UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Arthur do Prado Labaki

Uso de *web scrapers* para coleta de dados em fóruns especializados de segurança

Uberlândia, Brasil

2023

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Arthur do Prado Labaki

Uso de *web scrapers* para coleta de dados em fóruns especializados de segurança

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 Ciência da Computação.

Orientador: Prof. Dr. Rodrigo Sanches Miani

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2023

Arthur do Prado Labaki

Uso de *web scrapers* para coleta de dados em fóruns especializados de segurança

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 Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 29 de novembro de 2023:

Prof. Dr. Rodrigo Sanches Miani
Orientador

Prof. Dr. Diego Nunes Molinos
Professor

Prof. Dr. Paulo Henrique Ribeiro Gabriel
Professor

Uberlândia, Brasil
2023

Resumo

Fóruns de segurança da informação são espaços virtuais que promovem a troca de conhecimentos e experiências entre profissionais e entusiastas da área. Esses fóruns são essenciais para a construção de uma comunidade colaborativa de segurança da informação, pois permitem a discussão de questões relevantes, como ameaças emergentes, vulnerabilidades e técnicas de mitigação. Portanto, é justificável considerar que essas informações têm valor significativo e podem ser empregadas de diversas maneiras para viabilizar o desenvolvimento de ferramentas capazes de antecipar potenciais incidentes de segurança ou outras questões relacionadas à área. O objetivo deste trabalho é desenvolver *web scrapers* para extrair dados relevantes de um conjunto de fóruns de segurança da informação previamente selecionados, os quais serão armazenados em um banco de dados para, futuramente, serem utilizados como análise em outros trabalhos. Neste trabalho, foram extraídas cerca de 210 mil linhas de dados de quatro diferentes fóruns selecionados. Dentre esses dados, 35 mil correspondem a publicações, abrangendo título, autor, data, categoria e conteúdo, enquanto os 175 mil restantes referem-se a respostas, que incluem autor, data e conteúdo.

Palavras-chave: Segurança da informação, *Web Scraper*, Fóruns, Publicações, Incidentes de segurança.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
ID	*Identity*
JSON	*JavaScript Object Notation*
SQL	*Structured Query Language*
SSL	*Secure Sockets Layer*
TOR	*The Onion Router*
URL	*Uniform Resource Locator*
XML	*Extensible Markup Language*

Sumário

1 Introdução

Com a era da informação em pleno vigor, o mundo contemporâneo vivencia uma explosão de dados e conhecimentos sem precedentes. A ascensão das tecnologias digitais e a interconexão global trouxe consigo uma abundância de fontes de conhecimento, disponíveis ao alcance de um clique. Em uma era caracterizada pela predominância da informação, a disponibilidade da dados se torna um princípio fundamental em todas as áreas do conhecimento. A capacidade de acesso a dados relevantes e atualizados desempenha um papel crucial em inúmeras disciplinas, que influencia desde o avanço da ciência e tecnologia até a tomada de decisões políticas e a eficiência dos processos de negócios (MAYER-SCHÖNBERGER; CUKIER, 2013).

Essa disponibilidade de dados e informações relevantes é crucial para a prevenção e combate a ameaças cibernéticas no contexto específico da segurança da informação. A proteção de sistemas, redes e dados contra ameaças cibernéticas exige um conhecimento profundo das ameaças atuais e das melhores práticas de segurança (MAHMOOD; AFZAL, 2013). Com a crescente sofisticação dos ataques virtuais e a constante evolução das táticas empregadas por agentes mal-intencionados, a necessidade de acesso rápido e preciso a informações sobre incidentes de segurança tornou-se uma prioridade incontestável.

A disponibilidade de informações relevantes e atualizadas é, portanto, essencial para a prevenção de ataques, a detecção de vulnerabilidades e a resposta a riscos de segurança (WILLIAMS; WOODWARD, 2015). A falta de informações confiáveis pode deixar organizações vulneráveis a ataques e colocar em risco informações sensíveis, evidenciando a necessidade premente de estratégias eficazes de coleta, análise e disseminação de dados relacionados à segurança da informação (CRAIGEN; DIAKUN-THIBAULT; PURSE, 2014).

Com isso, de acordo com Motoyama et al. (2011), os fóruns especializados em segurança cibernética desempenham um papel crucial na disseminação e compartilhamento de conhecimento. Estes espaços virtuais servem como repositórios valiosos de conhecimento, onde especialistas compartilham experiências, discutem tendências emergentes e fornecem orientações práticas para enfrentar desafios no campo da segurança cibernética. A interação entre membros dessas comunidades contribui para o desenvolvimento coletivo do conhecimento, promovendo uma abordagem colaborativa e adaptativa na luta contra ameaças virtuais em constante evolução. Contudo, a vasta quantidade de informações geradas em fóruns especializados pode representar um desafio considerável para a coleta e análise de dados (MOTOYAMA et al., 2011).

No entanto, a eficaz extração de informações pertinentes desses fóruns representa

um desafio significativo. A abundância de dados disponíveis, aliada à diversidade de tópicos discutidos e à constante dinâmica das interações, dificulta a tarefa de identificar e capturar os conhecimentos essenciais para a pesquisa e a análise em segurança cibernética (MITCHELL, 2018). O processo manual de coleta de dados é moroso e sujeito a falhas, além de ser inviável para lidar com grandes volumes de informações. Este cenário demanda uma abordagem mais ágil e automatizada para a obtenção de dados pertinentes (THELWALL, 2001).

É sob essa perspectiva que se coloca a hipótese deste estudo. A aplicação de *web scrapers*, ferramentas de extração de dados automatizadas, pode representar uma solução promissora para otimizar e agilizar a coleta de informações em fóruns especializados de segurança cibernética. A ideia subjacente é que a implementação de *web scrapers* personalizados, adaptados às particularidades desses fóruns, pode proporcionar um método eficaz para capturar e processar os dados necessários para análises substanciais e informadas em segurança da informação.

1.1 Objetivos

O objetivo deste trabalho é identificar e selecionar fóruns que promovem discussões acerca de tópicos relacionados à segurança da informação, e, com base nesta seleção, elaborar mecanismos para automatizar a coleta e o armazenamento organizado das postagens desses fóruns, visando facilitar a análise dos dados em etapas posteriores. Os objetivos específicos desta monografia são:

1. Compreender o funcionamento de *web scrapers*;
2. Identificar um conjunto de fóruns com postagens relevantes de assuntos ligados a segurança;
3. Analisar e compreender o código-fonte dos fóruns de segurança da informação selecionados;
4. Desenvolver *web scrapers* para realizar a extração das postagens de maneira eficiente;
5. Estruturar os dados obtidos para fácil análise a posteriori.

1.2 Justificativa

A crescente complexidade das ameaças cibernéticas e a grande importância da segurança da informação demandam abordagens inovadoras para a coleta e análise de dados em fóruns especializados. Para a implementação de estratégias eficazes de proteção contra ataques cibernéticos, é crucial contar com informações atualizadas e relevantes para

a tomada de decisões informadas (LANDERS et al., 2016). A extração manual de dados em fóruns de segurança cibernética é uma tarefa demorada e sujeita a erros, limitando a capacidade de resposta e a eficácia das práticas de segurança. Portanto, a aplicação de técnicas de automação, como a utilização de *web scrapers*, representa uma oportunidade significativa para otimizar a coleta de dados, possibilitando uma análise mais abrangente e uma resposta mais ágil às ameaças digitais em constante evolução (VARGIU; URRU, 2013).

Além disso, de acordo com Mitchell (2018) a crescente recorrência de incidentes de segurança cibernética como um fator crucial na necessidade de implementar abordagens mais eficazes na extração de dados. Com ataques cada vez mais sofisticados e frequentes, a capacidade de obter e analisar dados pertinentes torna-se uma prioridade incontestável. A utilização de *web scrapers* para a coleta de dados em fóruns especializados de segurança cibernética não apenas agiliza o processo, mas também possibilita uma resposta mais ágil e informada diante de incidentes, que oferece conhecimentos valiosos que podem ser cruciais na identificação e mitigação de ameaças em tempo real (LANDERS et al., 2016).

Esses tipos de dados armazenados não só possibilitam uma análise retrospectiva para entender padrões de ataques passados, mas também servem como um recurso valioso para a detecção proativa de ameaças emergentes. A capacidade de identificar rapidamente novos vetores de ataque, estratégias maliciosas e vulnerabilidades potenciais concede uma vantagem significativa na defesa contra ameaças cibernéticas em constante evolução. Além disso, esses dados podem alimentar sistemas de inteligência artificial e aprendizado de máquina, aprimorando a capacidade de prever e responder a eventos de segurança com maior eficácia (MOTOYAMA et al., 2011).

É esperado que este trabalho fornecerá informações relevantes sobre a eficiência do uso de *web scrapers* na coleta de dados em fóruns especializados de segurança cibernética, estabelecendo uma base sólida para o avanço das práticas de pesquisa e monitoramento no campo da segurança digital. Além disso, aguarda-se identificar melhores práticas, desafios e limitações associadas ao uso de automações nesse contexto, culminando em um aprimoramento significativo na capacidade de resposta e na proteção contra ameaças cibernéticas.

O restante do trabalho está organizado da seguinte forma. O Capítulo 2 aborda o referencial teórico, discutindo conceitos essenciais para esta monografia, bem como trabalhos correlatos. O Capítulo 3 trata do desenvolvimento de cada objetivo delineado. No Capítulo 4, são apresentados os resultados obtidos através dos *web scrapers* criados. Finalmente, o Capítulo 5 conclui o trabalho com as considerações finais.

2 Referencial Teórico

Para o desenvolvimento deste trabalho, é essencial compreender alguns conceitos básicos tais como segurança da informação no contexto da Internet, a função e impacto dos *web scrapers*, a utilidade dos fóruns especializados em segurança da informação, a estrutura de armazenamento, especialmente no formato textual, bem como uma exploração dos recursos da linguagem de programação *Python* e dos *frameworks* utilizados na realização desta monografia. Por fim será realizada uma breve análise dos trabalhos correlatos.

2.1 Segurança da informação

Segurança da informação é o conjunto de medidas e práticas que visam proteger os dados e sistemas de informação contra ameaças, sejam elas externas ou internas, tais como acesso não autorizado, uso indevido, divulgação, modificação ou destruição. Além disso, é essencial considerar os princípios fundamentais que garantem a proteção e integridade das informações, a disponibilidade dos recursos quando necessários, a validação da identidade dos usuários e a irrefutabilidade das ações realizadas (VACCA, 2012).

De acordo com Brown e Stallings (2017), é possível dividir esse conceito nos seguintes pilares:

- **Confidencialidade:** proteção das informações contra acessos não autorizados, garantindo que as informações só sejam acessadas por pessoas autorizadas e que sejam mantidas em sigilo;
- **Integridade:** proteção das informações contra alterações não permitidas, garantindo que as informações não sejam alteradas por pessoas não autorizadas, assim como certificar que as informações mantenham sua integridade durante o processamento e armazenamento;
- **Disponibilidade:** garantir que as informações estejam disponíveis para uso quando necessário, assegurando que os sistemas de informação estejam sempre em funcionamento e que as informações sejam acessíveis de maneira rápida e eficiente.
- **Autenticação:** processo que verifica a identidade de um usuário, sistema ou entidade, garantindo que apenas indivíduos autorizados tenham acesso às informações e recursos protegidos.

- **Não-repúdio:** assegura que uma parte envolvida em uma transação não possa negar sua participação ou a autenticidade das informações trocadas, fornecendo evidências irrefutáveis da ocorrência da comunicação ou transação.

A área de segurança da informação é muito importante para garantir a proteção dos dados e sistemas de informação. A realização de estudos de falhas e vulnerabilidades em aplicações é uma parte importante dessa área, pois permite identificar e corrigir problemas de segurança antes que eles sejam explorados por atacantes, acarretando incidentes de segurança.

As ameaças cibernéticas constituem um espectro diversificado de perigos que podem comprometer a integridade, confidencialidade e disponibilidade de dados e sistemas de informação (SCHULTZ; SHUMWAY, 2001). Essas ameaças abrangem uma variedade de ataques maliciosos, como *malware*, que visa infectar sistemas para ganhos ilícitos, o *phishing*, que utiliza engenharia social para obter informações sensíveis e ataques de negação de serviço, que visam sobrecarregar sistemas e torná-los inacessíveis, entre outros tipos (MANDIA; PROSISE, 2001).

Além disso, o campo de segurança da informação encontra-se em constante evolução, pois as ameaças cibernéticas estão sempre se atualizando e novas vulnerabilidades são descobertas com o tempo. Portanto, a criação de novas ferramentas e métodos é essencial para possibilitar a contenção dessas ameaças, como é importante também que os profissionais da área estejam sempre atualizados e capacitados para lidar com esses desafios (PFLEEGER; PFLEEGER, 2012).

2.2 Web Scraping

O *web scraping*, também conhecido como *web crawling* ou raspagem de dados da web, refere-se ao processo automatizado de extração de informações de páginas da Internet. Esse processo envolve a utilização de *web scrapers*, conhecidos como *scrapers* ou *crawlers*, que navegam pelos *websites*, identificam e capturam dados específicos, transformando-os em um formato estruturado para posterior análise, conforme pode ser observado na Figura 1. Os *web scrapers* podem ser projetados para coletar textos, imagens, tabelas e outros elementos presentes nas páginas web (GLEZ-PEñA et al., 2013).

Embora os termos *crawler* e *scraper* sejam frequentemente usados de forma intercambiável, Glez-Peña et al. (2013) afirma que eles têm algumas diferenças sutis. Um *web crawler* é um programa que navega na web de forma automatizada, coletando informações de páginas da web, enquanto um *web scraper* é um programa que extrai informações específicas de uma página da web.

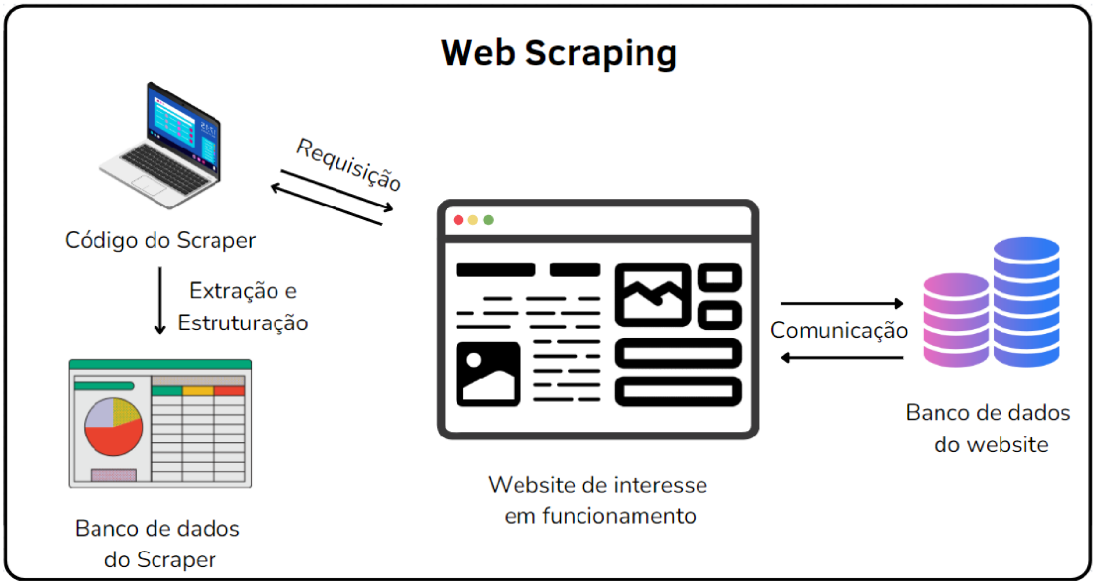

Figura 1 – Processo simplificado de um *Web Secaping*. Fonte: Do Autor.

De acordo com Mitchell (2018) funcionamento básico de um *web scraper* envolve os seguintes passos:

1. Requisição: O *web scraper* envia solicitações *HTTP* para links específicos, simulando o comportamento de um navegador;
2. Carregamento e Análise: O *HTML* da página web é carregado e analisado pelo *web scraper* para identificar os elementos relevantes a serem extraídos;
3. Extração: Os dados desejados são extraídos do *HTML*, seguindo padrões definidos, como seletores CSS ou expressões regulares;
4. Estruturação: Os dados extraídos são transformados em um formato estruturado, como *CSV*, *JSON* ou bancos de dados;
5. Armazenamento: Os dados estruturados podem ser armazenados localmente ou em sistemas de gerenciamento de dados.

Em tempos atuais, o *web scraping* desempenha um papel significativo em várias áreas do conhecimento (THELWALL, 2001). Por exemplo, pesquisadores podem utilizá-lo para coletar dados para análises, estudos de tendências e compilação de informações para suas pesquisas, empresas podem empregar essa técnica para monitorar os preços praticados pelos concorrentes, obter o *feedback* dos clientes e discernir padrões de mercado. Além disso, o *web scraper* é uma ferramenta valiosa para o treinamento de modelos de aprendizado de máquina, que promove aprimoramentos notáveis na precisão e eficácia desses modelos.

É crucial reconhecer que, embora *web scraping* ofereça benefícios significativos, essa técnica também levanta considerações éticas e legais complexas. Muitos sites possuem termos de uso que explicitamente proíbem a coleta automatizada de dados, e ignorar esses termos pode resultar em ações ilegais. Ademais, a coleta de dados pessoais ou sensíveis sem consentimento viola a privacidade dos usuários e pode infringir regulamentações de proteção de dados.Além das implicações legais, o uso excessivo de web scraping pode sobrecarregar servidores, prejudicando a experiência dos usuários legítimos e comprometendo a disponibilidade dos recursos, pois a intensidade da atividade de *web scraping* pode resultar em um aumento significativo no tráfego de dados, consumindo largura de banda e recursos do servidor (MITCHELL, 2018).

2.3 Fóruns especializados em segurança da informação

Os fóruns especializados têm uma importância significativa no campo da cibersegurança, pois servem como plataformas virtuais para compartilhar conhecimentos, colaborar e trocar informações entre profissionais, pesquisadores e entusiastas da área. Esses grupos, também conhecidos como comunidades de *hackers*, desempenham um papel importante no aprendizado coletivo, permitindo que pessoas de diferentes níveis de experiência compartilhem ideias, conhecimentos e aprendizados. A colaboração e resolução de problemas são as bases desses fóruns, onde profissionais enfrentam desafios complexos juntos, oferecendo soluções alternativas e orientações valiosas. Além disso, os fóruns atuam como um radar para identificar tendências emergentes e ameaças recentes na cibersegurança, mantendo os participantes atualizados e preparados para responder às evoluções do cenário digital (WAGNER et al., 2019).

Um bom exemplo é o *0x00sec*[1], um fórum completamente focado em segurança da informação e computação em geral, disponível na Internet convencional[2]. Nele existem diversas categorias como *Programming*, *Networking*, *Exploits*, *Reverse Engineering*, e muitas outras. Os membros compartilham conhecimentos, discutem tendências e se apoiam na resolução de desafios de segurança. Além disso, o fórum também oferece tutoriais detalhados e recursos valiosos para profissionais e entusiastas da segurança cibernética. A comunidade ativa e engajada torna o *0x00sec* um ambiente propício para aprendizado e aprimoramento contínuo no campo da segurança da informação.

Apesar das vantagens oferecidas pelos fóruns especializados de segurança da informação, também é importante reconhecer os problemas potenciais associados a essas plataformas. Alguns desses desafios incluem a divulgação irresponsável de informações sensíveis, o compartilhamento de técnicas e ferramentas maliciosas, conflitos éticos e le-

[1] https://0x00sec.org/

[2] A "Internet Convencional"refere-se à parte acessível publicamente da *World Wide Web*, enquanto a "*Dark/Deep Web*"consiste em áreas não indexadas por mecanismos de busca convencionais.

gais em relação às discussões, disseminação de desinformação, bem como a exposição a ataques e ameaças cibernéticas. Além disso, existe a possibilidade de que os participantes adquiram uma falsa sensação de competência ou dependam exclusivamente das informações dos fóruns (MITNICK, 2011). Para mitigar esses problemas, é crucial praticar uma abordagem cautelosa, adotar a divulgação responsável de informações, verificar a credibilidade dos fóruns e membros, e manter uma postura ética e legal ao participar dessas comunidades.

2.4 Armazenamento de dados

De acordo com Meier e Kaufmann (2019), a maneira mais eficaz de organizar e armazenar informações, além de permitir o acesso rápido e preciso, são com banco de dados. Eles são coleções organizadas de dados que se relacionam entre si para garantir a melhor eficiência, independentemente dos tipos de dados, como informações pessoais, dados financeiros, dados de transações comerciais, entre outros. Esses relacionamentos entre dados podem ser classificados em duas diferentes classes, sendo banco de dados relacional e não relacional.

Bancos de dados relacionais são baseados em tabelas e relações entre elas, onde cada tabela possui uma entidade fundamental chamada de chave primária usada para estabelecer relações entre as tabelas. Além disso, eles possuem uma estrutura rígida e pré-definida, que seguem esquemas específicos baseados na álgebra relacional, gerando integridade referencial, que garante consistência entre as tabelas, além de fornecer suporte a consultas complexas. Alguns exemplos de sistemas gerenciadores de banco de dados relacionais são o *MySQL* e o *PostgreSQL*.

Já o banco de dados não relacional, normalmente chamado de *NoSQL*, é estruturado por documentos e não por tabelas, o que impede a criação de uma forte relação entre os dados. Mesmo com essa desvantagem, essa característica torna o banco de dados muito mais flexível, permitindo armazenar dados de diferentes tipos e estruturas em um mesmo documento. Nele ainda é possível acrescentar propriedades sem se preocupar em como as novas informações impactarão as já existentes, o que gera uma grande escalabilidade, tanto horizontalmente quanto verticalmente. Essa categoria de banco troca a alta performance por flexibilidade nos dados e facilidade de armazenamento (MEIER; KAUFMANN, 2019).

2.5 Python e frameworks

Python é uma linguagem de programação de alto nível conhecida por sua simplicidade e legibilidade. Criada por Guido van Rossum e lançada pela primeira vez em

1991, *Python* se tornou uma das linguagens mais populares devido à sua simplicidade e sua vasta gama de bibliotecas e *frameworks* (RAWAT, 2020). Sua sintaxe otimizada e estrutura clara favorecem o desenvolvimento rápido e eficiente, o que permite com que os programadores expressem suas ideias de forma concisa.

Uma das principais características do *Python* é sua ênfase na legibilidade do código, incentivando a escrita de programas que se assemelham a pseudocódigo em termos de clareza. Isso torna a linguagem acessível a iniciantes e experientes, o que facilita a colaboração em projetos. O *Python* é interpretado, o que significa que não requer um processo de compilação separado, permitindo um fluxo de desenvolvimento mais ágil e menos propenso a erros. Além disso, a linguagem suporta múltiplos paradigmas de programação, o que inclui programação funcional e orientada a objetos, oferecendo flexibilidade aos desenvolvedores (ROSSUM, 2007).

A popularidade do *Python* também se deve à sua comunidade ativa, que contribui para a criação de bibliotecas e ferramentas poderosas. Dentre elas é possível destacar *Selenium*, *Requests* e *Beautiful Soup* como as principais bibliotecas utilizadas nesta monografia.

2.5.1 Selenium

Selenium é uma ferramenta popular de automação de navegador que permite a interação programática com páginas da web. É frequentemente utilizado para testes automatizados de software, raspagem de dados da web e simulação de interações de usuário em sites. A biblioteca *Selenium* oferece uma *API* em várias linguagens de programação, incluindo *Python*. Com o *Selenium*, é possível criar *scripts* que controlam um navegador da web real, como *Chrome* ou *Firefox*, para realizar ações como clicar em botões, preencher formulários, navegar por páginas e extrair informações.

Além disso, o *Selenium* oferece suporte a diferentes navegadores, permitindo uma ampla gama de opções para os desenvolvedores. A biblioteca também inclui recursos avançados, como a capacidade de lidar com janelas *pop-up*, manipular *cookies* e esperar por elementos específicos antes de realizar ações, tornando-a uma ferramenta poderosa para automação de testes e interações complexas na Internet. Com a combinação da flexibilidade do *Python* e a robustez do *Selenium*, os desenvolvedores têm à disposição uma ferramenta eficaz para diversas necessidades relacionadas à interação automatizada com páginas da web.

2.5.2 Requests

A biblioteca *Requests* é uma ferramenta popular em *Python* que permite realizar requisições HTTP de forma simples e eficaz. Ela permite que os desenvolvedores enviem

solicitações HTTP para servidores web e interajam com *APIs*, recuperando informações, enviando dados e gerenciando *cookies*. A biblioteca simplifica a complexidade de lidar com a comunicação HTTP, o que torna o processo mais legível e conveniente para os programadores. Seus principais recursos são:

- Realizar uma variedade de requisições *HTTP*, como *GET*, *POST*, *PUT* e *DELETE*;
- Personalizar cabeçalhos de requisição para fornecer informações específicas ao servidor;
- Gerenciar sessões persistentes com suporte a *cookies*, permitindo autenticações contínuas;
- Lidar automaticamente com autenticação básica ou mais complexa, como autenticação por *token*;
- Decodificar automaticamente conteúdo de respostas, independentemente do formato;
- Verificar a segurança de sites por meio de de verificações de *SSL* semelhantes aos navegadores;
- Gerenciar conexões e compartilhar recursos de conexão para melhorar a eficiência;
- Dentre outros.

2.5.3 Beautiful Soup

Por fim, o *Beautiful Soup* é uma biblioteca *Python* amplamente usada para analisar e extrair informações de documentos *HTML* e *XML*, sendo uma escolha popular quando é necessário coletar dados específicos de páginas da web. Ela funciona transformando o código-fonte de uma página em uma árvore de objetos do *Python*, representando a estrutura do documento, o que facilita a busca e extração de elementos, como links e textos.

Com a biblioteca *Beatiful Soup* é possível localizar elementos *HTML* por suas *tags*, classes, *IDs* e outros atributos, permitindo a extração de informações específicas. Além disso, é possível acessar o conteúdo de texto, atributos e outros dados dentro dos elementos *HTML*, como imagens e informações de tabelas. A biblioteca também oferece métodos para navegar pela estrutura hierárquica do documento, o que torna mais descomplicado o acesso a elementos relacionados. Além de extrair informações, ela também permite modificar o conteúdo dos documentos *HTML* ou *XML*, possibilitando a manipulação de dados.

2.6 Trabalhos Correlatos

Nesta seção, serão discutidos trabalhos que se dedicaram tanto ao desenvolvimento quanto a utilização de *web scrapers* ou *crawlers* para aquisição de dados provenientes de diversas fontes na Internet e para diferentes usos.

Teixeira (2022) investigou e demonstrou a criação de *crawlers* para obter dados de fóruns da *Dark Web*. Nele é possível compreender a dificuldade de localizar links para sites nessa região, devido à alta instabilidade dos sites, o que gera uma vida útil baixa dependendo do serviço oferecido, criando a necessidade de desenvolver ferramentas que automatizam essa etapa de obtenção dos links para análise e coleta de dados. Com essa dificuldade, também foi desenvolvido uma ferramenta destinada a realizar navegação automatizada na *darknet*, com o propósito de extrair novos links provenientes de outros serviços disponíveis na rede *TOR*, por meio de um ponto de acesso específico. O autor encontrou o fórum mais adequado e desenvolveu um *web scraper* para coletar seus dados, que utiliza as bibliotecas *Beautiful Soup* e *Requests*, usando uma versão otimizada para a rede *Tor*. Como resultado, ele conseguiu extrair um total de 19.654 postagens, das quais 2.689 foram categorizadas como relacionadas a *hacking*, para uma possível análise futura.

Ainda sobre o tema da *Dark Web*, Schäfer et al. (2019) apresenta o *BlackWidow*, que é um sistema modular altamente automatizado que monitora os serviços da *Dark Web* e funde os dados coletados em uma única estrutura analítica. Ele também consegue obter dados em tempo real e de forma contínua de sites e serviços nessa parte da *web*, além de realizar a parametrização da busca por links, ou seja, dado manualmente os primeiros links de fóruns, ele analisa o conteúdo desses fóruns para obter mais endereços para outros alvos de maneira automatizada em iterações posteriores. Além dessas características, o *BlackWidow* consegue realizar a tradução automática de sites, que pode ultrapassar a barreira linguística.

Para conseguir obter os dados, Schäfer et al. (2019) utiliza o *crawler Puppeteer*, que é uma biblioteca de código aberto do *Google* que permite controlar um navegador de maneira automatizada, semelhante ao *Selenium*. Os alvos dessa coleta foram sete fóruns de diferentes linguagens, com um total de cerca de 550 mil postagens extraídas. Após essa etapa, o *BlackWidow* ainda realiza uma análise sobre dados, gerando um gráfico de conhecimento que relaciona as postagens obtidas.

Han e Anderson (2021) aborda a transformação da Internet em um canal de distribuição de destaque para reservas de viagens e enfatiza a importância da análise do comportamento do cliente online. Além disso, os autores afirmam que muitos pesquisadores na área de hospitalidade ainda continuam a depender de métodos tradicionais de coleta de dados, com limitações no que diz respeito às questões de pesquisa abordadas e à aplicabilidade de seus resultados. Com essas análises, é proposto uma solução simples

para a coleta de dados online, utilizando a linguagem de programação *Python*, com foco na exploração do processo de *scraping* em plataformas de viagens amplamente utilizadas.

Os autores também demonstram orientações práticas sobre a utilização das técnicas de *XPath* e *Selenium* para a obtenção de informações de sites de viagens. Ao final, Han e Anderson (2021) ainda propõe o uso dos *web scrapers* desenvolvidos em três sites diferentes de plataformas de viagens, em que é obtido diversas informações úteis, como a diferença entre os preços ofertados, a distribuição das pontuações das avaliações e até a diferença entre as avaliações por plataforma.

Já Budiarti et al. (2016) tem um objetivo diferente, automatizar a coleta de dados de qualidade da água por meio de sensores em várias áreas. Para isso, são utilizados controladores instalados em diferentes locais para monitorar indicadores de qualidade da água, tais como pH, oxigênio dissolvido, turbidez, sólidos suspensos totais, temperatura da água e cloro. A coleta de dados é realizada utilizando a biblioteca *Beautiful Soup* em *Python*, a qual é empregada para extrair informações de documentos *HTML* gerados pelos sensores. Esses dados são então processados e armazenados em um banco de dados *PostgreSQL* para análises posteriores. Como resultado dessa coleta de dados, o trabalho atingiu uma precisão de aproximadamente 99%, com uma taxa de erro inferior a 1%.

Chaulagain et al. (2017) introduz uma abordagem inovadora que utiliza a computação em nuvem como sua base. A crescente necessidade de coletar informações da web de forma escalável e econômica levou os autores a desenvolverem uma arquitetura que utiliza serviços da *Amazon* para criar um ambiente de raspagem de dados distribuída, permitindo com que as organizações coletam dados de forma eficiente, escalável e flexível, com a capacidade de aumentar ou diminuir os recursos de acordo com a demanda.

Além disso, Chaulagain et al. (2017) ainda destacam a importância do uso do *framework Selenium* para a raspagem de dados devido à sua capacidade de imitar o comportamento humano na web, o que torna a coleta de informações mais eficaz e confiável. Os resultados do experimento demonstram a eficácia dessa abordagem, mostrando que a arquitetura baseada em nuvem é escalável, eficiente em termos de custos e adequada para processar grandes volumes de dados. A capacidade de integrar essa arquitetura a aplicativos de *big data* também destaca seu potencial para análise e visualização de dados em larga escala.

Os trabalhos mencionados nesta seção compartilham objetivos semelhantes, que consistem em coletar dados de várias fontes para uma análise mais aprofundada posteriormente. No entanto, cada artigo utiliza metodologias diferentes para alcançar esse objetivo. O trabalho proposto segue uma abordagem semelhante a esses artigos, envolvendo a criação de *web scrapers* para extrair dados de sites específicos relacionados à segurança da informação. Diferentemente dos artigos anteriores, que, no máximo, compartilharam o código-fonte de seus *web scrapers*, este trabalho busca não apenas apresentar os códigos

dos *web scrapers*, mas também divulgar os nomes dos fóruns e todas as informações associadas à coleta de dados. O propósito dessa coleta de dados é permitir uma análise mais detalhada no futuro, utilizando os dados obtidos com a extração.

3 Desenvolvimento

Neste capítulo, será apresentada uma visão geral das etapas que compõem o desenvolvimento deste trabalho, seguido pela explicação e demonstração de cada uma dessas etapas.

3.1 Descrição geral

Os passos do desenvolvimento do trabalho foram divididos da seguinte forma:

1. Procurar e selecionar os fóruns que serão utilizados;
2. Analisar o código-fonte dos sites selecionados;
3. Desenvolver os *scripts* dos *web scrapers* para obter os dados;
4. Utilizar técnicas para melhorar a eficiência dos *web scrapers* criados;
5. Inserir as informações obtidas no banco de dados.

Na Seção 3.2, são apresentados os procedimentos utilizados para a busca dos fóruns e os critérios de seleção adotados. A Seção 3.3 aborda a análise estrutural dos sites escolhidos, identificando os elementos-chave para a posterior criação dos *scripts*. Nas Seções 3.4 e 3.5, são discutidos o desenvolvimento e aprimoramento dos códigos dos *web scrapers*, visando à obtenção dos dados com um bom desempenho. Por fim, a Seção 3.6 engloba a criação das tabelas no banco de dados, bem como o desenvolvimento do script para inserir os dados obtidos. É importante destacar que todos os códigos desenvolvidos estão acessíveis em um repositório no *GitHub*, conforme detalhado no Capítulo 4.

3.2 Seleção dos Fóruns

Nesta primeira etapa, foram realizadas diversas buscas manuais visando encontrar o maior número possível de fóruns. Durante essa investigação, foi identificado o site *0ut3r Space* (HOEK, 2023), que é um site pessoal contendo notícias, artigos, guias e links relacionados à tecnologia da informação. Nesse site, foi encontrado uma postagem sobre fóruns de hackers, onde o autor discute a dificuldade de encontrar fontes de alta qualidade sobre o tema e a importância de adquirir conhecimento através de livros, cursos e práticas. A postagem também inclui links para alguns fóruns, com um alerta aos leitores para usá-los com cautela e evitar confiar em todas as informações encontradas.

No total foram obtidos 60 links diferentes de fóruns sobre *hacking*, *cracking* ou *carding*. Com essas informações, foi realizado uma primeira análise dos sites para identificar os que melhores se enquadram na proposta deste trabalho. Para isso, foram utilizados os seguintes critérios de seleção:

- Alta frequência de acessos ao site, com boa quantidade de *posts* novos e respostas por mês;
- Deve existir uma ou mais categorias de segurança da informação;
- Deve ser possível ver o *post* e respostas sem precisar realizar o login na maioria das vezes;
- O site deve oferecer acesso ilimitado aos *posts* e, em sua maior parte, ser gratuito;
- Deve ter uma alta estabilidade, com poucas quedas, ficando disponível a maioria do tempo;
- Não pode existir nenhuma medida contra *scraping*, como o *captcha*.

Após a avaliação com base nesses critérios, foram selecionados os seguintes fóruns:

1. *Legit Carders* [1]
2. *Hackonology Forums* [2]
3. *NulledBB* [3]
4. *Niflheim World* [4]

3.3 Analisando o código-fonte da página web

Com base nos fóruns selecionados, foi realizada uma primeira análise geral dos sites com o intuito de compreender suas estruturas e determinar a melhor abordagem para o desenvolvimento dos *web scrapers* de maneira eficiente. Os quatro fóruns apresentam semelhanças em sua estrutura, seguindo um formato padronizado que facilita a navegação e a interação dos usuários, em que proporciona uma experiência consistente para os participantes e ajuda a manter a organização e a compreensão das discussões. Além disso, a estrutura semelhante pode ser benéfica para os administradores do fórum, o que pode facilitar a moderação e a manutenção do ambiente online.

[1] https://legitcarders.ws
[2] https://hackonology.com/forum/index.php
[3] https://nulledbb.com
[4] https://niflheim.world

3.3.1 Analisando as categorias

As postagens são feitas em diferentes categorias para organizar e agrupar discussões com base em temas ou assuntos similares, proporcionando uma organização que beneficia tanto os administradores quanto os usuários. Essa estrutura facilita a navegação, estimula a especialização e mantém um ambiente propício e adequado para a discussão.

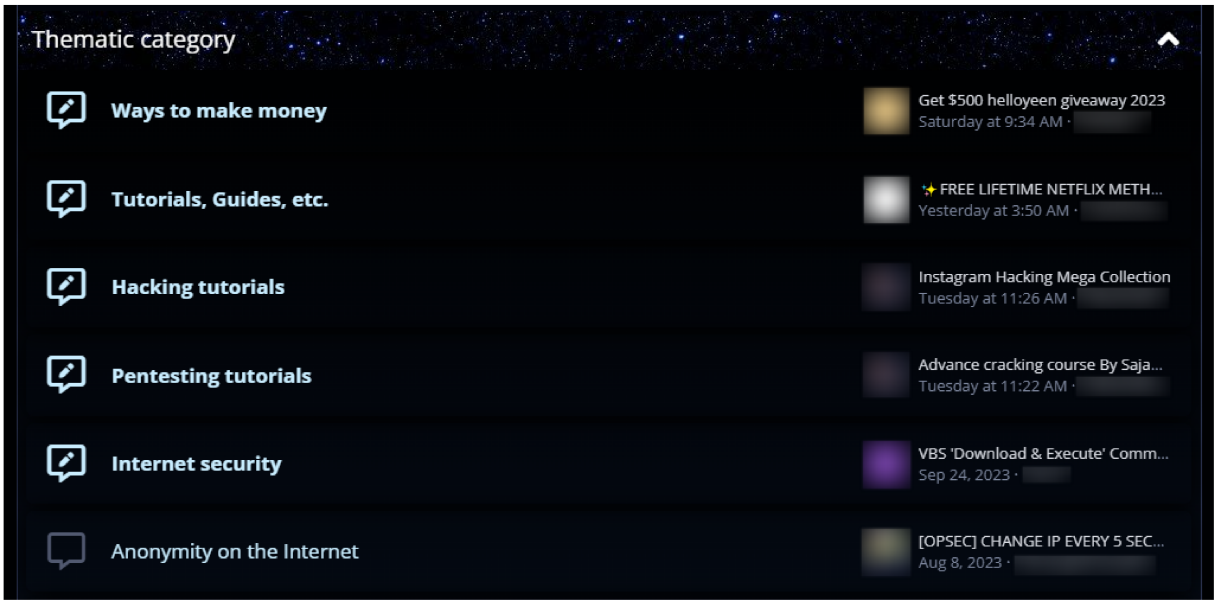

Figura 2 – Demonstração das categorias do fórum *Niflheim World*.

Existem diversas categorias, como *Computing*, *Hacking*, *Programming* e *Technology*. Conforme ilustrado na Figura 2, as categorias são separadas em uma lista, geralmente na página principal do fórum. Normalmente existem categorias gerais, como a *Thematic category*, em que contém as categorias principais, como a *Hacking tutorials*, além de que é possível existirem subcategorias para as categorias principais.

Dado que existem categorias que não estão diretamente ligadas à segurança da informação ou tecnologia em geral, a escolha delas em cada site foi realizada manualmente, envolvendo a criação de um dicionário para cada categoria a ser extraída, como será detalhado na Seção 3.4. Além disso, a presença de diversas categorias em cada fórum, identificadas por diferentes nomes e links para cada site, reforça a necessidade de realizar essa seleção de forma manual, sendo crucial para evitar a extração de dados irrelevantes.

3.3.2 Inspecionando as listas de publicações

Dentro de cada categoria, existe uma pré-visualização de cada postagem realizada, em que, na maioria das vezes é exibido o título da postagem, o nome do autor, a data da postagem, o número de comentários e visualizações. Ainda é possível que este *pre-*

view contenha a última resposta publicada, a quantidade de *likes*, *rating* ou algum outro medidor de engajamento.

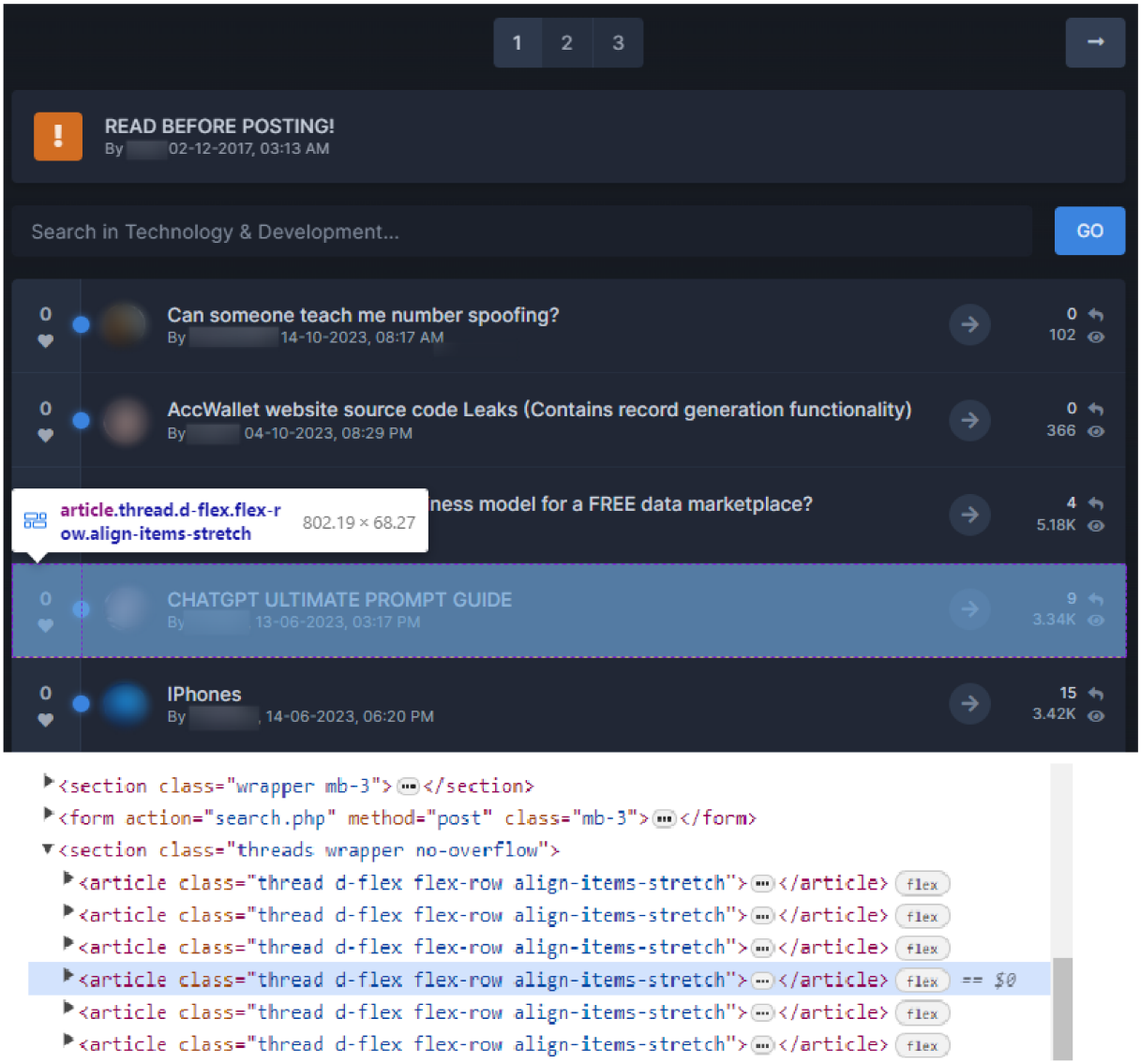

Figura 3 – Inspecionando um elemento da lista de *posts* do *NulledBB*.

Conforme demonstrado na Figura 3, cada pré-visualização é representada pela *tag HTML article*, em que contém todas as informações visualizadas, além da URL do próprio *post*. Todos os fóruns têm essa mesma estrutura, alterando apenas o nome da classe, o que facilita o desenvolvimento de seus *web scrapers* individuais.

3.3.3 Investigando publicações e respostas

Cada postagem possui sua própria página individual, na qual, além das informações já mencionadas, é apresentado o conteúdo da postagem, podendo incluir links e imagens. Adicionalmente, é possível escrever respostas a esses *posts*, o que gera um ambiente ainda mais propício para discussões em grupo.

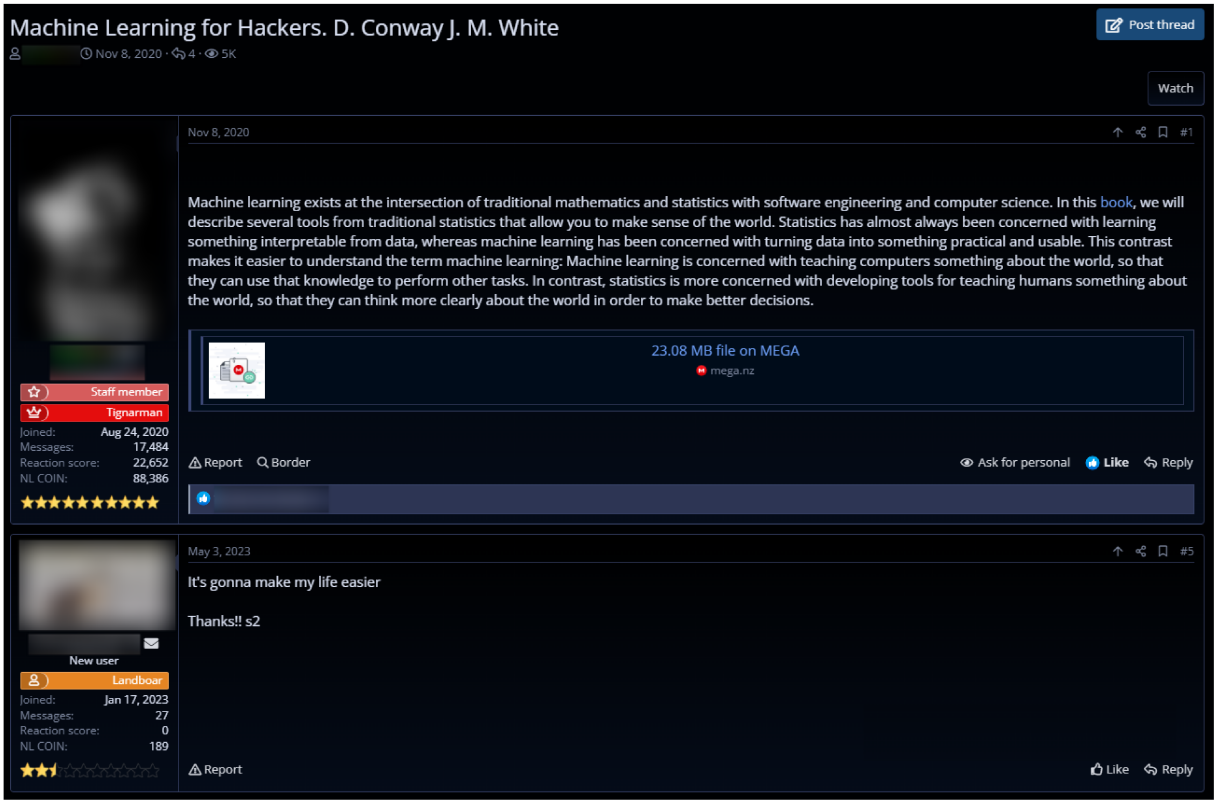

Figura 4 – Exemplo de uma postagem do *Niflheim World*.

Observando a Figura 4, é possível perceber que todas as informações que devem ser extraídas se concentram nas páginas de cada publicação, como o título, o autor, as datas, o conteúdo e outros dados relevantes. No entanto, cada fórum gera essas páginas de maneiras diferentes, o que resulta em um código-fonte diferente para cada site. Com este resultado, cada *web scraper* deve realizar essa busca de informações de forma específica para cada site, impossibilitando a criação de um único *web scraper* para todos os fóruns selecionados.

Após todas essas análises, foi possível compreender e selecionar quais informações podem ser obtidas. Com base nessa seleção, foi obtido:

- Título da publicação;
- Autor da publicação;
- Categoria da publicação;
- Data da publicação;
- Conteúdo da publicação;
- Respostas da publicação:
  - Autor da resposta;

- Data da resposta;
- Conteúdo da resposta.

3.4 Desenvolvimento dos *web scrapers*

Após concluir todas as análises e seleção dos alvos, além de confirmar quais informações serão obtidas dos fóruns, foi possível iniciar o desenvolvimento dos códigos dos *scripts* dos *web scrapers*.

3.4.1 Requisitos gerais do *web scraper*

Antes de iniciar o desenvolvimento do código, é crucial descrever as necessidades, características, funcionalidades e restrições que o *web scraper* deve ter. Ao analisar o progresso até então realizado, foram identificados os seguintes itens:

- O *web scraper* deve extrair todas as publicações e respostas de todas as categorias selecionadas;
- O *web scraper* deve ter uma lógica simplificada e objetiva;
- O *web scraper* deve ser capaz de extrair todas as informações previamente selecionadas;
- O *web scraper* deve ser eficiente, processando grandes volumes de dados em tempo hábil;
- O *web scraper* deve ser desenvolvido em *Python* e utilizar as bibliotecas *Requests*, *Selenium* e *Beautiful Soup* para a extração eficiente de dados;
- O *web scraper* deve operar em conformidade com as leis de proteção de dados;
- O acesso ao fórum e a extração de dados devem ser realizados de acordo com os termos de serviço do fórum, evitando a violação de políticas do site.

3.4.2 Estrutura básica dos *web scrapers*

Devido à semelhança estrutural dos fóruns selecionados, foi viável estabelecer um conceito fundamental que pode ser implementado em todos. As distinções se manifestaram apenas durante a coleta de dados.

Como também pode ser observado na Figura 5, ao iniciar o *script*, é necessário realizar o acesso ao site do fórum para confirmar sua disponibilidade. Em seguida, deve

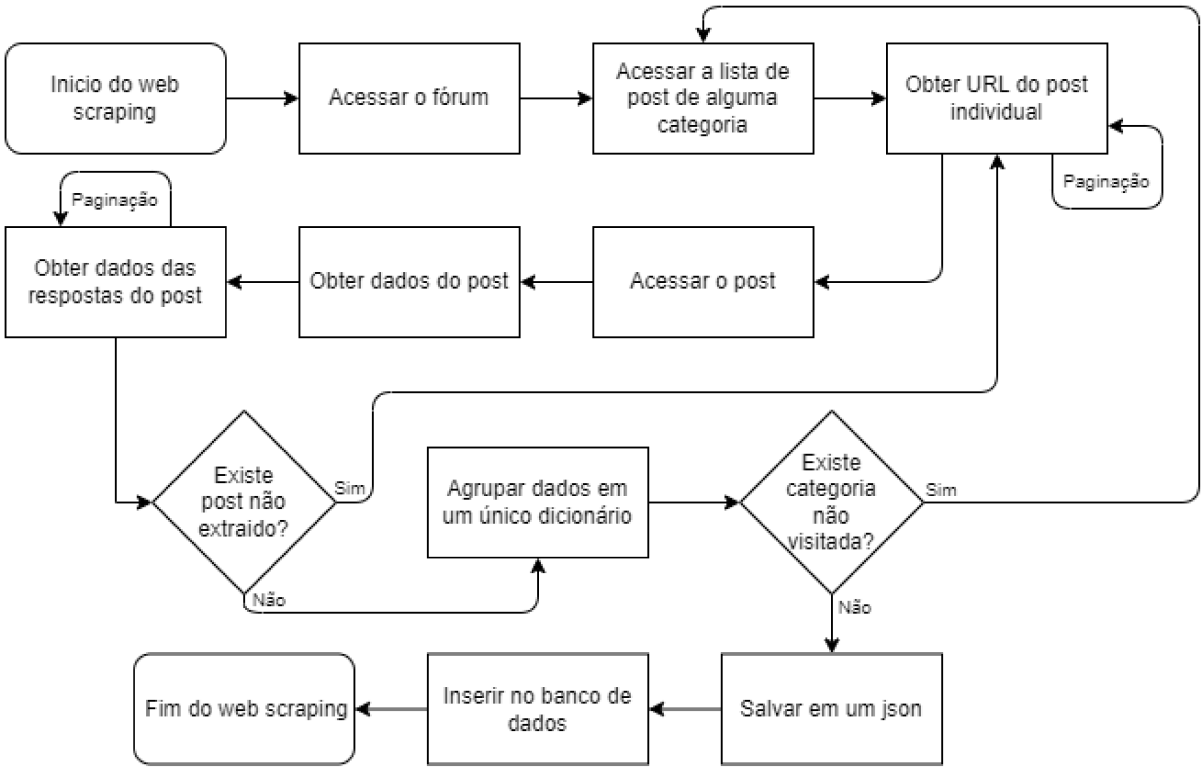

Figura 5 – Fluxograma geral dos *web scrapers* Fonte: Do Autor.

ser redirecionado o acesso para alguma categoria seguindo previamente uma lista ou dicionário. Após o acesso, deve ser obtido o código-fonte em *HTML* do site totalmente carregado. Com este código-fonte, é possível obter a *URL* de cada *post* individual naquela página, utilizando uma estrutura de laço para abranger todas as publicações. Além disso, é importante notar que pode haver mais de uma página na lista de *posts*, portanto, o *script* também deve ser capaz de lidar com a paginação completa da categoria.

Com o *link* de cada *post*, o código deve acessar individualmente cada publicação naquela categoria, obter o *HTML* da página correspondente e extrair todos os dados, incluindo informações sobre o publicador e quaisquer respostas existentes. É importante ressaltar que pode haver mais de uma página de respostas para um *post*, portanto, o *script* também deve lidar com a paginação, se necessário. Após concluir a obtenção dos dados, o código deve armazenar as informações obtidas em um único dicionário.

Ao término de toda a lista de *posts* de uma categoria, o *script* deve verificar se existe outra categoria a ser extraída, como pode ser observado na Figura 5. Se existir, o *script* deve repetir estes passos com essa outra categoria. Se não existirem mais categorias a serem extraídas, o código deve salvar todas as informações armazenadas no dicionário em um arquivo *JSON*, terminando sua execução.

3.4.3 Desenvolvendo o código em *Python*

Dada essa explicação geral sobre como os *web scrapers* foram construídos, agora será apresentado o script desenvolvido, utilizando o código próprio do fórum *hackonology* como exemplo. Na Figura 6 é possível compreender mais claramente a estrutura do código a ser apresentado.

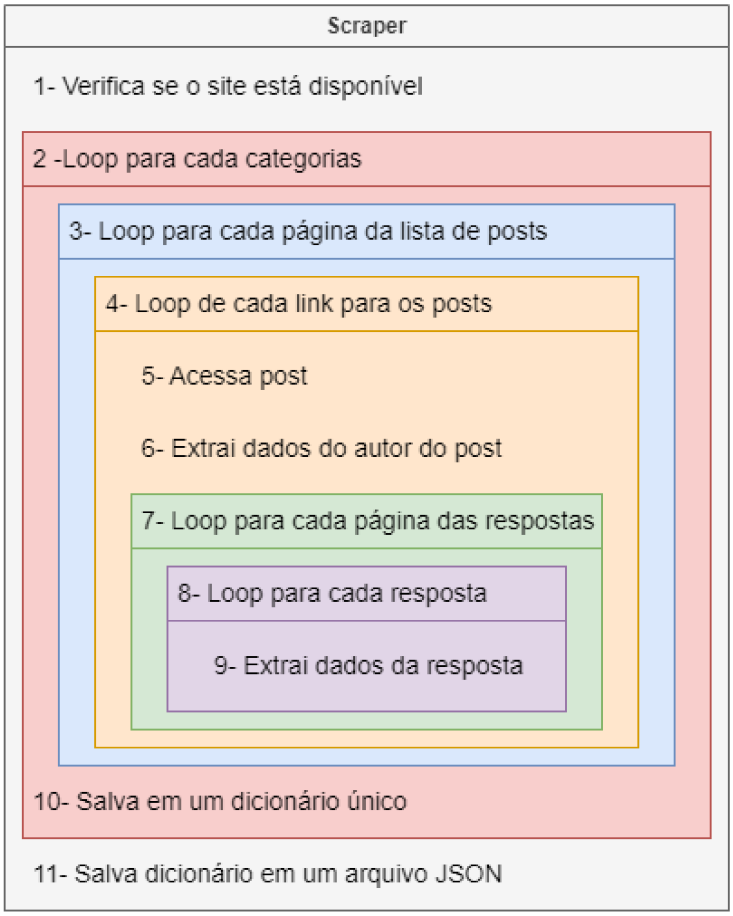

Figura 6 – Estrutura geral do código dos *web scrapers*. Fonte: Do Autor.

Logo ao iniciar o código, como é possível observar no item 1 da Figura 6, é verificado se o site está disponível da seguinte maneira:

```
headers = { "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64;
   x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0
   Safari/537.36" }
url = "https://hackonology.com/forum/"
site = requests.get(url, headers=headers)
if site.status_code != 200:
    return(f"Status invalido: {site.status_code}")
```

Foi empregado o cabeçalho *"User-agent"* para emular uma solicitação convencional semelhante às realizadas por navegadores padrão. Além disso, a função *requests.get* foi utilizada para efetuar a requisição ao site. Caso a solicitação seja bem-sucedida, o fluxo do código continua. Foi empregado um dicionário para mapear as categorias de cada site, onde a chave representa o *FID* e o valor corresponde ao nome da categoria.

```
categorias = {
        # Fid: Nome da categoria
        14: "Hacking e cracking - Advanced Hacking",
        6: "Tech - Configuration Scripts",
        124: "Cyber Security - Cyber Security Tips",
        139: "Cyber Security - Training and Tutorials",
        ...
    }
```

O *FID* é um valor numérico ou *string* associado a cada categoria no URL do fórum, como nos exemplos abaixo:

- https://hackonology.com/forum/forumdisplay.php?fid=81
- https://nulledbb.com/forum-Computing
- https://legitcarders.ws/forums/hacking-guides.24/
- https://niflheim.world/forums/internet-security.43/

Se tudo ocorrer com sucesso, o código deve acessar a página da categoria, obter e processar o *HTML* do site, buscar os links para os *posts* individuais e, se houver, realizar a paginação. Utilizando as análises do código-fonte, é observado que, para o fórum *Hackonology*, é utilizado a classe *'pagination'* para toda a estrutura de paginação e a classe *'pages'* para as páginas individuais. Assim será obtido o valor da última página disponível, ou 1 caso não exista a paginação.

```
for fid, cat in categorias.items():
    url = f"https://hackonology.com/forum/forumdisplay.php?fid={
       fid}"
    site = requests.get(url, headers=headers)
    soup = BeautifulSoup(site.content, "html.parser")

    page_class = soup.find("div",class_=re.compile("pagination"))
    if page_class is None:
```

```
        page = 1
    else:
        page = pagina.find("span", class_= re.compile("pages"))
        page = page.get_text()

    for i in range(1, int(page) + 1):
        url_page = f"https://hackonology.com/forum/forumdisplay.
           php?fid={fid}&page={i}"
        site = requests.get(url_page, headers=headers)
        soup = BeautifulSoup(site.content, "html.parser")
```

Ainda com a análise do código-fonte, é possível determinar que, para o fórum *Hackonology*, os nomes das classes utilizadas são:

- Cada contêiner da lista de *posts* visto na Figura 3 chama *inline_row*;
- O link da publicação nesse contêiner chama *subject_new*;
- Cada conjunto de publicação ou resposta visto na Figura 4 chama *post classic*, em que o primeiro é a própria publicação;
- O nome do autor chama *hf_member*;
- A data da publicação chama *post_date*;
- O conteúdo da publicação ou da resposta chama *post_body scaleimages*.

Conforme evidenciado nos itens 4, 5 e 6 da Figura 6, será realizada uma iteração para cada link de *post* obtido. Posteriormente, cada link será acessado e as informações da publicação serão extraídas, desta forma:

```
foruns = soup.find_all("tr", class_=re.compile("inline_row"))
for forum in foruns:

    links = forum.find("span", class_=re.compile("subject_new"))
    link = links.find("a")["href"]
    url_post = f"https://hackonology.com/forum/{link}"
    site_post = requests.get(url_post, headers=headers)
    soup_post = BeautifulSoup(site_post.content, "html.parser")

    posts = soup_cont.find_all("div", class_=re.compile("post
       classic"))
```

```
    for index, post in enumerate(posts):
        if index == 0:
            titulo = links.get_text()
            autor = post.find("span", class_=re.compile("
               hf_member")).get_text()
            data = post.find("span", class_=re.compile("post_date
               ")).get_text()
            conteudo = post.find("div", class_=re.compile("
               post_body scaleimages")).get_text()
```

É importante notar que, no exemplo em questão, tanto a publicação quanto às respostas tem o mesmo nome de classe, já realizando o item 8 da Figura 6 nessa mesma página. Para possibilitar a paginação, caso seja necessário, e extrair os dados de resposta, foi feito da seguinte maneira:

```
if index == 0:
    # Extracao de dados da publicacao
else:
    autor_resp = post.find("span", class_=re.compile("hf_member")
       ).get_text()
    data_resp = post.find("span", class_=re.compile("post_date"))
       .get_text()
    conteudo_resp = post.find("div", class_=re.compile("post_body
        scaleimages")).get_text()

page_comment = soup_post.find("div", class_=re.compile("
   pagination"))
if page_comment:
    page_com = page_comment.find("span", class_=re.compile("pages
       ")).get_text()
    for j in range(2, (int(page_com) + 1)):
        url_coment = f"https://hackonology.com/forum/{link}&page
           ={j}"
        site_coment = requests.get(url_coment, headers=headers)
        soup_coment = BeautifulSoup(site_coment.content, "html.
           parser")
        # Repete o codigo
        # (loop para cada comentario e para a extracao de dados)
```

Após a obtenção dos dados, são consolidados em um único dicionário. Ao final do

programa, este dicionário é então armazenado em um arquivo *JSON* através do seguinte método:

```
respostas = []
resposta = {
    "autor": autor_resp,
    "data": data_resp,
    "conteudo": conteudo_resp,
    }
respostas.append(resposta)

post = {
    "titulo": titulo,
    "autor": autor,
     "categoria": cat,
    "data": data,
    "conteudo": conteudo,
     "respostas": respostas,
    }
dict_forum[str(ID) + "-" + str(fid)] = post
dicionario.update(dict_forum)

with open("hackonology.json", "w", encoding="utf-8") as arquivo:
    json.dump(dicionario, arquivo, ensure_ascii=False)
```

Por fim, é importante explicar que cada resposta é adicionada a uma lista de respostas, que por sua vez é anexada ao próprio *post*. Cada publicação é identificada de forma única usando a *string 'str(ID) + '-' + str(fid)*', onde o *ID* é uma sequência numérica que inicia em 1. No final, todos os *posts* são anexados a um único dicionário e este é então inserido no arquivo *JSON*.

3.5 Melhorando a eficiência

Nesta seção, foram exploradas técnicas e práticas que puderam ser empregadas para aprimorar a eficiência de *web scrapers* criados. Estudando diferentes técnicas, a que mais se destacou foi a utilização de *multithreading*. *Multithreading* é uma técnica de programação que permite que um processo ou programa execute várias tarefas simultaneamente. Cada tarefa é chamada de *thread* e é uma sequência independente de execução dentro de um programa (TULLSEN; EGGERS; LEVY, 1995).

Através da implementação dessa prática, foi viável reestruturar o conceito fundamental do trabalho, simplificando o processo de desenvolvimento e proporcionando uma compreensão mais clara do funcionamento real dos códigos, como ilustrado na Figura 7.

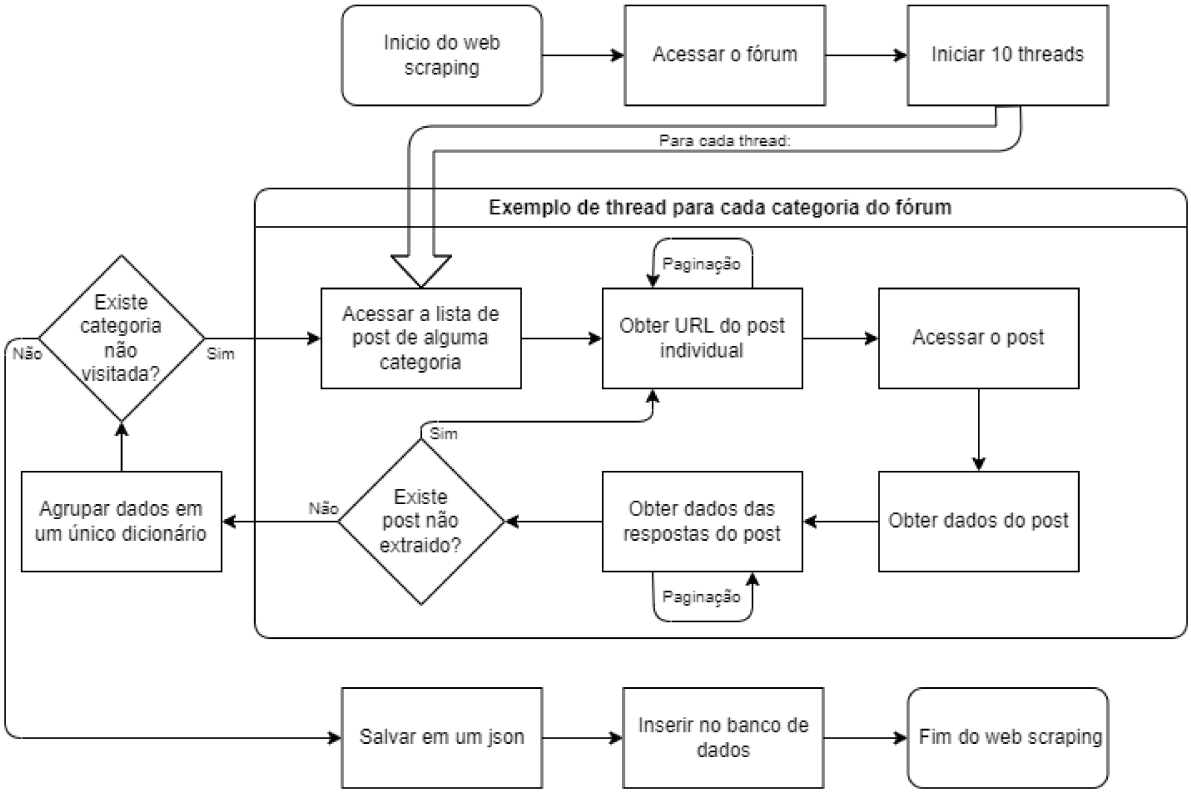

Figura 7 – Fluxograma completo dos *web scrapers*. Fonte: Do Autor.

Para evitar sobrecarregar o processamento da máquina e não comprometer o funcionamento dos sites, foi crucial selecionar uma quantidade adequada de *threads*. Após diversos testes e investigações, constatou-se que o uso de 10 *threads* foi o suficiente para aprimorar o desempenho do código, sem causar quaisquer prejuízos. Na Seção 4.1 será demonstrado os testes realizados que identificaram o numero apropriado de *threads*.

Com isso, foi possível modificar o código, implementado o *multithreading* da seguinte maneira:

```
threads = []

def thread_function():
    while categorias:
        with lock:
            fid, cat = categorias.popitem()
            processar_categoria(fid, cat)
```

```
lock = threading.Lock()
for _ in range(10):
    thread = threading.Thread(target=thread_function)
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()
```

Todos os códigos anteriores foram inseridos na função processar_categoria que recebe como argumentos o *fid* e o nome da categoria. Com essa mudança, foi definida a função *thread_function* que contém uma iteração para processar categorias enquanto houver disponíveis. Um objeto de bloqueio do tipo *lock* é utilizado para garantir acesso seguro à lista de categorias.

No *loop*, cada *thread* obtém uma categoria do dicionário categorias, processa-a e, ao final, aguarda até que todas as *threads* tenham terminado sua execução antes de salvá-las no *JSON*. O programa cria e inicia 10 *threads*, cada uma executando a função *thread_function*. O uso do *lock* assegura que somente uma *thread* por vez pode acessar a lista de categorias.

3.6 Inserção no banco de dados

Com os dados já obtidos no *JSON*, bastou desenvolver um script para inseri-los no banco de dados. No entanto, também foi necessário criar a infraestrutura do banco de dados para possibilitar o armazenamento dos dados extraídos.

Para cada fórum foram criadas duas tabelas distintas. A Tabela 1 representa a estrutura criada para armazenar as publicações. Já a Tabela 2 reflete a arquitetura utilizada para registrar as respostas. Com isso, tem-se um total de oito tabelas, sendo quatro destinadas às publicações e outras quatro às respostas.

Com as tabelas já criadas, foi possível desenvolver o script para inserir os dados extraídos no banco de dados. Desta forma, usando como exemplo o script do fórum *Hackonology*, obtém-se:

```
conn = psycopg2.connect(**db_params)
cur = conn.cursor()

cur.execute("DELETE FROM hackonology_answer")
cur.execute("DELETE FROM hackonology_post")
```

Publicações					
ID	Título	Autor	Categoria	Data	Conteúdo
Serial	Varchar	Varchar	Varchar	Timestamp	Text

Tabela 1 – Estrutura do banco de dados para as publicações.

Respostas				
ID	Post_ID	Autor	Data	Conteúdo
Serial	Integer	Varchar	Timestamp	Text

Tabela 2 – Estrutura do banco de dados para as respostas.

```
cur.execute("ALTER SEQUENCE hackonology_answer_id_seq RESTART
   WITH 1")
cur.execute("ALTER SEQUENCE hackonology_post_id_seq RESTART WITH
   1")
```

O processo de inserção de dados começa ao estabelecer uma conexão com o banco de dados, que utiliza a estrutura presente em *db_params*. Após estabelecer a conexão, é crucial limpar qualquer informação preexistente em ambas as tabelas, assim como redefinir o contador serial do ID para 1. Com estes passos de preparação concluídos, torna-se viável realizar a inserção dos dados nas tabelas da seguinte maneira:

```
with open("hackonology.json", "r", encoding="utf-8") as file:
    seu_json = json.load(file)

    for post_id, post_data in seu_json.items():
        titulo = post_data["titulo"]
        autor = post_data["autor"]
        categoria = post_data["categoria"]
        data = post_data["data"]
        conteudo = post_data["conteudo"]

        cur.execute( "INSERT INTO hackonology_post (title, author
           , category, created_at, body)  VALUES (%s, %s, %s, %s,
           %s) RETURNING id",
        (titulo, autor, categoria, data, conteudo),)
        post_id_db = cur.fetchone()[0]

        respostas = post_data["respostas"]
        if respostas:
```

```
            for resposta in respostas:
                autor_resposta = resposta["autor"]
                data_resposta = resposta["data"]
                conteudo_resposta = resposta["conteudo"]

                cur.execute( "INSERT INTO hackonology_answer (
                   post_id, author, created_at, answer_body)
                   VALUES (%s, %s, %s, %s)",
                    (
                        post_id_db,
                        autor_resposta,
                        data_resposta,
                        conteudo_resposta,
                    ),
                )

conn.commit()
conn.close()
```

Os dados são extraídos de seu *JSON* correspondente e organizados conforme suas chaves. Posteriormente, cada publicação é inserida no banco de dados através do comando *"INSERT INTO hackonology_post ... RETURNING id"*, onde o identificador é armazenado.

Se houver respostas associadas a essa publicação, cada uma delas é inserida utilizando o comando *"INSERT INTO hackonology_answer ..."*, usando o identificador da publicação correspondente. Se todas as operações forem concluídas sem erros, as inserções são confirmadas através do *commit*, os dados são salvos e, em seguida, a conexão com o banco é encerrada.

4 Resultados

Neste capítulo, serão apresentados os resultados obtidos com a execução dos *web scrapers* individuais para cada fórum. Adicionalmente, é relevante ressaltar que os *scripts* completos de cada *web scraper* foram configurados para serem executados uma vez por semana, abordando um fórum por dia, a fim de evitar sobrecarga na máquina ou no site, considerando que a máquina não é exclusivamente dedicada a este trabalho. Além disso, os experimentos foram conduzidos em uma estação de trabalho equipada com:

- CPU: Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz;
- Memória RAM: 16GB, DDR4, 2133 MHz;
- Sistema Operacional: Ubuntu 20.04.4 LTS;
- Versão do *Python*: v3.8.10.

Inicialmente na Seção 4.1 será detalhado os testes que foram realizados para a escolha da quantidade ideal de *threads* utilizadas. Na Seção 4.2, serão apresentados os resultados gerais da execução dos *web scrapers*, destacando como é parte do arquivo *JSON*. Já nas Seções 4.3, 4.4, 4.5 e 4.6, serão detalhados os resultados específicos de cada *web scraper* para os respectivos fóruns selecionados.

Por fim, todos os elementos desenvolvidos para a realização desta monografia, incluindo códigos, planilhas, tabelas, imagens, *JSONs* e arquivos de teste e demais informações, foram disponibilizados pelo autor e podem ser encontrados em (LABAKI, 2023).

4.1 Testes do *Multithreading*

Foram conduzidos vários testes para determinar a quantidade ideal de *threads* para o *web scraper*, evitando impactos negativos no site ou na máquina. A Tabela 3 demonstra o resultado desses testes, utilizando o código do *Legit Carders*, demonstrando o tempo total de execução, e a média de consumo de memória e processamento da máquina.

Analisando os resultados, percebe-se que a utilização de 10 *threads* é a opção ideal, pois apresenta um tempo de execução significativamente menor sem ultrapassar o potencial da máquina. Por outro lado, ao utilizar 15 *threads*, o ganho de tempo não é tão expressivo, chegando quase ao limite médio de consumo de memória e, principalmente, de processamento. É importante destacar que não foi viável conduzir um teste de estresse nos fóruns, pois poderia ser prejudicial ao site, além de violar as suas políticas.

Testes de *Threads* para o *Legit Carders*			
Threads utilizadas	Tempo de execução	Memória	Processamento
Sem *threads*	1296,98 segundos	32%	39%
5	892,42 segundos	49%	54%
10	566,69 segundos	61%	60%
15	549,21 segundos	85%	92%

Tabela 3 – Testes de *Threads* para o fórum *Legit Carders*.

4.2 Resultados Gerais

Ao executar os *web scrapers*, os dados são inicialmente salvos em um arquivo *JSON* antes de serem inseridos no banco de dados, como exemplificado na Figura 8.

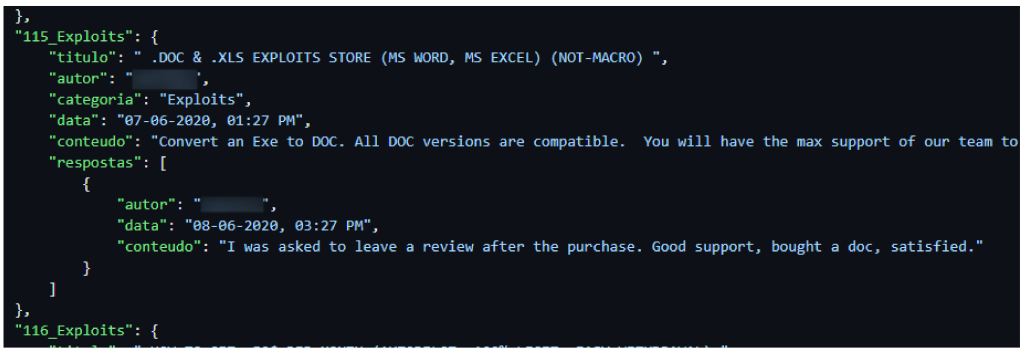

Figura 8 – Parte do *JSON* resultante do fórum *NulledBB*.

Após a execução completa de todos os códigos, foram obtidas cerca de 210 mil linhas de dados extraídos. Esses dados compreendem aproximadamente 35 mil publicações e 175 mil respostas, sendo que o processo de extração levou quase quatro horas para ser concluído.

4.3 *Legit Carders*

O *web scraper* utilizado no fórum *Legit Carders* obteve resultados notáveis em sua operação de extração. Como evidenciado na Tabela 4, foram utilizadas apenas 21 das 30 categorias disponíveis no site. O código teve um tempo de execução de 9,4 minutos, o que resultou na obtenção de 7.708 dados de publicações e respostas para este fórum.

Legit Carders				
Categorias	Tempo de execução	Publicações	Respostas	Taxa de extração
21 / 30	566,69 segundos	2.560	5.148	13,6

Tabela 4 – Resultados das extrações para o fórum *Legit Carders*.

Ao aprofundar a análise, torna-se viável calcular a taxa de extração de dados ao dividir a quantidade total de publicações e respostas extraídas pelo tempo total gasto, o que resulta na quantidade de informação obtida por segundo. Essa taxa é útil apenas para verificar se os *web scrapers* desenvolvidos estão com desempenhos parecidos, evidenciando possíveis problemas com o código. Para este fórum, a taxa obtida é de aproximadamente 13,6 postagens extraídas por segundo, o que indica uma performance significativa para este site.

4.4 *Hackonology Forums*

Diferentemente dos resultados encontrados durante a coleta do fórum *Legit Carders*, o fórum *Hackonology* não apresentou resultados tão satisfatórios. Ao ser analisado os dados da Tabela 5, é notado que o tempo de execução foi de 15,5 minutos, resultando em 7.074 informações extraídas.

Hackonology Forums				
Categorias	Tempo de execução	Publicações	Respostas	Taxa de extração
73 / 91	933,33 segundos	5.014	2.060	7,5

Tabela 5 – Resultados das extrações para o fórum *Hackonology Forums*.

Ao ser comparado com os resultados obtidos no *Legit Carders*, o desempenho do *Hackonology* foi inferior, demonstrando uma taxa de extração de dados de 7,5 por segundo. Isso pode ser atribuído ao elevado número de categorias, ressaltando que as melhorias de desempenho realizadas são mais eficazes em fóruns com um número reduzido de categorias. Mesmo assim, o resultado ainda é satisfatório.

4.5 *Niflheim World*

O resultado obtido pelo *web scraper* do *Niflheim World* foi surpreendente. Dado o foco primário do fórum em tecnologia, praticamente todas as suas categorias foram selecionadas, como indicado na Tabela 6, o que resultou na obtenção impressionante de 54.198 dados de publicações e respostas. No entanto, também é notável que sua performance não foi satisfatória, levando aproximadamente 50,3 minutos. Apesar disso, o resultado ainda é bastante significativo.

Embora tenha uma duração prolongado, o *web scraper* apresentou uma impressionante taxa de extração de dados de 17,9 por segundo, destacando-se como o fórum de melhor desempenho entre todos. Uma análise mais detalhada permite concluir que o fó-

Niflheim World				
Categorias	Tempo de execução	Publicações	Respostas	Taxa de extração
54 / 55	3.018,83 segundos	10.435	43.763	17,9

Tabela 6 – Resultados das extrações para o fórum *Niflheim World.*

rum é extremamente ativo, evidenciado pelo elevado número de respostas em comparação com o número de publicações.

4.6 *NulledBB*

O *NulledBB* é o maior fórum entre os quatro selecionados, contando com 141.458 dados de publicações e respostas extraídos de apenas 16 das suas 77 categorias, conforme demonstrado na Tabela 7. O tempo de execução de 2,6 horas é compreensível, dada a abundância de informações coletadas.

NulledBB				
Categorias	Tempo de execução	Publicações	Respostas	Taxa de extração
16 / 77	9.509,79 segundos	17.545	123.913	14,8

Tabela 7 – Resultados das extrações para o fórum *NulledBB.*

Sua performance é consideravelmente boa, alcançando uma taxa de extração de dados de aproximadamente 14,8 por segundo. Embora não seja o melhor em desempenho, a quantidade de dados extraídos ainda é a sua principal característica.

5 Conclusão

O principal objetivo deste trabalho era desenvolver *web scrapers* para um grupo específico de fóruns relacionados à segurança da informação, visando obter uma quantidade significativa de dados para análise posterior. Ao longo do processo, tornou-se evidente a dificuldade em localizar links diversos de fóruns relevantes para a segurança da informação, e mais ainda em identificar aqueles que atendem aos critérios de seleção.

Apesar disso, devido à semelhança estrutural dos fóruns selecionados, a elaboração dos códigos para os *web scrapers* individuais dos sites tornou-se mais direta. Além disso, a implementação de *multithreading* se revelou crucial para otimizar a eficiência dos *web scrapers*, o que resultou em um tempo de execução do código aceitável.

Com o *JSON* gerado, a inserção dos dados em qualquer categoria de banco de dados, seja ele relacional ou não, torna-se extremamente simples, proporcionando uma maior flexibilidade para adaptação a uma variedade de trabalhos distintos. Em suma, a raspagem dos dados resultou em cerca de 210 mil linhas, compostas por 35 mil publicações e 175 mil respostas. Essa vasta quantidade de informações está diretamente ligada à tecnologia em geral, e o processo de obtenção levou pouco menos de quatro horas para ser concluído.

Para trabalhos futuros, sugere-se utilizar as informações obtidas nos diversos fóruns para criar mecanismos de detecção de incidentes de segurança, implicando em uma análise que deve utilizar diversas técnicas de mineração de dados, visando identificar possíveis ameaças à segurança, como atividades criminosas, ciberataques e espionagem, entre outras. Além disso, é sugerido desenvolver *web scrapers* para outras fontes de dados, como redes sociais ou até mesmo fóruns na *dark web*, visando ampliar a quantidade de informações provenientes de diversas fontes, para estabelecer assim uma sólida base de dados para análises mais abrangentes.

Referências

BROWN, L.; STALLINGS, W. **Segurança de computadores: princípios e práticas**. [S.l.]: Elsevier Brasil, 2017. v. 2. Citado na página 12.

BUDIARTI, R. P. N.; WIDYATMOKO, N.; HARIADI, M.; PURNOMO, M. H. Web scraping for automated water quality monitoring system: A case study of pdam surabaya. In: **2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)**. [S.l.: s.n.], 2016. p. 641–648. Citado na página 20.

CHAULAGAIN, R. S.; PANDEY, S.; BASNET, S. R.; SHAKYA, S. Cloud based web scraping for big data applications. In: **2017 IEEE International Conference on Smart Cloud (SmartCloud)**. [S.l.: s.n.], 2017. p. 138–143. Citado na página 20.

CRAIGEN, D.; DIAKUN-THIBAULT, N.; PURSE, R. Defining cybersecurity. **Technology Innovation Management Review**, v. 4, p. 13–21, 10 2014. Citado na página 9.

GLEZ-PEñA, D.; LOURENçO, A.; LóPEZ-FERNáNDEZ, H.; REBOIRO-JATO, M.; FDEZ-RIVEROLA, F. Web scraping technologies in an API world. **Briefings in Bioinformatics**, v. 15, n. 5, p. 788–797, 04 2013. ISSN 1467-5463. Disponível em: <https://doi.org/10.1093/bib/bbt026>. Citado na página 13.

HAN, S.; ANDERSON, C. K. Web scraping for hospitality research: Overview, opportunities, and implications. **Cornell Hospitality Quarterly**, v. 62, n. 1, p. 89–104, 2021. Citado 2 vezes nas páginas 19 e 20.

HOEK. **Hacker forums**. 2023. Disponível em: <https://0ut3r.space/2020/09/25/hacker-forums/>. Citado na página 22.

LABAKI, A. do P. **Monografia: Web Scrapers para coleta de dados em fóruns de segurança**. 2023. Disponível em: <https://github.com/rsmiani/deteccao-incidentes/tree/main/scraperSurface>. Acesso em: 10 nov 2023. Citado na página 38.

LANDERS, R. N.; BRUSSO, R. C.; CAVANAUGH, K. J.; COLLMUS, A. B. A primer on theory-driven web scraping: Automatic extraction of big data from the internet for use in psychological research. **Psychological methods**, American Psychological Association, v. 21, n. 4, p. 475, 2016. Citado na página 11.

MAHMOOD, T.; AFZAL, U. Security analytics: Big data analytics for cybersecurity: A review of trends, techniques and tools. In: . [S.l.: s.n.], 2013. p. 129–134. ISBN 978-1-4799-1288-9. Citado na página 9.

MANDIA, K.; PROSISE, C. **Incident response: investigating computer crime**. [S.l.]: McGraw-Hill, Inc., 2001. Citado na página 13.

MAYER-SCHÖNBERGER, V.; CUKIER, K. **Big data: A revolution that will transform how we live, work, and think**. [S.l.]: Houghton Mifflin Harcourt, 2013. Citado na página 9.

MEIER, A.; KAUFMANN, M. **SQL & NoSQL databases**. [S.l.]: Springer, 2019. Citado na página 16.

MITCHELL, R. **Web scraping with Python: Collecting more data from the modern web**. [S.l.]: "O'Reilly Media, Inc.", 2018. Citado 4 vezes nas páginas 10, 11, 14 e 15.

MITNICK, K. **Ghost in the wires: My adventures as the world's most wanted hacker**. [S.l.]: Hachette UK, 2011. Citado na página 16.

MOTOYAMA, M.; MCCOY, D.; LEVCHENKO, K.; SAVAGE, S.; VOELKER, G. M. An analysis of underground forums. In: **Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference**. New York, NY, USA: Association for Computing Machinery, 2011. (IMC '11), p. 71–80. ISBN 9781450310130. Disponível em: <https://doi.org/10.1145/2068816.2068824>. Citado 2 vezes nas páginas 9 e 11.

PFLEEGER, C. P.; PFLEEGER, S. L. **Analyzing computer security: A threat/vulnerability/countermeasure approach**. [S.l.]: Prentice Hall Professional, 2012. Citado na página 13.

RAWAT, A. A review on python programming. **International Journal of Research in Engineering, Science and Management**, v. 3, n. 12, p. 8–11, Dec. 2020. Disponível em: <https://journal.ijresm.com/index.php/ijresm/article/view/395>. Citado na página 17.

ROSSUM, G. van. Python programming language. In: . Santa Clara, CA: USENIX Association, 2007. Citado na página 17.

SCHULTZ, E.; SHUMWAY, R. **Incident response: A strategic guide to handling system and network security breaches**. [S.l.]: Sams, 2001. Citado na página 13.

SCHäFER, M.; FUCHS, M.; STROHMEIER, M.; ENGEL, M.; LIECHTI, M.; LENDERS, V. Blackwidow: Monitoring the dark web for cyber security information. In: **2019 11th International Conference on Cyber Conflict (CyCon)**. [S.l.: s.n.], 2019. v. 900, p. 1–21. Citado na página 19.

TEIXEIRA, L. J. R. Investigação da darknet como fonte de dados para ciberataques. Universidade Federal de Uberlândia, 2022. Disponível em: <https://repositorio.ufu.br/handle/123456789/34740>. Citado na página 19.

THELWALL, M. A web crawler design for data mining. **Journal of Information Science**, v. 27, n. 5, p. 319–325, 2001. Disponível em: <https://doi.org/10.1177/016555150102700503>. Citado 2 vezes nas páginas 10 e 14.

TULLSEN, D.; EGGERS, S.; LEVY, H. Simultaneous multithreading: Maximizing on-chip parallelism. In: **Proceedings 22nd Annual International Symposium on Computer Architecture**. [S.l.: s.n.], 1995. p. 392–403. Citado na página 33.

VACCA, J. **Computer and information security handbook**. [S.l.]: Newnes, 2012. Citado na página 12.

VARGIU, E.; URRU, M. Exploiting web scraping in a collaborative filtering-based approach to web advertising. **Artificial Intelligence Research**, v. 2, 01 2013. Citado na página 11.

WAGNER, T. D.; MAHBUB, K.; PALOMAR, E.; ABDALLAH, A. E. Cyber threat intelligence sharing: Survey and research directions. **Computers Security**, v. 87, p. 101589, 2019. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S016740481830467X>. Citado na página 15.

WILLIAMS, P.; WOODWARD, A. Cybersecurity vulnerabilities in medical devices: A complex environment and multifaceted problem. **Medical devices (Auckland, N.Z.)**, v. 8, p. 305–16, 08 2015. Citado na página 9.