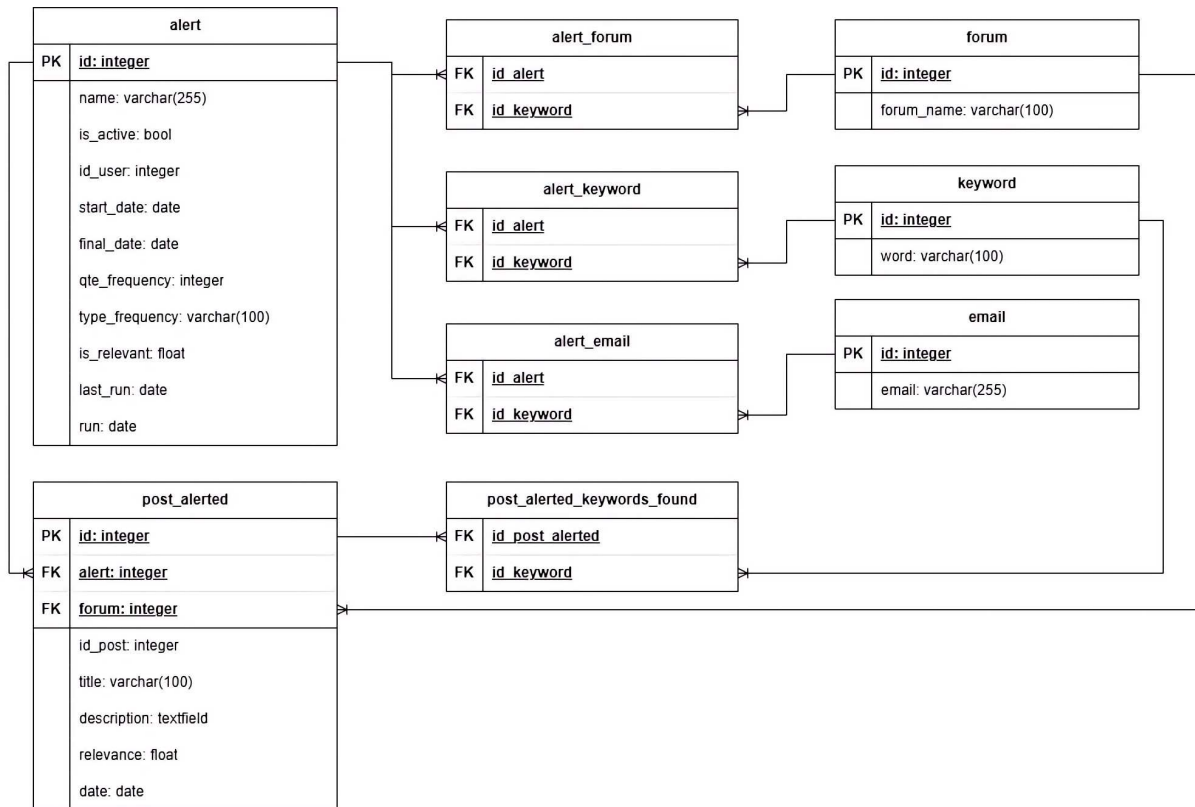


Figura 4 – Modelagem do banco de dados para parametrização (Fonte: Do autor)

O funcionamento dos perfis acontece da seguinte maneira. O usuário, ao criar um perfil de alerta, escolhe quais são as palavras relevantes (tabela *keywords*), a partir de qual relevância (coluna *is_relevant* da tabela *alert*) e oriundo de quais fóruns (tabela *forum*) as postagens devem ser notificadas. Além disso, devem ser informados, o período em que o perfil é válido, com qual frequência ele deve notificar e para quais e-mails devem ser enviados as notificações.

Além dos perfis, os *posts* que foram identificados como relevantes também são salvos neste banco de dados, com o objetivo de torná-los disponíveis para a visualização *web*. Essa necessidade acontece por um motivo, os e-mails enviados, apenas notificam a quantidade de postagens relevantes para determinado alerta, ou seja, por motivos de segurança, os seus conteúdos não são enviados, por consequência disso, essas postagens precisam ser recuperáveis pelo *front*, com as informações de quais palavras relevantes aparecem em cada *post*, de qual *forum* ele foi retirado e qual é a seu percentual de classificação, além do seu título, da sua descrição, da sua data de postagem e de qual alerta ele foi notificado.

Tendo em vista os dois principais núcleos da API de parametrização, aqui estão listados os principais *endpoints*, juntamente com seus métodos de chamada, que acontecem com autenticação *Basic*, implementados na aplicação *Django REST*, com uma breve

descrição para cada item:

- **POST alert/**: Chamada para criação de perfil de alerta, devem ser passadas as informações do nome do alerta, qual o id do usuário que está criando, data de início, data de fim, quantidade da frequência e tipo da frequência (por exemplo, diariamente, quantidade é '1' e tipo é 'dia'), a partir de qual relevância as postagens devem ser relevantes, quais fóruns devem ser analisados, quais palavras devem estar nas postagens e para quais e-mails as notificações devem ser enviadas.
- **GET alert/**: Busca de todos os perfis de alertas criados.
- **GET alert/id-alert/update/run/**: Chamada para atualização de um alerta, identificado na URL da requisição, de quanto foi feita a última análise e de quando será feita a próxima análise de postagens. Caso seja o último dia, será desativado.
- **GET alert/user/id-user/**: Busca todos os alertas criados por um usuário, identificado na URL da requisição.
- **GET alert/active/user/id-user/**: Busca todos os alertas ativos criados por um usuário, identificado na URL da requisição.
- **GET alert/run/today/**: Busca todos os perfis de alerta que devem ser analisados no dia atual.
- **POST post_alerted/**: Chamada para criação de postagem relevante para determinado alerta, devem ser enviados na requisição o *id* do *post* (no banco de *posts* classificado), o título, a descrição (seu conteúdo), a sua relevância, a data de postagem, o *id* do alerta que gerou a notificação, o fórum de origem e as *keywords* encontradas nele.
- **GET post_alerted/**: Busca todas as postagens notificados.
- **GET post_alerted/alert/id-alert/**: Busca todas as postagens notificadas do alerta identificado na URL da requisição.

4.2.2 Identificação de alertas

Para a geração das notificações, foi implementado um *script Python* que faz a leitura dos perfis ativos que devem ser executados no dia atual, e também busca as novas postagens classificadas desde a última análise feita para cada perfil. Com essas informações, as postagens são filtradas, considerando, primeiro, os fóruns e a relevância dos *posts* e por fim, a verificação da presença de palavras-chave. Com as informações filtradas, as postagens são salvas, por meio da API de parametrização, e o e-mail de quantos alertas foram gerados é enviado.

Para que a frequência de alertas seja mantida adequadamente às periodicidades dos perfis criados, o *script* rodará diariamente, ou seja, ele estará agendado no servidor para executar uma vez por dia. A fluxograma do código está demonstrado na Figura 5.

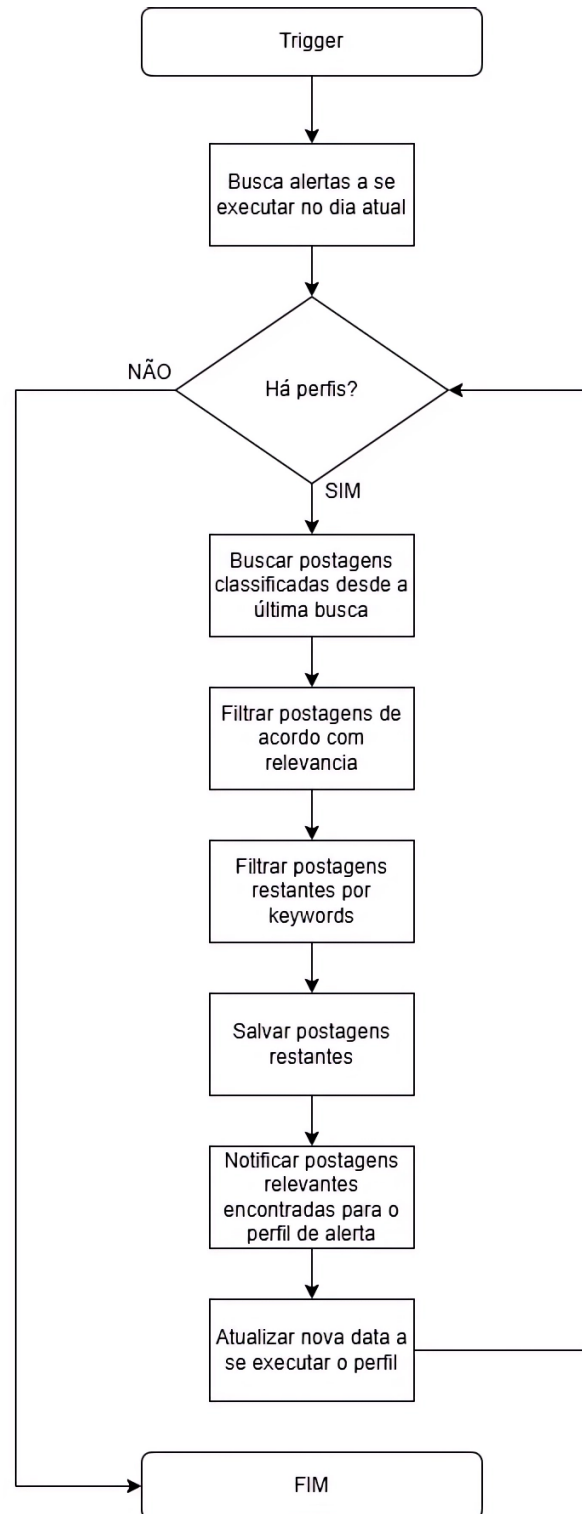


Figura 5 – Fluxograma do algoritmo para identificação e notificação de alertas (Fonte: Do autor)

4.2.3 Docker e Docker-compose

Para a disponibilização da API de parametrização, foi utilizado *Docker*, garantindo maior versatilidade na execução dos serviços em diferentes ambientes. Além disso, essa abordagem assegura que todo o sistema de geração de alertas permaneça isolado, integrado e funcione de maneira mais estruturada como uma funcionalidade adicional à interface *web*.

O *Docker compose* deste trabalho é composto por três *services*, demonstrados separadamente na Figuras 6, 7 e 8, apesar de estarem em um mesmo arquivo, e os *Dockerfiles* necessários. Sendo os serviços, "api", Figura 6, o código em *Django REST*, "db", Figura 7, o *PostgreSQL* exclusivo para o gerador de alertas e "nginx", Figura 8, o *webservices* utilizado para disponibilização da API.

```
1  api:
2    build:
3      context: ./api
4      dockerfile: Dockerfile
5    container_name: cti_alerts_api
6    restart: always
7    env_file:
8      - ./api/.env
9    command: gunicorn cti.wsgi:application --bind 0.0.0.0:8000
10   volumes:
11     - ./api:/api
12   expose:
13     - "8000"
```

Figura 6 – Configuração da API *Django* no *Docker* (Fonte: Do autor)

```
1  db:
2    image: postgres:15
3    container_name: cti_alerts_db
4    restart: always
5    env_file:
6      - ./api/.env
7    volumes:
8      - postgres_data:/var/lib/postgresql/data
9      - ./db/init-scripts:/docker-entrypoint-initdb.d
10   # expose:
11   ports:
12     - "5432:5432"
```

Figura 7 – Configuração do *PostgreSQL* no *Docker* (Fonte: Do autor)

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It displays a Docker Compose configuration for an NGINX service. The configuration includes build context, Dockerfile, container name, restart policy, environment file, volumes, ports, and a dependency on the 'api' service.

```
1  nginx:
2    build:
3      context: ./nginx
4      dockerfile: Dockerfile
5    container_name: cti_alerts_nginx
6    restart: always
7    env_file:
8      - ./api/.env
9    volumes:
10     - ./nginx/nginx.conf:/etc/nginx/nginx.conf
11     - ./api/staticfiles:/api/staticfiles
12     - ./api/media:/api/media
13    ports:
14     - "8877:8877"
15    depends_on:
16     - api
```

Figura 8 – Configuração do *NGINX* no *Docker* (Fonte: Do autor)

4.3 Resultados obtidos

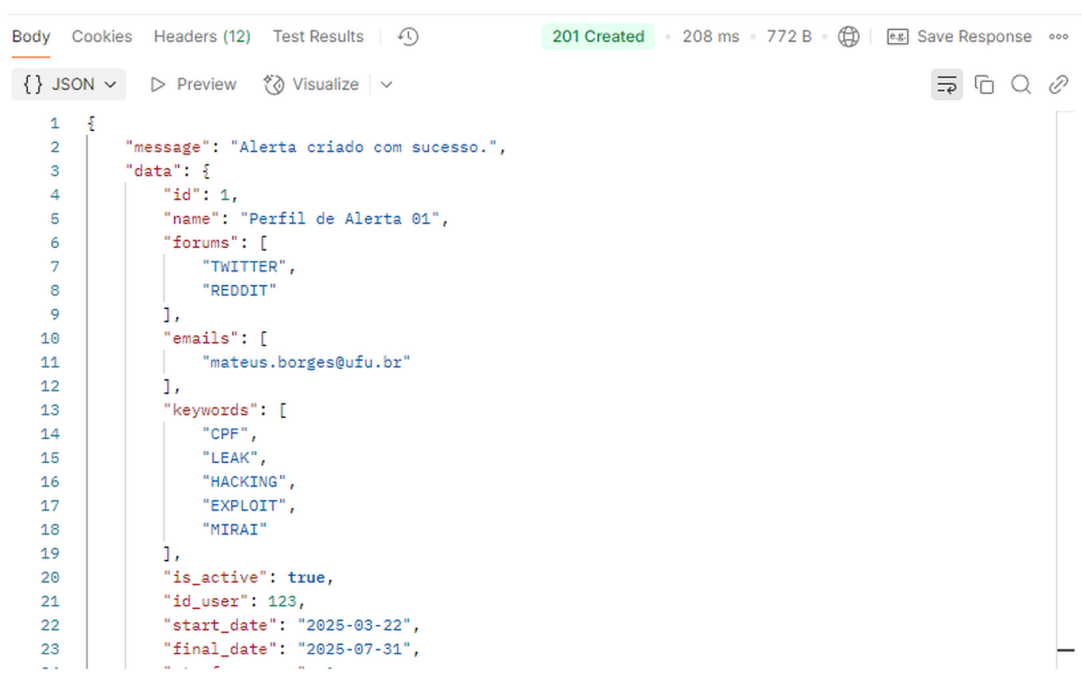
Para validar os resultados da criação da API, utilizou-se a ferramenta *Postman*, que permite realizar chamadas HTTP com diversos métodos, autenticações e outras funcionalidades. Nessa seção estão listados e demonstrados os principais *endpoints* para o gerador de alertas, bem como os testes realizados.

4.3.1 Criação de perfil de alerta

Para criação de um perfil de alertas, um *JavaScript Object Notation (JSON)* deve ser fornecido, nesse objeto devem ser passadas as informações necessárias para criação de um perfil, como pode ser visto na Figura 9. A resposta, como exposto na Figura 10, retorna com '201 - CREATED' e com a confirmação das informações do perfil criado.

```
1  {
2      "name": "Perfil de Alerta 01",
3      "id_user": 123,
4      "start_date": "2025-03-22",
5      "final_date": "2025-07-31",
6      "qte_frequency": 1,
7      "type_frequency": "days",
8      "is_relevant": 0.6,
9      "forums": ["twitter", "reddit"],
10     "emails": ["mateus.borges@ufu.br"],
11     "keywords": ["cpf", "leak", "hacking", "exploit", "mirai"]
12 }
```

Figura 9 – Requisição para criação de perfil de alerta (Fonte: Do autor)



The screenshot shows a REST client interface with the following details:

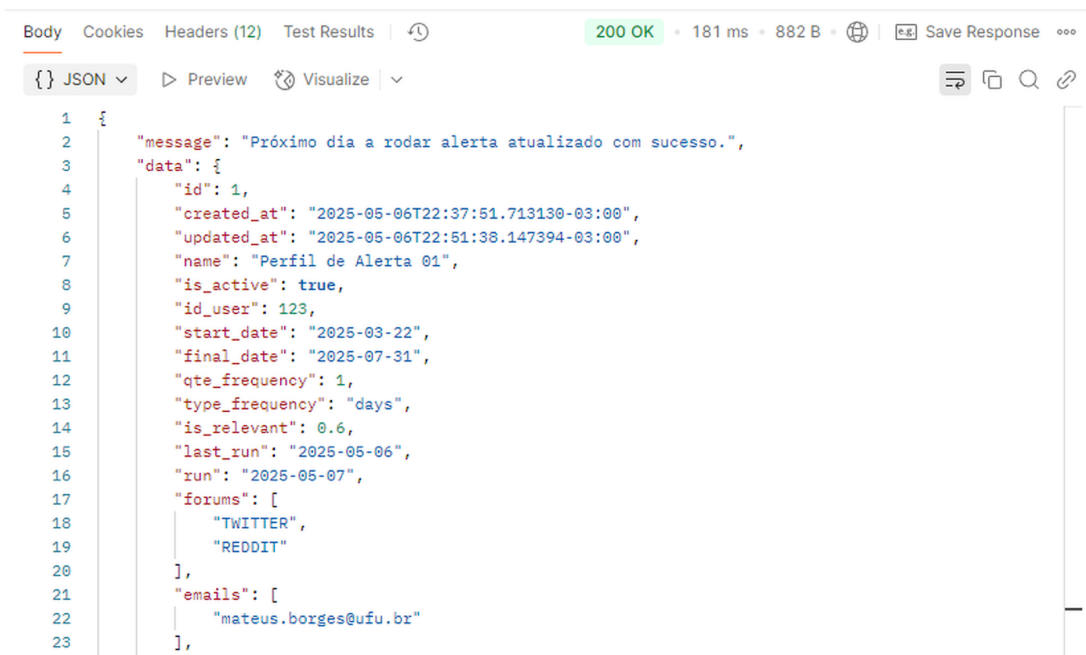
- Status:** 201 Created
- Time:** 208 ms
- Size:** 772 B
- Response Type:** JSON
- Response Body:**

```
1  {
2      "message": "Alerta criado com sucesso.",
3      "data": {
4          "id": 1,
5          "name": "Perfil de Alerta 01",
6          "forums": [
7              "TWITTER",
8              "REDDIT"
9          ],
10         "emails": [
11             "mateus.borges@ufu.br"
12         ],
13         "keywords": [
14             "CPF",
15             "LEAK",
16             "HACKING",
17             "EXPLOIT",
18             "MIRAI"
19         ],
20         "is_active": true,
21         "id_user": 123,
22         "start_date": "2025-03-22",
23         "final_date": "2025-07-31",
24     }
25 }
```

Figura 10 – Resposta bem sucedida para criação de perfil de alerta (Fonte: Do autor)

4.3.2 Atualização de data de execução do perfil de alerta

Para atualizar a data, de acordo com a próxima execução do perfil, há um *endpoint* onde as regras são tratadas e processadas de acordo com a frequência cadastrada, essa atualização pode ser vista na Figura 11.

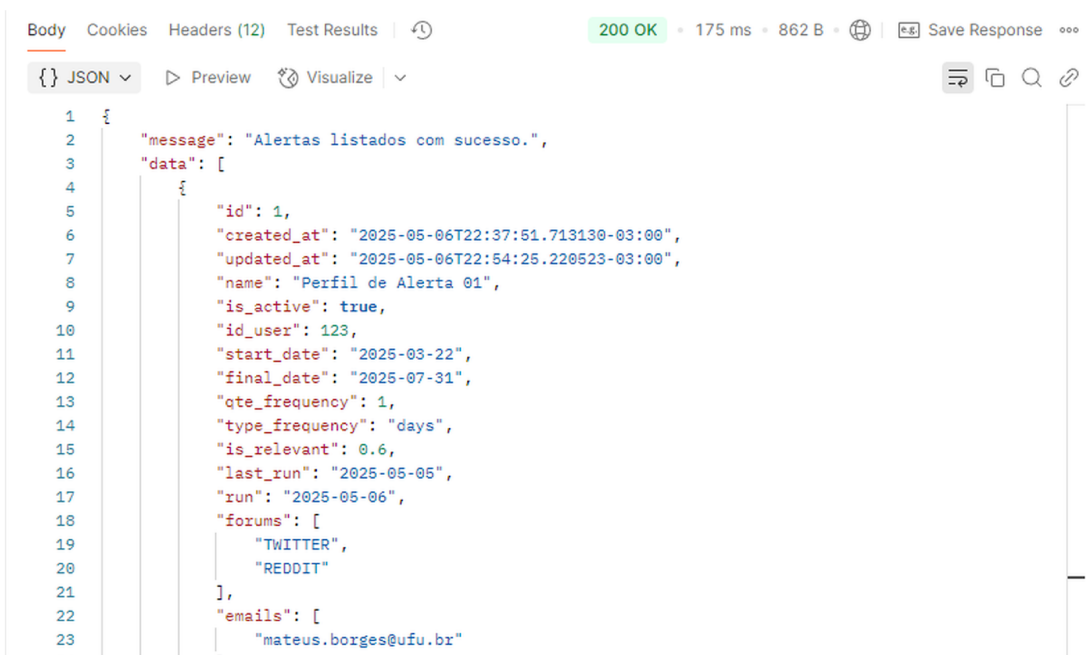


```
1 {
2   "message": "Próximo dia a rodar alerta atualizado com sucesso.",
3   "data": {
4     "id": 1,
5     "created_at": "2025-05-06T22:37:51.713130-03:00",
6     "updated_at": "2025-05-06T22:51:38.147394-03:00",
7     "name": "Perfil de Alerta 01",
8     "is_active": true,
9     "id_user": 123,
10    "start_date": "2025-03-22",
11    "final_date": "2025-07-31",
12    "qte_frequency": 1,
13    "type_frequency": "days",
14    "is_relevant": 0.6,
15    "last_run": "2025-05-06",
16    "run": "2025-05-07",
17    "forums": [
18      "TWITTER",
19      "REDDIT"
20    ],
21    "emails": [
22      "mateus.borges@ufu.br"
23    ],
24  },
25 }
```

Figura 11 – Resposta para atualização de data do perfil de alerta (Fonte: Do autor)

4.3.3 Buscar perfis de alerta que devem ser executados no dia atual

Como exposto na Figura 12, para facilitar e otimizar o *script* de notificação dos alertas, há uma chamada de API que retorna os perfis de alerta que devem ser executados no dia atual.



```
1 {
2   "message": "Alertas listados com sucesso.",
3   "data": [
4     {
5       "id": 1,
6       "created_at": "2025-05-06T22:37:51.713130-03:00",
7       "updated_at": "2025-05-06T22:54:25.220523-03:00",
8       "name": "Perfil de Alerta 01",
9       "is_active": true,
10      "id_user": 123,
11      "start_date": "2025-03-22",
12      "final_date": "2025-07-31",
13      "qte_frequency": 1,
14      "type_frequency": "days",
15      "is_relevant": 0.6,
16      "last_run": "2025-05-05",
17      "run": "2025-05-06",
18      "forums": [
19        "TWITTER",
20        "REDDIT"
21      ],
22      "emails": [
23        "mateus.borges@ufu.br"
24      ],
25    },
26  ],
27 }
```

Figura 12 – Requisição para busca de alertas que devem ser executados no dia atual (Fonte: Do autor)

4.3.4 Registro de *post*

Para o registro de um *post* notificado pelo gerador de alertas, também foi desenvolvido um *endpoint*, conforme pode ser visto a sua requisição na Figura 13 e a resposta na Figura 14.

```
1  {
2      "id_post": 1234,
3      "title": "Tweet 1",
4      "description": "Hacking CPF.",
5      "relevance": 0.9,
6      "date": "2025-05-06",
7      "alert": 1,
8      "forum": "twitter",
9      "keywords_found": ["Hacking", "CPF"]
10 }
11
```

Figura 13 – Requisição feita para salvar uma postagem alertada por um perfil (Fonte: Do autor)



The screenshot shows a REST client interface with the following details:

- Body: Cookies Headers (12) Test Results 201 Created 182 ms 607 B Save Response
- JSON Preview Visualize
- Response body (lines 1-17):

```
1  {
2      "message": "Post alertado criado com sucesso.",
3      "data": {
4          "id": 1,
5          "alert": 1,
6          "keywords_found": [
7              "CPF",
8              "HACKING"
9          ],
10         "forum": "TWITTER",
11         "relevance": 0.9,
12         "id_post": 1234,
13         "title": "Tweet 1",
14         "description": "Hacking CPF.",
15         "date": "2025-05-06"
16     }
17 }
```

Figura 14 – Resposta para criação de postagem alertada (Fonte: Do autor)

4.3.5 Testes *end-to-end*

Para a realização dos testes *end-to-end*, foram utilizadas postagens coletadas e classificadas pela pesquisa de identificação de ameaças cibernéticas. No total foram sele-

cionadas 10.087 postagens retiradas da rede social X e classificadas pelo identificador de ameaças. Esses textos foram escolhidos, por proporcionar certa variedade de percentuais de relevâncias, com isso pode-se analisar os resultados do identificador de alertas com postagens de conteúdos diversos.

No momento em que os testes foram realizados, as postagens não estão sendo coletadas ou classificadas, pelo modelo de identificação de ameaças, então, foi necessário simular um ambiente de produção, isto é, tabelas iguais às usadas e os dados, previamente classificados, sendo inseridos manualmente nas tabelas ao longo dos teste. Dessa forma, para o gerador de alertas, as postagens classificadas estão sendo registradas de maneira semelhante ao que seriam as inserções em tempo real.

Para que as postagens comecem a ser analisadas foram criados os perfis de alerta apresentados na Tabela 1, enquanto a Tabela 2 exibe as *keywords* associadas a cada um deles. Ao todo, foram executadas 7 rodadas de testes, sendo inseridos, no início de cada uma, 1441 postagens sem repetições, retiradas das 10.087 escolhidas inicialmente. A Tabela 3 demonstra quais perfis foram executados em cada rodada, considerando suas respectivas frequências.

ID	Nome	Relevância	Frequência
17	Alerta 1	0,4	Cada 1 dia
18	Alerta 2	0,8	Cada 2 dias
19	Alerta 3	0,6	Cada 1 dia
20	Alerta 4	0,9	Cada 3 dias

Tabela 1 – Perfis de alerta criados para testes *end-to-end* (Fonte: Do autor)

<i>Keyword</i>	IDs de <i>posts</i>
<i>leak</i>	17, 18
<i>hack</i>	17, 18
<i>discord</i>	17, 18
<i>backdoor</i>	17, 18
<i>vulnerability</i>	17, 18
<i>ddos</i>	19, 20
<i>phising</i>	19, 20
<i>cybersecurity</i>	19, 20
<i>vulnerable</i>	19, 20
<i>password</i>	19, 20

Tabela 2 – *Keywords* e perfis associados a cada um (Fonte: Do autor)

Rodada	IDs que executam
Rodada 1	17, 18, 19, 20
Rodada 2	17, 19
Rodada 3	17, 18, 19
Rodada 4	17, 19, 20
Rodada 5	17, 18, 19
Rodada 6	17, 19
Rodada 7	17, 18, 19, 20

Tabela 3 – Perfis de alertas que são executados em cada rodada (Fonte: Do autor)

Com a realização dos testes, foi possível analisar, na Tabela 4, a quantidade de postagens atribuídas a cada alerta ao longo das rodadas. O alerta 1, programado para rodar diariamente e com menor nível de relevância, gerou o maior número de notificações, enquanto o alerta 4, de maior relevância, foi o que resultou em menos postagens alertadas. Essa relação evidencia a possibilidade de ajustar os parâmetros para obter resultados mais ou menos restritos. Além disso, a análise das palavras-chave mais relevantes permite otimizar os perfis para aprimorar os alertas.

Rodada	Alerta 1	Alerta 2	Alerta 3	Alerta 4
R1	421	136	127	18
R2	330	-	140	-
R3	360	296	116	-
R4	367	-	137	47
R5	409	381	146	-
R6	415	-	121	-
R7	443	329	145	83
Total	2745	1142	932	148

Tabela 4 – Quantidades de postagens alertadas por rodada (Fonte: Do autor)

Outra análise, apresentada na Tabela 5, evidencia a relação entre as palavras-chave e os alertas. Como os alertas 1 e 2 compartilham as mesmas *keywords*, assim como os alertas 3 e 4, é possível observar a variação na quantidade de notificações geradas apenas pela alteração do nível de relevância. A palavra '*LEAK*', por exemplo, com uma relevância de 0,4, ocupa a primeira posição com 1106 alertas. No entanto, em um perfil mais restrito, sua posição cai para terceiro lugar, com apenas 313 postagens alertadas, menos da metade do outro alerta.

Alerta	Palavra-chave	Qtd. de alertas
Alerta 1	<i>LEAK</i>	1106
Alerta 1	<i>VULNERABILITY</i>	717
Alerta 1	<i>HACK</i>	476
Alerta 1	<i>BACKDOOR</i>	384
Alerta 1	<i>DISCORD</i>	62
Alerta 2	<i>BACKDOOR</i>	381
Alerta 2	<i>HACK</i>	354
Alerta 2	<i>LEAK</i>	313
Alerta 2	<i>VULNERABILITY</i>	66
Alerta 2	<i>DISCORD</i>	28
Alerta 3	<i>PHISHING</i>	490
Alerta 3	<i>CYBERSECURITY</i>	309
Alerta 3	<i>DDOS</i>	57
Alerta 3	<i>PASSWORD</i>	55
Alerta 3	<i>VULNERABLE</i>	21
Alerta 4	<i>CYBERSECURITY</i>	95
Alerta 4	<i>PASSWORD</i>	24
Alerta 4	<i>PHISHING</i>	20
Alerta 4	<i>DDOS</i>	8
Alerta 4	<i>VULNERABLE</i>	1

Tabela 5 – Relação de Postagens alertadas por palavra-chave e alerta (Fonte: Do autor)

Por fim, como mostrado nas Figuras 15, 16, 17, 18 e 19, o envio das notificações por e-mail acontece, fornecendo informações sobre a quantidade de postagens alertadas para cada perfil, além de destacar a percentual de maior relevância encontrado.

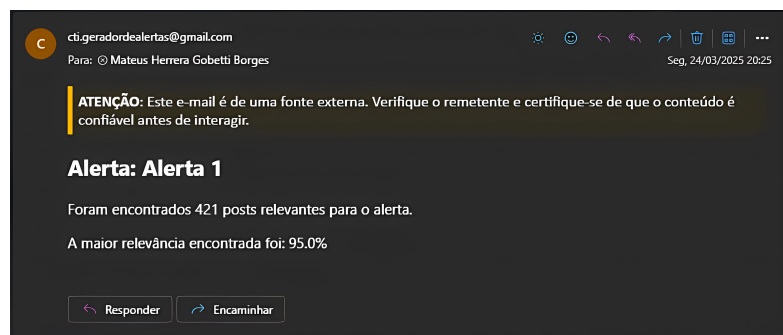


Figura 15 – E-mail recebido para alerta 1 da primeira rodada (Fonte: Do autor)

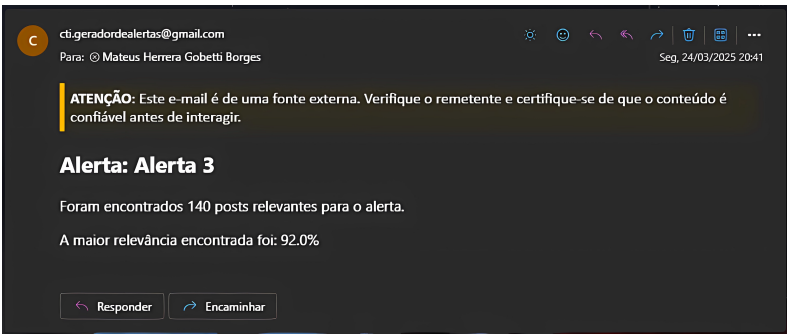


Figura 16 – E-mail recebido para alerta 3 da segunda rodada (Fonte: Do autor)

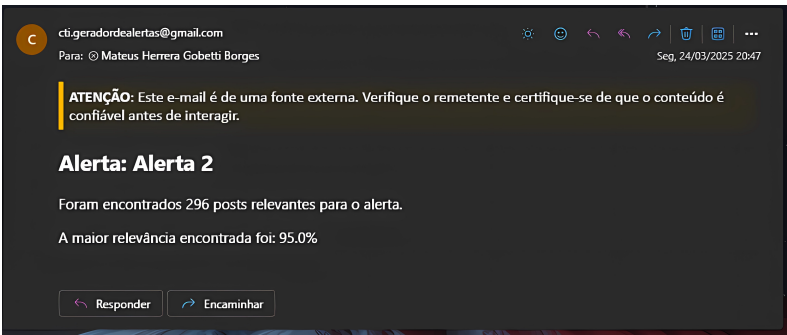


Figura 17 – E-mail recebido para alerta 2 da terceira rodada (Fonte: Do autor)

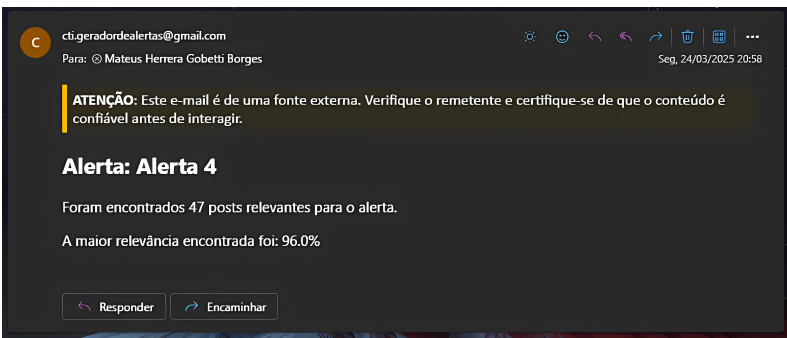


Figura 18 – E-mail recebido para alerta 1 da quarta rodada (Fonte: Do autor)

De	Assunto	Recebido(a) ▾
▾ Hoje		
☐ cti.geradordealertas@gmail.com	Posts Relevant...	21:21
☐ cti.geradordealertas@gmail.com	Posts Relevant...	21:20
☐ cti.geradordealertas@gmail.com	Posts Relevant...	21:19
☐ cti.geradordealertas@gmail.com	Posts Relevant...	21:18
> Último mês		
> 2024		

Figura 19 – Lista dos e-mails enviados para cada alerta na sétima rodada (Fonte: Do autor)

5 Conclusão

Dentro do contexto de cibersegurança, a velocidade de se conhecer os atacantes e de se obter informações que circulam na *Internet* sobre o assunto, é crucial para a melhor tratativa de possíveis incidentes. Considerando isso, um sistema de coleta e classificação de postagens quanto à sua relevância para o tema e, aliado a isso, a capacidade de alertar os usuários sobre textos classificados que são de seus interesses, torna-se uma ferramenta muito útil para a prevenção de incidentes cibernéticos.

O objetivo deste trabalho foi a implementação de uma funcionalidade para notificação de postagens classificadas e que atendam a parâmetros fornecidos pelos usuários. Os resultados obtidos mostram que esse objetivo foi alcançado, uma vez que, com o sistema funcionando em sua totalidade, o gerador de alertas é capaz de notificar, via e-mail, e salvar as postagens para futuras visualizações.

Para evolução da qualidade das notificações, sugere-se a implementação de mecanismos de buscas, nos textos classificados, mais robustas, por exemplo, a aplicação de *TF-IDF* para recuperação de postagens classificadas, que atendam aos parâmetros, e ainda ofereçam um ranque de *posts*, dado a busca. Além disso, analisar a viabilidade de utilização de serviços para armazenamento e transmissão de dados em tempos real, como o *Apache Kafka*, para tornar a funcionalidade mais ativa em relação aos usuários.

Referências

- ABU, M. S.; SELAMAT, S. R.; ARIFFIN, A.; YUSOF, R. Cyber threat intelligence—issue and challenges. **Indonesian Journal of Electrical Engineering and Computer Science**, v. 10, n. 1, p. 371–379, 2018. Disponível em: <<https://doi.org/10.11591/ijeecs.v10.i1.pp371-379>>. Citado na página 10.
- AIZAWA, A. An information-theoretic perspective of tf-idf measures. **Information Processing & Management**, Elsevier, v. 39, n. 1, p. 45–65, 2003. Disponível em: <[https://doi.org/10.1016/S0306-4573\(02\)00021-3](https://doi.org/10.1016/S0306-4573(02)00021-3)>. Citado na página 13.
- BAEZA-YATES, R.; RIBEIRO-NETO, B. **Recuperação de Informação-: Conceitos e Tecnologia das Máquinas de Busca**. [S.l.]: Bookman Editora, 2013. Citado na página 12.
- BIEHL, M. **API architecture**. [S.l.]: API-University Press, 2015. v. 2. Citado na página 14.
- CASELI, H. M.; NUNES, M. G. V. (Ed.). **Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português**. [S.l.]: BPLN, 2023. <<https://brasileiraspln.com/livro-pln>>. ISBN 978-65-00-80693-9. Citado na página 13.
- CONTI, M.; DARGAHI, T.; DEGHANTANHA, A. **Cyber threat intelligence: challenges and opportunities**. [S.l.]: Springer, 2018. Disponível em: <<https://doi.org/10.1007/978-3-319-73951-9>>. Citado na página 12.
- FIGUEIREDO, R. M. de C.; FERNANDES, R. E. S.; FIGUEIREDO, R. M. de C.; ARAUJO, C. M.; MENEZES, J. S. de. Um estudo comparativo de características das ferramentas de automação de teste end-to-end: Cypress vs qa wolf vs testcafé. In: **SBC. Encontro Nacional de Computação dos Institutos Federais (ENCompIF)**. [S.l.], 2023. p. 53–60. Disponível em: <<https://doi.org/10.5753/encompif.2023.230852>>. Citado na página 15.
- FILHO, S. A. d. J. Identificação de posts maliciosos na dark web utilizando aprendizado de máquina supervisionado. Universidade Federal de Uberlândia, 2024. Citado 2 vezes nas páginas 17 e 19.
- JANGLA, K. Docker compose. In: **Accelerating Development Velocity Using Docker: Docker Across Microservices**. [S.l.]: Springer, 2018. p. 77–98. Disponível em: <<https://doi.org/10.1007/978-1-4842-3936-0>>. Citado na página 15.
- LABAKI, A. d. P. Uso de web scrapers para coleta de dados em fóruns especializados de segurança. Universidade Federal de Uberlândia, 2023. Citado na página 17.
- LEE, K.-C.; HSIEH, C.-H.; WEI, L.-J.; MAO, C.-H.; DAI, J.-H.; KUANG, Y.-T. Sec-Buzzer: cyber security emerging topic mining with open threat intelligence retrieval and timeline event annotation. **Soft Computing**, Springer, v. 21, p. 2883–2896, 2017. Disponível em: <<https://doi.org/10.1007/s00500-016-2265-0>>. Citado na página 12.

- LIAO, X.; YUAN, K.; WANG, X.; LI, Z.; XING, L.; BEYAH, R. Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In: **Proceedings of the 2016 ACM SIGSAC conference on computer and communications security**. [S.l.: s.n.], 2016. p. 755–766. Disponível em: <<https://doi.org/10.1145/2976749.2978315>>. Citado na página 12.
- MCMILLAN, R. Definition: threat intelligence. **Gartner. com**, v. 5, 2013. Citado na página 12.
- MIELL, I.; SAYERS, A. **Docker in practice**. [S.l.]: Simon and Schuster, 2019. Citado na página 15.
- PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software-9**. [S.l.]: McGraw Hill Brasil, 2021. Citado na página 15.
- RAHMAN, M. R.; HEZAVEH, R. M.; WILLIAMS, L. What are the attackers doing now? Automating cyberthreat intelligence extraction from text on pace with the changing threat landscape: A survey. **ACM Computing Surveys**, ACM New York, NY, v. 55, n. 12, p. 1–36, 2023. Disponível em: <<https://doi.org/10.1145/3571726>>. Citado 4 vezes nas páginas 10, 12, 13 e 17.
- REIS, A. L. F. Análise de posts maliciosos na dark web usando aprendizado de máquina não supervisionado. Universidade Federal de Uberlândia, 2024. Citado na página 17.
- RUBIO, D.; RUBIO, D. Rest services with django. **Beginning Django: Web Application Development and Deployment with Python**, Springer, p. 549–566, 2017. Disponível em: <https://doi.org/10.1007/978-1-4842-2787-9_12>. Citado na página 14.
- SCELLER, Q. L.; KARBAB, E. B.; DEBBABI, M.; IQBAL, F. Sonar: Automatic detection of cyber security events over the twitter stream. In: **Proceedings of the 12th International Conference on Availability, Reliability and Security**. [S.l.: s.n.], 2017. p. 1–11. Citado na página 17.
- SONI, A.; RANGA, V. Api features individualizing of web services: Rest and soap. **International Journal of Innovative Technology and Exploring Engineering**, v. 8, n. 9, p. 664–671, 2019. Disponível em: <<https://doi.org/10.35940/ijitee.I1107.0789S19>>. Citado na página 14.
- SUN, N.; DING, M.; JIANG, J.; XU, W.; MO, X.; TAI, Y.; ZHANG, J. Cyber Threat Intelligence Mining for Proactive Cybersecurity Defense: A Survey and New Perspectives. **IEEE Communications Surveys & Tutorials**, IEEE, 2023. Disponível em: <<https://doi.org/10.1109/COMST.2023.3273282>>. Citado na página 10.
- TOUNSI, W.; RAIS, H. A survey on technical threat intelligence in the age of sophisticated cyber attacks. **Computers & security**, Elsevier, v. 72, p. 212–233, 2018. Disponível em: <<https://doi.org/10.1016/j.cose.2017.09.001>>. Citado na página 12.
- ZHU, Z.; DUMITRAS, T. Chainsmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports. In: IEEE. **2018 IEEE European symposium on security and privacy (EuroS&P)**. [S.l.], 2018. p. 458–472. Disponível em: <<https://doi.org/10.1109/EuroSP.2018.00039>>. Citado na página 12.