MASARYK
UNIVERSITY

FACULTY OF INFORMATICS

Service identification system for network traffic analysis

Master's Thesis

BC. PETR ADAMEC

Brno, Spring 2024

MASARYK UNIVERSITY

FACULTY OF INFORMATICS

Service identification system for network traffic analysis

Master's Thesis

BC. PETR ADAMEC

Advisor: RNDr. Martin Laštovička, Ph.D.

Department of Computer Systems and Communications

Brno, Spring 2024

FACULTAS ARTIS INFORMATICÆ • UNIVERSITAS MASARYKIANA •

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source. During the preparation of the thesis I use Grammarly for grammar checking I declare that I used is tool in accordance with the principles of academic integrity. I checked the content and took full responsibility for it

bc. Petr Adamec

Advisor: RNDr. Martin Laštovička, Ph.D.

Acknowledgements

I would like to thank my advisor, RNDr. Martin Laštovička, for his guidance and patience. Special thanks to my consultant, Ing. Martin Holkovič, Ph.D., for his valuable advice and insights. Lastly, I want to express my gratitude to my family and Anna for their support during my studies.

Abstract

The thesis solve problem of gathering and providing data about services or applications for network traffic analysis. This work implements system which periodically store data from multiple data sources, handles data anomalies including collisions and large changes and export data for furher usage in Flowmon Anomaly Detection System. Part of the solution is data source which using web scraping techniques to obtain data.

Keywords

IPFIX, IP Flow, Service, Application, Progress Software, Cybersecurity

Contents

1 Introduction **1**

2 Detection of Anomalies in IP Flows **2**
2.1 Passive Monitoring of the Network 2
2.2 From Packets to Anomalies 3
2.3 Intrusion Detection System 7
2.4 Flowmon Anomaly Detection System 8

3 Network Applications and Services **10**
3.1 The Selection Criteria for Applications 10
3.2 Selected Applications and Services 11
3.3 Data Publication of Application 13

4 Web Scraping **15**
4.1 Web Scraping Process 15
4.2 Applications of Web Scraping 16
4.3 Web Scraper Structure 17
4.4 The Robots Exclusion Protocol 18

5 Conceptual Design of Application Database **19**
5.1 Cooperation with Progress Software 19
5.2 Requirements 20
5.3 High Level Overview 21
5.4 Use Cases 24
5.5 Data Anomalies 26
5.6 Example Run of Core Part 29

6 Logical Design of Application Database **31**
6.1 Data Structure 31
6.2 Code Structure 32
6.3 Tickets and Rules 33
6.4 Final State of Application Database 36

7 Implementation of Scraper **38**
7.1 Requirements 38
7.2 Structure of Scraper 39

7.3 Deployment and Validation 40

8 Conclusion **42**

A Electronic attachments **43**

B Diagrams **44**

Bibliography **47**

1 Introduction

Network monitoring is a crucial part of the company's security. Nowadays, company security is handled by security operators responsible for many security areas. These areas are covered mainly by multiple tools, which generate vast amounts of data that operators must process. Security operators became overwhelmed by the amount of information, and these situations bring us a new challenge: How to extend the readability of data from multiple security tools.

The thesis focuses on one of these security tools called Flowmon Anomaly Detection System (Flowmon ADS), which Progress Software is developing. Flowmon ADS detects anomalies in local traffic, which could indicate security problems, like dictionary attacks. Every anomaly has assigned the source and destination IP address of the traffic. IP addresses could provide additional helpful information like the name of the application behind the address and help security operators resolve anomalies faster. Flowmon ADS provides an application mapping feature, which solves the challenge mentioned above by mapping application information to source or destination IP addresses to improve the readability of the anomalies.

Application mapping feature uses multiple data sources, which provide information about applications and their IP address. The thesis aims to create and implement a system to merge data from various data sources and export them for further usage in Flowmon ADS. The thesis solution includes creating a new data source focusing on specific applications. The thesis solution was consulted with developers from Progress Software, who participated in the requirement part.

The second chapter of the thesis describes creating an event that represents an anomaly in Flowmon ADS. The third chapter continues with a description of selected applications, where we describe the importance of selected applications in local network traffic. The fourth chapter describes the web scraping technique, which is then used in our solution. The following chapters focus on implementation, whereas the fifth chapter describes concepts and provides a high-level overview of the thesis solution. The sixth and seventh chapters describe our implemented software more technically.

2 Detection of Anomalies in IP Flows

Anomalies in network traffic can indicate bad network behavior that we should be aware of. Anomalies can indicate a security vulnerability, a misconfiguration of a particular network service, or even an intrusion into the local network.

This chapter provides an overview of how network anomalies are detected in Flowmon ADS. It begins by explaining the process of gathering data from the network using a passive approach. The second section covers the transformation process from gathered data (packets) to anomalies. The fourth and fifth sections then describe how IP flow data is utilized. The third section defines the Intrusion Detection System, and the last section then describes the commercial example of an Intrusion Detection System that uses anomaly detection.

2.1 Passive Monitoring of the Network

Passive network monitoring is based on the observing of the network, this observing minimally affect network traffic. Passive monitoring could be divided into two approaches.

Packet Capture

This approach captures complete packets. Packet capturing provides a whole captured payload like the HTML page, images, and text messages. It generally provides the most insight into the network traffic. However, packet capturing can be expensive as it requires high-end hardware to handle large amounts of traffic, which can be computationally intensive. Nowadays, large amounts of traffic are encrypted, and packet capturing cannot easily decrypt captured traffic, which leads to the captured traffic being unreadable and provides a small amount of information. [1]

Flow Monitoring

Another approach is flow monitoring, where packets are aggregated into flows and exported. Flows contain packet header fields like source

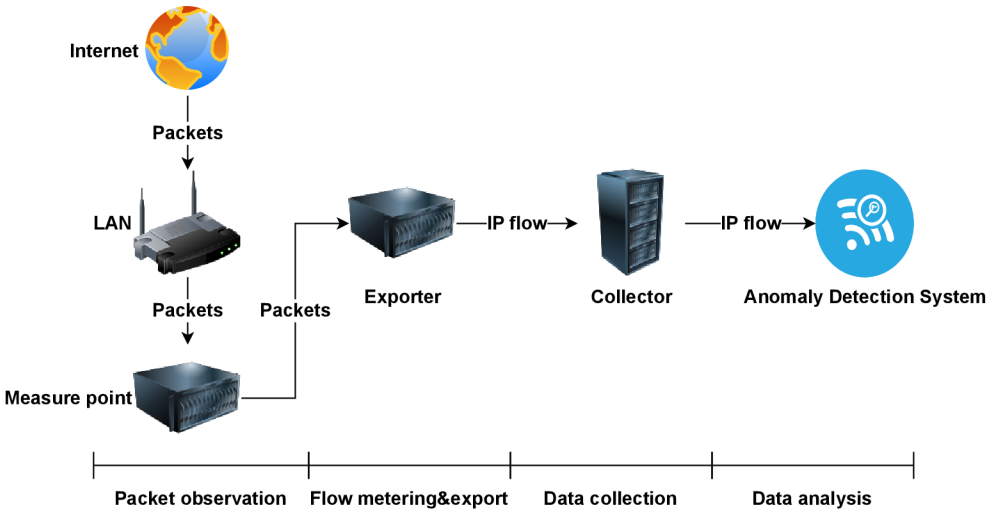

Figure 2.1: From the Internet to Anomaly [1]

and destination IP address. Flow is defined in Section 2.2. The flow monitoring approach provides multiple advantages:

- **Price**: Most regular network devices support flow exporting. With devices supporting flow exporting, flow monitoring could be used without any additional infrastructure for capturing. [2]
- **Data Reduction**: Flow monitoring achieves significant data reduction, typically around 1/2000 of the original volume [1].
- **Traffic encryption**: Flow monitoring could still gather information from encrypted packets, like the number of packets, size of packets, destination or source addresses, and their ports, which, for example, could be helpful in the detection of SSH brute force attacks. [1]

2.2 From Packets to Anomalies

In this section we describe how is packet observed, then transformed into flow, collected and analyzed. The whole process is depicted in the Figure 2.1

Packet Observation & Preprocessing

Packet observation is a first part of flow monitoring that captures packets from a Observation Point. *'An Observation Point is a location in the network where packets can be observed.'* [3] Examples of Observation Points could be a port on the router or a set of router interfaces. Packet observation continues with packet preprocessing. Packet preprocessing consists of mutliple steps:

- **Timestamping**: Every packet gets a accurate timestamp. A critical area for timestamping is multipoint measurements: the ability to correlate events at one link in the network relative to observations at a different link. [4]

- **Truncation**: Select only specific bytes of packet that fit into preconfigured snapshot length. For example, flow exporters mostly select only packet header fields and ignore packet payloads. This step is optional. [1]

- **Packet Sampling & Filtering**: This is the last optional step of packet preprocessing. These steps reduce the number of captured packets, which will be passed in the next steps of flow monitoring. Packet sampling works by selecting only a sample of data from the captured packets, and a sampling strategy is used to create this sample. For example, the sampling strategy is to select every third packet of captured packets. Filtering is the deterministic selection of packets based on the packet header, payload and properties. [5]

Flow Metering & Export

Flow Metering & Export is the second stage/part, where packets are aggregated into flows. But what is flow? *'A flow is a set of packets passing an Observation Point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties.'* [3] These properties could be source and destination IP addresses, port numbers, or other information from the packet header.

The whole part is divided into these steps:

- **Metering Process**: Every flow consists of multiple fields like source IP address or protocol identifier. Fields that can be exported in IP flow records are named Information Elements. Flows are stored in the flow cache after the aggregation process. *Flow caches are tables within a Metering Process that store information regarding all active network traffic flows*. [1] Flows wait for additional packets, which could be added to flow based on Information Elements. The flow in the flow cache expires after some time or for another reason, like the flow cache is full, and then flows are moved on to the second step. [1]

- **Flow Sampling & Filtering**: This step aims to reduce the number of flows that will be exported. Flows are filtered and sampled by similar techniques like packets, which are described in 2.2

- **Exporting Process**: Flow is exported by the Exporting process, which means that flow is placed into the datagram of flow export protocol and sent to the flow collector. Flow collectors are responsible for the storage and preprocessing of exported flows. [1]

Data Collection

The third stage of flow monitoring is where the flow collector preprocess and stores data from the previous stage. The collector receives flows from one or more flow exporters. It processes the received flows; that is, it parses and stores the flow information. Flows can be optionally aggregated before being stored on the persistent storage. [6]

Flow preprocessing includes these operations:

- **Data Compression**: The possibility of data compression depends on network flow monitoring protocols like NetFlow9 or IPFIX. For example, IPFIX supports compressing using bzip2 or gzip. Data compression increases scalability. [7]

- **Anonymization**: Flow has a significant privacy protection advantage over packets. Flow does not provide a packet payload, which could contain sensitive information. However, it can still

be used to track individual user activity in the network. For example, best practices for flow anonymization are described by CAIDA in [8, 1].

- **Aggregation**: A basic example is flows aggregated by source address; this aggregation results in a time series. Aggregating Flows by source IP address in time series (i.e., with a regular interval) can be used in subsequent heavy-hitter analyses and as a source parameter for statistical anomaly detection techniques. [9]

Data Analysis

The last stage of flow monitoring is where collected data will be used for analysis. This data could be used in these areas :

- **Flow Analysis & Reporting**: This is the most common usage, where we can analyze statistic overview extended with alerts. For example, a statistical overview can show the amount of data transferred between two IP addresses. If we can assign specific services to specific public IP addresses, we can detect outages of some services, which could result in alerts.

- **Threat Detection**: There are two categories of threat detection. The first is detection based on specific behavior like an SSH dictionary attack, where an attacker tries to log in many (e.g., 8000) times per hour with common login credentials. The second type is detection based on source or destination IP address, where one of those IP addresses is, for example, a well-known attacker. In the second example, there is also the possibility of using a service IP address to detect some services frequently used by attackers, such as TOR. Threat detection is handled by the Intrusion Detection System, which is described in section 2.3. [1]

- **Application Performance Monitoring**: It monitors running services and collects metrics like delay, jitter, or response time. These metrics could help detect troubles with performance. [1]

2.3 Intrusion Detection System

To define an Intrusion Detection System (IDS), we must start with the definition of Intrusion Detection from [10] *'Intrusion detection is the process of monitoring the events occurring in a computer system or network, and analyzing them for signs of intrusion'*.

With this definition, we can define IDS. *'The intrusion detection system is the software or hardware system to automate the intrusion detection process'* . [11]

Intrusion detection could be divided into three approaches:

- **Signature-based Detection**: This approach is used to detect threats using a specific pattern or string called a signature; for instance, the IP address of a well-known attacker can be used as a signature. This approach is simple and effective in detecting known attacks. On the other hand, the possibility of Signature-based Detection depends on the quantity and quality of signatures, and it primarily cannot detect new unknown attacks. [10]

- **Anomaly-based Detection**: It detects an anomaly, a deviation from a known or expected behavior. It uses a profile representing the normal behavior of users, hosts, or network connections. Anomaly-based Detection can effectively detect new unforeseen vulnerabilities or threats. On the other hand, it highly depends on the profile's accuracy, making it difficult to detect intrusion at the right time. [11]

- **Stateful Protocol Analysis**: Approach that involves tracing protocol states by pairing protocol requests and responses. This approach is similar to Anomaly-based Detection. The key difference is that Stateful Protocol Analysis depends on profiles of protocols developed by the protocol vendor.

These approaches can be combined to achieve the best possible detection.

2.4 Flowmon Anomaly Detection System

Flowmon Anomaly Detection System (ADS) is a commercial hybrid intrusion detection system; hybrid IDS combines approaches mentioned in the previous section. [12]

Events

Events in Flowmon ADS represent anomalies that happen on the network. An example of a Flowmon event is shown on Figure 2.2. Every event contains these data:

1. Source IP address of the event
2. Destination IP addresses of the event
3. Type and subtype of detection method, which detects this event
4. Detection time, time of first flow, and time of last update of the event

Flowmon ADS events store more information, but for the purpose of this thesis, we only need to know that events contain source and destination IP addresses. [12]

Application Mapping in Flowmon ADS

Part of the thesis solution is gathering data about services and applications. These data enhance events by adding the name of the application or service behind the source or destination IP address. This process is called application mapping and helps users mainly with higher comprehensibility. Most users do not know more than 20 IP addresses, but they know more than 40 application and service names, so they can process events using the application's name and IP address rather than only the IP address. [12]

The application's name could help the user identify unexpected behavior in the local network by detecting usage of internally forbidden services, for example, an application that provides Remote Desktop Access.

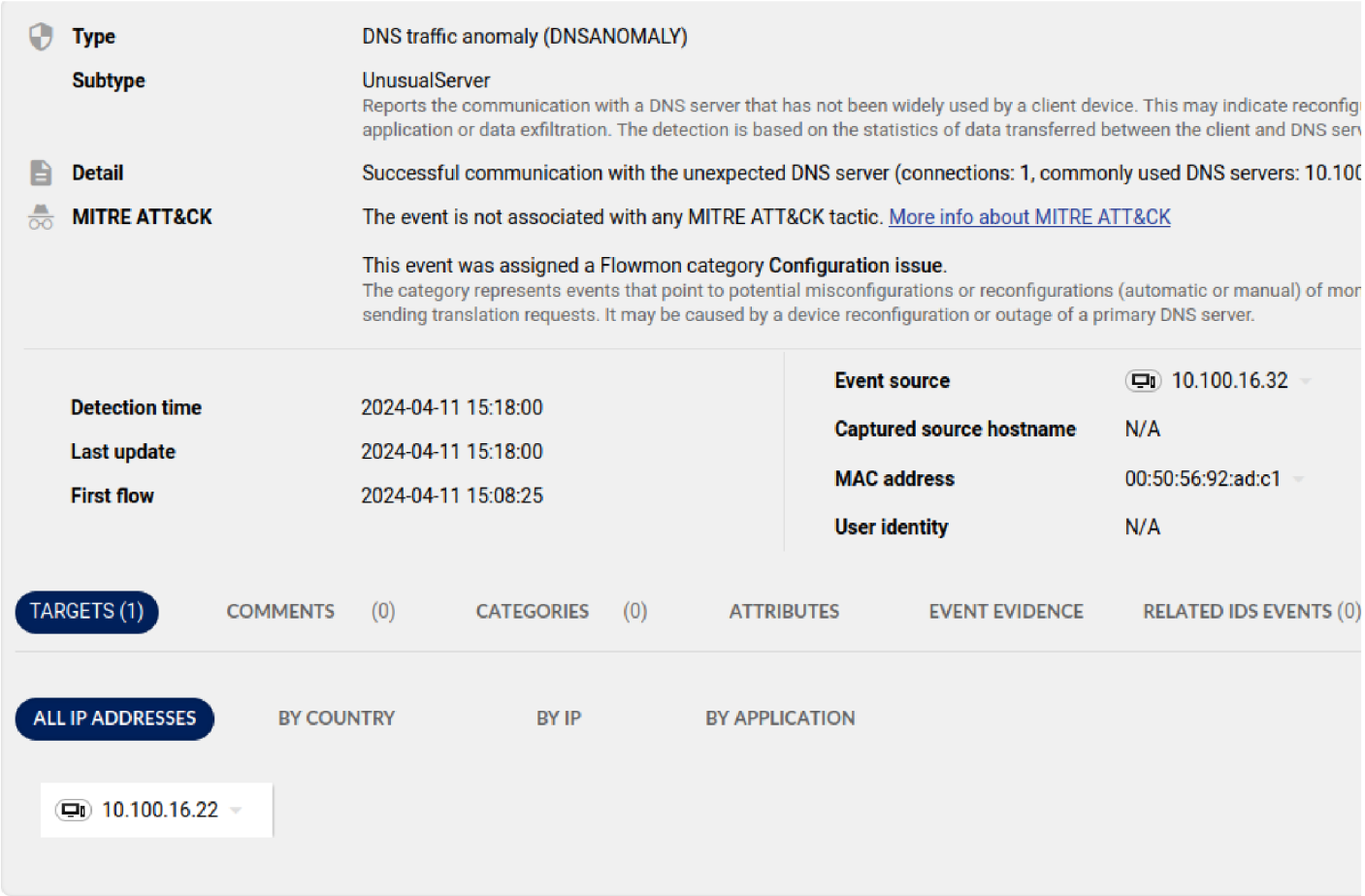

Figure 2.2: Flowmon ADS event [12]

3 Network Applications and Services

The Internet contains hundreds of thousands of services, presenting several challenges. How can we find the specific services that communicate with someone in our local network? Which of these services is critical for the Network Operator to detect? The answer to the previous question is crucial because the number of services is enormous, and most of them will not be helpful to the Network Operator, a typical user of Flowmon products.

The first section of the chapter describes the process of selecting network applications and services for identification. The second section describes selected types of services or applications. All types have one thing in common: they produce suspicious traffic. The last section describes how the vendors publish their IP addresses, domains, or URLs.

3.1 The Selection Criteria for Applications

We need to choose the correct criteria to identify the correct categories of services. We identify several possible approaches.

Largest Is the Best

This approach assumes that most of the internet traffic captured in Flowmon infrastructure is between the stations of the most significant technological companies and the stations in the monitored network. To select the most significant companies, we can use multiple methods. For example, we can compare companies by revenue or by the number of public IPv4 addresses.

This approach was selected by other commercial data providers, which already provide this data, so we decided to choose a different approach than this.

Operator Point of View

This approach assumes it focuses not on the quantity of internet traffic but the specific traffic. It should be interesting for the operator, who

monitors the internal network and is a typical user of Flowmon products. We focus on the services or applications that could be marked as suspicious for multiple reasons, like hiding something.

This approach is unique to the first one, so we decided to choose it and describe selected types of services in the Section 3.2.

Local Services and Applications

This approach selects the most significant applications or services used in the Czech Republic or its neighbors. It focuses only on these services, whose impact is limited by a few countries. The reason is that services with global impact are already included in approach *Largest Is the Best*. The reason for choosing the Czech Republic is based on the customer base of Flowmon solutions, primarily in central Europe.

3.2 Selected Applications and Services

The section provides information about selected applications or services. Every selected application or service is described from the point of view of the security operator, who is responsible for monitoring the local network.

Virtual Private Network

One of the most general definitions of Virtual Private Network(VPN) is: *' VPN is a private network constructed within a public network infrastructure, such as the global Internet.'* [13]

Nowadays, VPNs are also used to hide traffic, where origin packets are ciphered, packaged into new packets, and sent to the endpoint station. Any device on the Internet except the endpoint station cannot read the origin packet, which leads to Intrusion Detection Systems not being able to see the origin sender's IP address, port, and other information from the packet header. This technique could sometimes masquerade suspicious traffic.

Proxy Server

'A WWW proxy server, a proxy for short, provides access to the Web for people on closed subnets who can only access the Internet through a firewall machine.' [14]. Nowadays, attackers can use external proxies as intermediaries in network traffic to hide their location and IP address. For this reason, traffic between external proxy servers and some endpoint stations is suspicious.

Cryptocurrency Mining Pools

A cryptocurrency mining pool is a team that shares rewards from mining Bitcoin or another cryptocurrency. The mining process consumes a large amount of computational capacity and electric power; for this reason, mining is something that is usually not permitted on the internal company network. Another reason we assume that mining pools are important in the organization's traffic is because some crypto malware compromises the host station and then starts the silent mining process, which covers communication with the crypto mining pool. [15]

Remote Access Services

Remote access services enable users to connect to and control devices from a distance. The attacker could compromise this service to gain control of the victim's device. For this reason, traffic connected to remote access services could be marked as suspicious. [16]

Web Crawlers

A web crawler is a software that systematically and automatically explores the World Wide Web, navigating its graphic structure by following hyperlinks between web pages. They play a crucial role in indexing and organizing online content for search engines. Every large search engine maintains a few public web crawlers, which scan the entire Internet. [17]

A web crawler performs extensive scanning, potentially impacting the organization's local traffic. While scanning activities could initially raise concerns about a possible cyber attack, it is not considered a cyber

attack in that context when search engine providers scan. Detected communication with a web crawler can show the operator some open ports accessible to everyone from the Internet.

Vulnerability Online Scanners

Vulnerability scanning is the process of using one computer to identify vulnerabilities in another computer or network. Security experts employ this method to discover and secure vulnerabilities, while attackers may exploit it to harm systems. Some scanning services scan the entire Internet in periodical periods and expose vulnerabilities publicly on their websites. [18]

The network operator may consider vulnerability scanners suspicious because they produce unwanted scans and publish sensitive information about their infrastructure.

3.3 Data Publication of Application

Not all applications or services publicly share information about used IP addresses or domains; even if they share, they could still provide wrong data.

One of the everyday purposes of public IP addresses or domains is to avoid blacklisting application servers, which publish patches and security fixes, and application clients periodically access them. One of these examples is Cisco Firepower, which provides a list of IP addresses in its configuration manual on [19]. Another good example is a search crawler from Seznam called Seznam Bot, which publishes their IP ranges on [20].

Some providers provide more extensive IP addresses than they use, for example, AWS service at [21], offering a comprehensive list of IP addresses. After validating this page, it was found that there were multiple inactive IP addresses, leading to the decision not to use it.

Other situation-specific services do not want to publish their IP range because they could be blacklisted, for example, Shodan, which is a vulnerability scanner. However, there are few community-based lists of IP addresses and domains that Shodan uses, for example, [22].

Data Format

There is not widely accepted format of published IP addresses or URLs. Some vendors use CIDR format, while others use IP range format (192.168.1.0 - 192.168.255.255) or something different, like using .x to match everything in the IPv4 block. All found formats are described in the Table 3.1. The difference is in general data format because most vendors publish their data on HTML pages, but few publish data in JSON or CSV format. Another difference includes ports, which are important for virtual private networks or public proxies.

Table 3.1: Table of applications with different data formats

Vendor name	Format specification	Example
Amazon	Data in JSON.	IP: 50.19.0.1
Seznam	Using x.	77.75.76.x
Cisco	CIDR.	146.112.63.0/24
Community proxies	Data in CSV with port.	31.43.179.30:80

4 Web Scraping

The World Wide Web provides a vast amount of dynamically evolving data stored in web pages. Web scraping is a technique for extracting data from web pages and storing it in a file system or database for later retrieval or analysis. [23]

Typically, web scrapers simulate human exploration of the World Wide Web by implementing the low-level Hypertext Transfer Protocol (HTTP) or embedding a full web browser, such as Mozilla Firefox. Web scraping is closely related to web indexing, which indexes information on the web using a bot or web crawler and is a universal technique used by most search engines. [24]

This chapter describes web scraping techniques for gathering data with usages. At the end of the chapter, we describe The Robots Exclusion Protocol , which provide rules for scraping specific page.

4.1 Web Scraping Process

The web scraping process is divided into three stages, as shown in Figure 4.1. The process starts with the URL and ends with the data in the desired format. In this section, we describe each of these stages.

Fetching Stage

In this stage, the desired page must be accessed via HTTP or HTTPS protocol and then downloaded, or another approach is to use an embedded web browser, which uses the same protocol (HTTP or HTTPS)

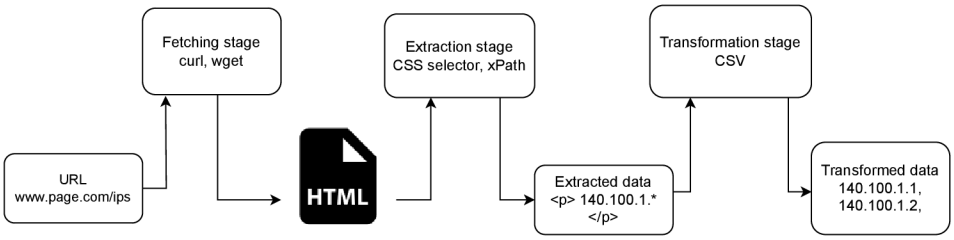

Figure 4.1: Scraping stages [25]

but will not download immediately. The stage ends when we receive the HTML content of the page as a response. [25]

Extraction Stage

Once the HTML page is fetched, the desired data needs to be extracted. This phase is called the extraction phase, and the technologies used are regular expressions, HTML parsing libraries, CSS selector, or XPath queries. The extracted data are raw text. In this thesis, we extract URLs, IP addresses, and CIDRs. [25]

Transformation Stage

Now that only the relevant data is left, it can be converted into a structured format like CSV or JSON and stored in persistent storage.

4.2 Applications of Web Scraping

Web scraping finds applications in various domains, making it a helpful tool. The following list highlights some key usages. [26]:

- **Online Price Comparison**: Gathering data for comparing prices across different online platforms.
- **Contact Scraping**: Extracting contact information from websites.
- **Weather Data Monitoring**: Aggregating and monitoring weather data from various sources.
- **Job Posting Information**: Scraping job-related details from job portals.
- **Real Estate Property Lists**: Collecting property listings from real estate websites.
- **Government Data Collection**: Gathering data from government websites for analysis.
- **Market Analysis**: Extracting relevant data for market research and analysis.

Web scraping is not limited to individual projects; it is also employed on a larger scale by search engines. At a macro-level, the metadata of nearly every website is continually scraped to build and update Internet search engines such as Google Search or Bing Search. [23]

4.3 Web Scraper Structure

Web scraper broadly consists of two parts as we can see in Figure 4.2. The first is the crawler for crawling links, and the second is the data extractor from crawled links.

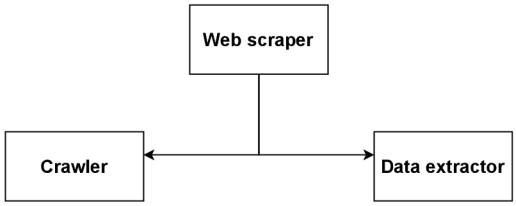

Figure 4.2: Tree diagram of web scraper [27]

Crawler

The initial step in extracting data is deploying a web crawler or spider. This intelligent bot methodically scans the vast expanse of the World Wide Web, primarily for web indexing purposes, which search engines later utilize [24].

Crawlers usually use a recursive algorithm to crawl websites. The crawling process starts with the list of URLs to crawl, called seed URLs. After visiting these seed URLs, the crawler identifies all the links on the page and stores them in the list of URLs to visit, called the crawl frontier. Now, URLs from the crawl frontier are visited recursively. This process continues until all URLs are visited.[27]

Data Extractor

The web pages are mainly designed for humans, not automatic scraping. This means that pages contain a lot of unnecessary elements (HTML tags, CSS styling, additional descriptive text...). All we need

to do is look for useful information, extract it, and convert it to the desired format. To separate the useful from the useless, we target only the part of the page that contains information. To target this specific part, we can use several technologies, such as CSS selectors or xPath. [27]

XPath is a language for addressing parts of an XML document. [28] The World Wide Web Consortium (W3C) has set standards for the use of XPath. Alternative technology to XPath is CSS selector, which is a method for finding HTML elements on web pages and extracting data from them. [29]

4.4 The Robots Exclusion Protocol

The Robots Exclusion Protocol has been proposed in RFC 9309 to provide crawler regulations. A file called robots.txt, which contains crawler access policies, is deployed at the root directory of a website and accessible to all crawlers. Ethical crawlers read this file and obey the rules while visiting the website. [30]

Robots.txt consists of multiple fields.

- **Disallow**: field specifies the directory or page that cannot be scraped
- **Allow**: field specifies the directory or page which could be scraped. Allow field allows overriding a disallow rule to allow the crawling of a subdirectory or page in a disallowed directory.
- **Crawl delay**: field specifies the number of seconds a crawler should wait between successive requests to the server.
- **Sitemap**: This rule provides the location of the XML sitemap to help search engines better understand the website's structure.

5 Conceptual Design of Application Database

The chapter aims to describe all concepts and requirements and provide the reader with nontechnical descriptions. Application Database is a software solution that is part of the thesis and was developed for Industrial Partner of Masaryk University Progress Software.

The chapter starts with a section about cooperation with engineers from Progress Software. The chapter continues with requirements specification and a high-level overview supplemented with a data flow diagram. Then, we focus on the use cases from two points of view. At the end of the chapter, we describe data anomalies in our data. The whole chapter finishes with a section describing the run of our proposed software. This section uses knowledge from previous sections.

5.1 Cooperation with Progress Software

Progress Software is a global company that produces multiple products, including Flowmon. Flowmon is a tool for network monitoring using IP flows. Flowmon has several modules, such as Application Performance Monitoring (APM) and Anomaly Detection System (ADS). Our goal is to implement software that provides data to the server, which is then used by Flowmon modules. The Flowmon Anomaly Detection System will then utilize the data we have provided.

One of the features of the Flowmon Anomaly Detection System is application mapping in ADS events. Application mapping provides users with additional information about the application behind the source or target in ADS events. This feature depends on the Indicator of Application (IoA) data, which represents IP addresses, URLs, and domain names of applications or services.

Our goal is to design and implement solution that improve data accuracy and allow us to extend existing data with new data sources. We named our solution the Application Database (App DB). The Application Database will be deployed on the production server and will serve Progress customers.

5.2 Requirements

Design process starts by defining functional and non-functional requirements. A functional requirement specifies a function that the product must be able to perform, and non-functional requirements specify the system's required overall attributes. [31]

Functional Requirements

Functional requirements were created based on the discussion with Progress Software software engineers. This requirement was later extended with additional requirements not part of the thesis software solution. These non-implemented requirements are the Presentation layer and security from non-functional requirements.

- **Export**: App DB must be able to export data (even the whole database) into .csv files. App DB exports data in only one format for application mapping, which is then used in Flowmon ADS.
- **Meaningful data**: Every IoA in App DB has an assigned application, company, and category (tag).
- **Data changes**: When App DB detects large changes in data, it must report an anomaly to an operator. A large change in data could mean a fundamental change in our data sources that compromises the accuracy of the data. For example, an application may be discontinued or purchased by another company.
- **Automation**: App DB stores decisions from the operator to automatically resolve repeating problems. For further details, see the section essential change 5.5
- **Multiple resources**: App DB stores data from multiple data input files and handles applications collisions. For further details, see the section collision 5.5

Non-functional Requirements

To make Application Database usable Application database will be later deployed on the production server, for this reason we must satisfy these requirements:

- **Data Integrity**: The App DB should only contain accurate data. Efforts should be made to prevent situations where anomaly events are labeled with the wrong application.

- **Availability**: The App DB should provide data even if some sources stop working. There are various reasons why a source may temporarily stop functioning, such as a web scraper losing access to certain websites. In such scenarios, the App DB must continue to function and utilize an older snapshot of working data.

- **Interoperability and Deployability**: The App DB will be deployed in a containerized environment to enhance system independence and ease of deployment.

- **Usability**: The App DB should be managed via a web frontend to make it accessible to a wider audience, including support teams, pre-sales, product management, and developers. Interaction with the App DB should not require changes to the source code but should be facilitated through a web interface. This requirement is not part of the thesis solution and will be implemented in the future.

- **Modularity**: It should be possible to add new sources of application mappings to the App DB.

- **Security**: All APIs and databases must be properly secured to ensure data integrity and protect against unauthorized access. This requirement is not part of the thesis solution because the API is not part of the thesis solution.

5.3 High Level Overview

Application Database depends on other systems and provides data to other systems. Our goal in this section is to describe the high-level overview, dependencies on the other systems and give necessary context for Figure 5.1.

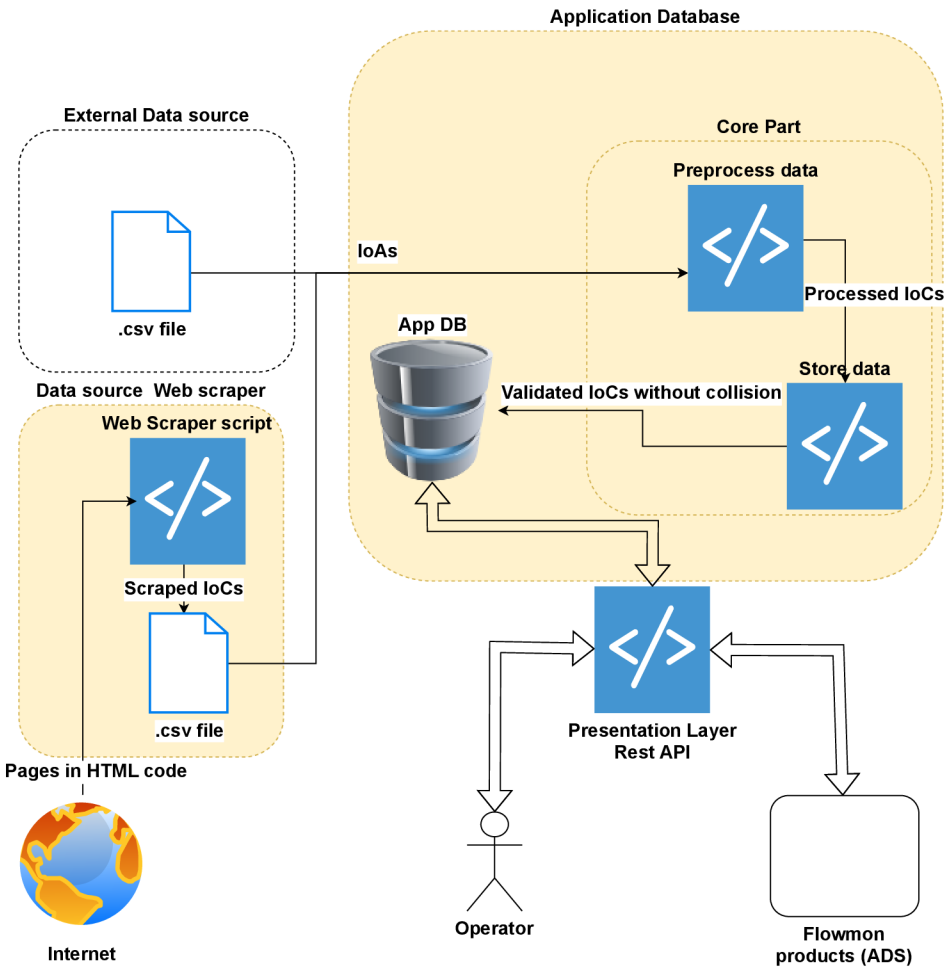

Figure 5.1: Data Flow diagram, where the yellow parts are part of the thesis implementation.

Application Database depends on data sources (internal or external), that provide data in files. These files are periodically updated. The Application Database could be divided into these three parts:

- **Core part**: periodically loads data from data sources, processes them, handles essential changes or application collisions, and stores them in the database.
- **Database**: stores data about Indicators of Application, Applications, Categories, and more.
- **Presentation layer**: is represented by the REST API server, which provides endpoints for the operator and Flowmon ADS. Operators resolve collisions, make essential changes, insert new data, and more. Use cases for operator are described in Section 5.4.

Application Database will be extended by a graphic user interface for the operator and some more modules, which are separate from the thesis solution; we will not discuss them more.

Data Sources

Application Database takes as input .csv files generated by data sources. The data source is a script that periodically runs and generates .csv files. For example, a data source could be a script that downloads data through the REST API server, parses them, and stores the result into .csv files. Our solution contains a web scraping solution. Data sources contain one unique source, a manual data source, and a manual data source created by the maintainer of the Application Database (operator). This data source has a bigger priority than every other source and could be used to correct data in unusual situations. For example, when Twitter was renamed to X, we did not need to wait until our data sources containing information about Twitter changed the naming and description of the application. We can use manual data sources to overwrite it.

Output

As we mentioned before, an Application Database will be necessary for the application mapping feature in Flowmon; after discussion with

software engineers working on Flowmon products, we identified additional possible usage of the data stored in the Application Database. Additional application data use cases are in network and performance monitoring fields. For this reason, we decided to provide an output database with a REST API server, which, in the future, can be easily extended by a new API endpoint that will provide the desired data. The REST API server is not part of the thesis solution. Output of the current state of the Application Database is now exporting handler and database.

5.4 Use Cases

Use cases are divided into two parts. The first part contains use cases for Flowmon products, which were defined in cooperation with engineers from Progress software. The second part consists of use cases for the operator responsible for maintaining data.

Flowmon Products

Application DB provides data in different formats compatible with:

1. **Application mapping feature**.
2. **New Backend for Flowmon Collector**.
3. **Whitelist format**: Whitelists are used in Flowmon ADS.

Operator

One potential issue for the App DB is data quality. All data is acquired through external sources, which will evolve. Any change in sources may cause the inability to obtain data. For this reason, the Application Database will have its operator responsible for maintaining all sources operational and up to date.

Functionality for the operator is not part of the thesis solution. It should be delivered later than the operator use cases.

1. Create and remove new application with scraper or external configuration.

   (a) New applications should be easy to add to increase the number of applications as much as possible.
   (b) Another reason is that we need a fast way to satisfy customer requests to add a new application.
   (c) External configuration defines the relation between the application in the external data source and the application in the Application Database. In most cases, the same application is defined by External source and Application Database, but in some special cases(acquisition), Application Database could have different name for the application.

2. Extend existing application with a new source of data (scraper).
3. Search existing applications by name and ignore specific IoA.
4. Ignore specific IoA.
   (a) By removing the IoA, we ensure that the IoA is never associated with any application or company. This step will help us to reduce the number of false positives and possibly respond to false positives in the data.
5. Manually insert IoA with a regular verification interval.
   (a) The operator can insert IoAs into the database and assign them for specific applications. This will be a static configuration with the highest priority over other sources.
   (b) Every manually inserted IoA has a time reminder to avoid old false positive records.
6. Resolve a conflict when one IoA is assigned to multiple applications.
7. Resolve essential change.
8. Display history of IoA and application.
9. List IoA, application and filter them by company, application name, IoA

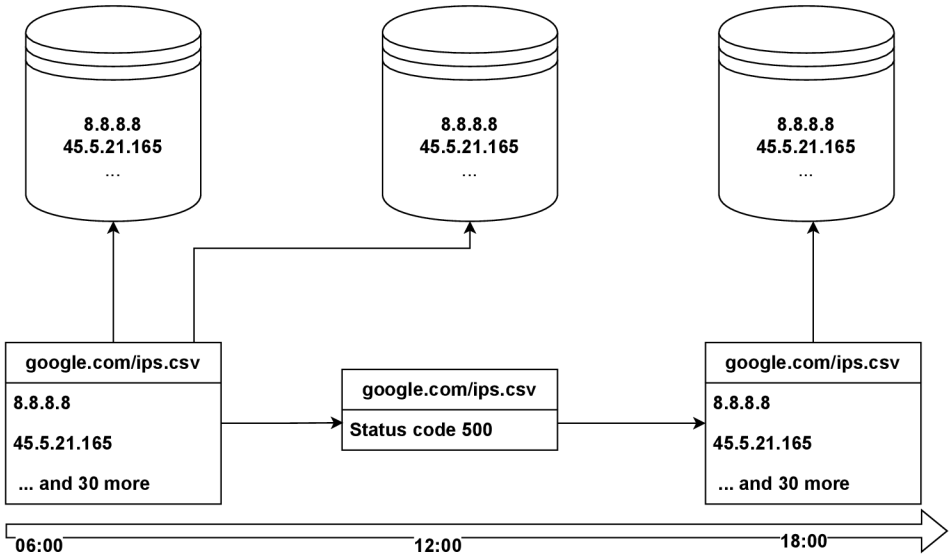

Figure 5.2: How Application Database handle data source outage

5.5 Data Anomalies

One non-functional requirement in the Application Database is Data Integrity. Application Database must handle more complex situations than just the wrong format of IPv4 address. Application Database depends on different types of data sources. Web scraper, for example, can be easily affected by changing the HTML structure of the scraped page. We identify two types of data anomalies that should be handled.

Essential Change

The Application Database receives Indicators of Application (IoAs) from multiple sources that change over time. If the amount of IoAs from a specific source is significantly lower or higher than the previous amount of a specific source, we define this change as essential. We use a predefined threshold to do this. An essential change in the number of IoAs may mean that the source no longer contains the correct information, has an outage, or is incomplete.

When an essential change happens, the Application Database uses old data from the same source for a defined time; after that, it stops

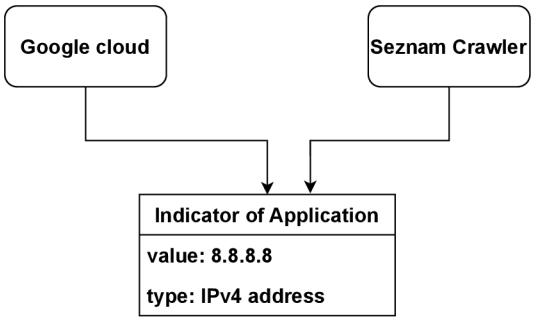

Figure 5.3: Application collision

using old data and returns no indicators for affected sources. When some source has an outage for a few hours, the Application Database handles this outage automatically without assistance, as shown in Figure 5.2. The Figure is described in the list below:

1. Initial run in Figure 5.2 gets 32 IoA from a specific source.
2. The second run cannot gather data from the source, so an essential change happens, and the Application Database uses data from the first run instead of the second.
3. In the third run, the data source provides 32 IoA, the same amount as in the first run, so the Application Database updates the data in the database.

If the source returns a significantly different number of indicators for a one week or more, then the operator must handle essential changes manually. The operator can allow the change, temporarily use old data, or disable the source.

Application Collision

Application collision refers to the situation when the Application Database detects two identical IoAs but with two different applications assigned. This situation could indicate company changes, such as new acquisitions, where one application became part of another, or when companies started sharing their infrastructure. Another example is the rebranding process, when some application changes its name. In

that case, the Application Database could have stored old and new applications as two different entities.

If the scraping frequency is high, then the Application Database will detect the same collision multiple times, so it should be able to resolve them automatically. For better understanding, see Figure 5.3.

For these situations, we introduce a hierarchy between applications. Every application could have a more generic application as a parent and more specific applications called children. For example, we could use an application called Google Cloud, and as children, we could have Google Cloud Notification Service. Each application has at most one parent and zero or more children.

Application collision should be resolved in this order:

1. Detect if one app is the child of another, e.g., application Google Cloud and application Google Cloud Notification Service. If yes, then set the source of IoA to the child, and the collision is resolved. Otherwise, continue with step two.

2. Find a rule (see Section 6.3) that applies to those applications and IoAs. If the rule is found, we resolve the collision automatically. Otherwise, continue to step 3.

3. Create a ticket (for more see Section 6.3) and let the operator resolve them.

4. After resolving a ticket, create a new record in the rule table, representing the operator's decision. The operator can decide not to create a rule in an extraordinary situation, such as a one-time decision.

Hierarchy of Applications

In larger companies, some applications incorporate smaller applications, such as Office 365 and Excel. We have decided to solve this by allowing each application to have one parent and multiple children. This hierarchy will help us resolve conflicts between the data and can provide other valuable information to the user.

After discussion, we decided not to implement this feature yet because we could not find this problem in our existing data from

scraper and external resources. We have concerns about misusing this feature by creating some general applications for the whole company like Microsoft General.

5.6 Example Run of Core Part

This section is designed to connect all the concepts discussed in this chapter by describing an example run of the Core part, described as a step-by-step algorithm in words to provide a clear and structured explanation.

1. Load manual-created applications with descriptions, icons, and categories.
2. Load applications from other data sources and merge them with manually created applications.
    - Merging of the applications depends on ID.
    - If we found an application with the same name but a different description, we would use manually created descriptions. This approach will be the same for icons and categories.
3. Repeat steps 1 and 2 for companies and then for categories.
4. Remove applications that the operator ignored.
5. Remove IoAs, which the operator ignored.
6. Load IoAs from the data source and assign each IoA to an application.
7. For every application in the data source detect, if the change of the data is essential:
    - Essential changes may indicate some bugs, such as assigning the IoA from the external source to another application (from Twitter to X) or changing the HTML code.

- If the change is essential, resolve it automatically using rules. If the change cannot be resolved automatically, the operator must resolve it.
- If the change is not essential, we will ignore this change and continue.

8. Combine data from individual data sources and assign them to an application.
   - If one IoA appears in multiple sources of one application, we log it.
9. Detect a conflict in IoAs:
   - IoAs conflict means two records have the same IP address (URL, domain) but different applications.
   - The operator selects the correct resource, and the Application DB resolves the next situation automatically.
10. Store IoAs in the database.

6 Logical Design of Application Database

This chapter aims to describe the Application database more technically; we focus on the data structure. In section 6.1, we describe the format of the data stored in the database using an ERD diagram. The structure of the proposed solution is described using the class diagram in Figure.B.2.

6.1 Data Structure

Application Database depends on the database, and in this section, every database table is described. The database provides data not only for the Application Database but also for Scraper. For more information about see Chapter 7.

The data structure is visualized in B.1, demonstrating the relation between entities (database tables). We also provide more detailed information about entities here:

- **Application**: This entity represents stored information about the application. It also stores foreign key to the parent application for further details, see Section 5.5.

- **External ID mapper**: Applications could be loaded from different external sources. One of them provides applications with their ID, which is already in the application mapping feature in Flowmon ADS. For this reason, we first try to assign the original ID. If we cannot, we will store the original ID in this table to reproduce it.

- **Source**: Key element of Application Database. The source represents the application's data source. Every application must have one or more sources. The source is unique for every application.

- **Indicator**: This table stores the indicator of the application with valid_until timestamp, which is used to clear the database from the old records.

- **Application Ticket**: This ticket is used for handling collisions. We decided to use a more generic name for the ticket to extend it in the future. For more information see Section 6.3.

- **Application Rule**: Rule for resolving Application tickets, for further details, see Section 6.3.

- **Source Ticket**: This ticket is used for essential change. We decided to use a more generic name for the ticket to extend it in the future. The name is based on comparing sources. For more information see Section 6.3.

- **Source Rule**: Rule for resolving Source tickets, for further details see section 6.3.

- **Temporary Indicator**: Table used for internal operations. It is filled with all loaded indicators and compared against the Indicator table. As a result, we select indicators of application that are loaded into the temporary table but not in the Indicator table. After that, Temporary Indicator table is truncated.

- **Scraper Config**: Entity stores configuration, which is necessary for scraping. For further info, see Chapter7.

6.2 Code Structure

The Application Database is written in Python3 using object-oriented programming. A class diagram at Figure B.2 describes the most important parts of the project. In the following list, we describe all important parts of the code:

- **Data loaders**: They are responsible for loading data from different types of inputs. The loader loads and transforms data into classes and proceeds them to the handlers. This part was inspired by the Factory pattern.

- **Handlers**: Handlers are the most complex part of the Application database. Handlers handle data anomalies and then store

Indicators in the database. The Single chain of responsibility pattern inspires this part. Every handler handles only one part. For example, a collision handler handles only application collision.

- **Repositories**: They are responsible for accessing the database; every table has its repository. All database transactions are in repository files to isolate and divide functionality.

Object Relational Mapping

Application Database uses the concept of Object Relational Mapping(ORM) through the SQLAlchemy library. ORM maps database tables to classes and records in the database to objects. The most significant advantage of this approach is faster development than using SQL language, and another advantage is to be less dependent on database technology like PostgreSQL. On the other hand, ORM is mostly slower than pure SQL.[32]

6.3 Tickets and Rules

In section 5.4, we describe data anomalies that could occur and must be handled by the Application Database. In this section, the solution is described in more detail. Tickets are created for every unresolved essential change and application collision. The format of the ticket is different for collisions and changes.

Rules resolve tickets. Application Rule resolves Application Ticket, and Source Rule resolves Source Ticket. Every rule could resolve one or more tickets. The rules are formatted according to the ticket format.

Tickets for Essential Change

Ticket is the reaction of Application Database to essential change, for further reading about essential change see Section 5.5.

The essential change is detected by the division of the number of new IoA of a specific source versus the number of old IoA stored in the database of a specific source. The process then continues:

1. The result of the division is compared against two general thresholds (higher and lower). If the result is between the higher

and lower threshold, the Application database detects the non-essential change and moves to the last step; otherwise, we continue with step two. This calculation proceeds if the number of records in the old source is higher than 10 to avoid false positive detection.

2. Application Database searches for the matching rule. The matching rule must have the same source ID as the affected source, and the rule's threshold must be bigger or smaller than the result. If the rule does not exist, we create a ticket.

3. Application Database moves affected indicators to the pending indicators table. Affected means that the indicator is freshly added, and the indicator's source is part of the essential change. This indicator will not be present in the indicator table because it could lead to a false positive indicator (indicator with wrong application).

4. Application Database extends the validity of old indicators from affected sources for a limited time. After this time, the old indicators will be removed. The operator must resolve this essential change by this time to avoid data loss.

5. Non-affected IoA continues to the collision handler

The ticket of essential change consists of these fields:

- **Old count**: count of Indicators of Applications with specific source stored in a database.
- **New count**: count of Indicators of Applications with specific source loaded from the input.
- **Resolved**: The boolean variable which is set to True when it was resolved by rule.
- **Created at**: creation time of ticket.
- **Foreign key of rule**: Empty until resolve.
- **Foreign key of source**

Rules for Essential Change

Rules for Essential change could, for specific sources, override general lower and higher thresholds. The format of the rule is the following:

- **Upper threshold**
- **Lower threshold**
- **Permanent**: The boolean variable set to False when the rule can be used only once; otherwise, the rule will be used in every run.
- **Foreign key of source**

Tickets for Application Collision

Ticket is the reaction of the Application database to collision. Collision happens when two same IoAs have assigned two different applications. Detection of collision is processed as follows:

1. Filter all stored indicators, we assume that all stored indicators are without essential change and collision.
2. Detect collision in remaining indicators and between remaining indicators and stored indicators.
3. Apply rules on collided indicators if some indicators remain unresolved. Then, create a ticket for them.
4. Remove remaining indicators. We do not move these indicators to the pending because the Application Ticket provides all the necessary information to recreate the collided indicator.

The ticket of application collision consists of these fields:

- Indicator value - Value of indicator, for example, IPv4 address, domain name, or URL.
- Foreign key of collided source - Key of source, which causes application collision
- Foreign key of rule - Empty until resolve

Rules for Application Collision

The rule for application collision consists of three fields:

- **Indicator value** : The indicator's value could be combined with an asterisk, which could be used as anything. For example, value 8.1.* matches every indicator beginning with 8.1.
- **Foreign key of the chosen application**
- **Foreign key of the ignored application**

We do not use a permanent field because we assume that application collision is a long-term problem that must be permanently resolved.

6.4 Final State of Application Database

The Application Database consists of two docker containers. The first container is the PostgreSQL database, and the second is the core script loading data into the database and handling all functionality mentioned in this chapter. Both containers were deployed into a clone of the production server and tested using internal data and data from Scraper.

Provided solutions combine multiple design patterns, which makes code easy to extend. For example, if we detect a new possible data source, the effort to integrate this source into the Application Database includes only creating a new loader script, which has a predefined interface, due to inheritance from a generic loader class. Another example would be adding new functionality into data processing, which handlers resolve. To add new functionality, we need to implement only a new handler and specify its order by putting it into the correct place in the chain of responsibility.

The provided solution solves functional requirements specified by the engineers from Progress Software, including detecting data anomalies, creating a system for tickets, and resolving mechanisms to solve these tickets, which makes the system more robust and solves one of the critical problems of application mapping feature, which is quality of provided data.

Deployment

The core script works with two inputs, and we tested it on a dataset of 20 MB of data, including applications, IP addresses, and CIDRs. The average runtime on the clone of the production server was about 4 minutes for the first run; the runtime decreased to approximately two minutes for the next run. The runtime decrease is caused by optimization, which inspects data anomalies (application collision and essential change) only in new data. This runtime could be different in the future, depending on the docker configuration and assigned resources like RAM and CPUs.

Validation

The actual solution was validated manually. Another validation was fulfilled using component and unit tests, which covered most of the project. The code was inspected using the static analysis tool Sonarqube, which detects code smells, possible bugs, and vulnerabilities in the code.

7 Implementation of Scraper

The chapter aims to describe the Scraper, which provides data for the Application database. The complexity of the Scraper is lower than that of the Application Database, so we decided to describe more briefly.

The chapter begins with the requirements part and then continues with the scraper structure, where we cover technical details. The last chapter focuses on the integration of Scraper into the Application Database.

7.1 Requirements

The process of creating requirements is similar to that in the Application Database. The biggest challenge is to provide a solution that is easy to expand.

Functional Requirements

1. **Extract**: Scraper extract Indicator of Applications (IoAs) from a given URL using a specified CSS selector, XPath, or custom algorithm.

2. **Load**: Scraper loads all necessary information (URL, CSS selector) for scraping pages from the database.

3. **Store**: Scraper stores all extracted information into a CSV files.

Nonfunctional Requirements

1. **Easy to expand**: It should only be necessary to provide the URL, CSS selectors, icon, and application description to scrape a new page.

2. **Interoperability and Deployability**: The Scraper will be deployed in a containerized environment to enhance system independence

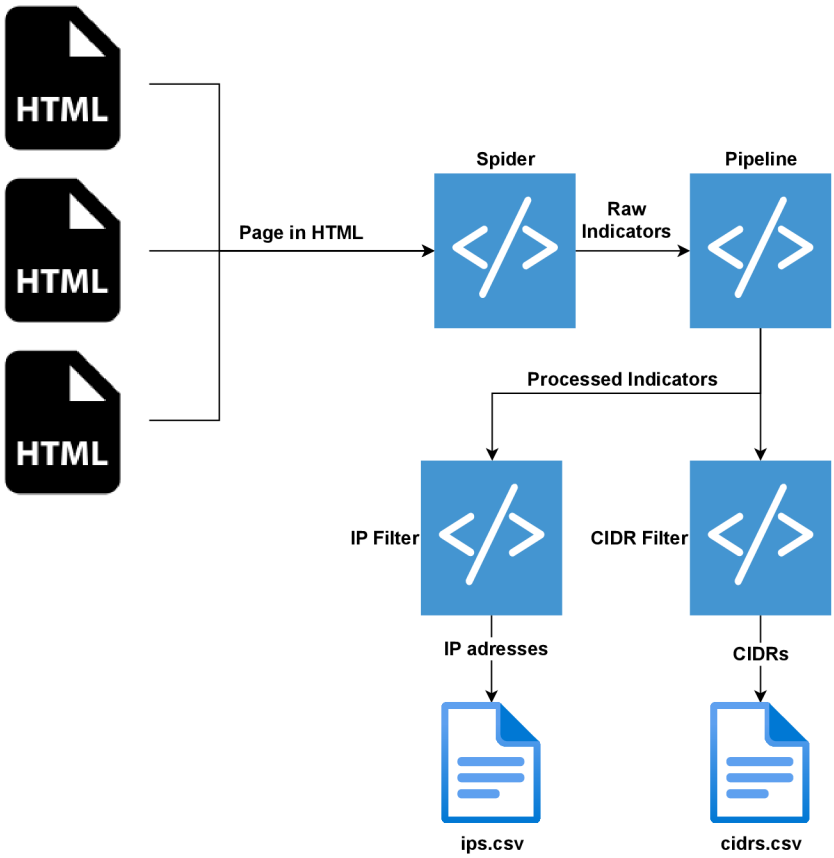

Figure 7.1: Scraper Data Flow Diagram

7.2 Structure of Scraper

Scraper is based on the framework Scrapy, which also defines the Scraper structure. Scraper Data Flow Diagram describes Scraper in 7.1. The whole Scraper consists of these small parts:

- **Spider**: The class responsible for parsing the whole page in HTML code. The output of the Spider is a list of raw indicators. Raw means that indicators still must be preprocessed. Raw indicators still have HTML tags, or are they in some weird format like IP range or IP address with an asterisk? The scraper has a generic spider that loads configuration from the database and then uses it to scrape the pages. Configuration contains CSS

selector and URL. Scraper could also have another specialized spider, which does not need a database. [33]

- **Pipelines**: Classes for preprocessing indicators. The final pipeline return Indicator is ready to be exported. Pipelines create one chain. Every pipeline class is responsible for one preprocessing step, like formatting CIDR. [33]
- **Filters**: Classes for storing indicators in the output file. Scrapy sends all indicators into all filters. The filter checks whether the Indicator is in the correct output format. Scraper has its filter for every output format; output format is, for example, IP address, URL, or CIDR. [33]
- **Setting**: Python module configuring the whole project. It provides many possibilities, which we mostly do not use. Scraper uses settings mainly to order pipelines and to configure filters before export.[33]

7.3 Deployment and Validation

Scraper was deployed on the clone of the production server as a containerized solution and adequately tested. We mainly focus on the testing pipelines during the test using Pytest technology. Due to the size of the Scraper, we have not created any acceptance testing criteria or component testing. The Scraper was validated manually by inspecting every single scraped page. The deployed version scraped eight different pages and produced more than 10,000 results. One page was scraped using a custom script, and the rest used only a single CSS selector.

The whole code was also analyzed with a static analyzing tool called Sonarqube, which detects code smells, security vulnerabilities, and possible bugs. The Scraper is a robust asynchronous solution that provides an easy-to-extend possibility. There are two possible ways to create a new scraper: one is using a CSS selector, which is required, and the second way is to create a custom script to scrape more complex pages.

The Scrapes include pipelines, which can parse specific formats of IP addresses with wildcards character asterisk. This character is

used in the IP address to match every possible number. For example, 8.8.8.* stands for 8.8.8.0, 8.8.8.1, 8.8.8.2 ... This asterisk could also occur in URL, which Scraper handles and processes correctly. The last nice feature of the Scraper possibilities is the possibility to parse IPv4 and IPv6 addresses or CIDRs.

Integration into Application Database

Scraper is deployed into the container, which is periodically run. Scraper uses the shared database, which provides scraping configuration and application information. Scraper provides CSV files, which the Application Database uses.

8 Conclusion

The thesis aimed to develop a software solution that would enhance the application mapping feature in the Flowmon Anomaly Detection System. This solution, named Application Database, was required to aggregate data from multiple data sources, manage data anomalies, and store and export data in the desired format. The solution provides easy-to-extend possibilities and solves the critical problem of application mapping, which is data quality.

To demonstrate the Application Database's functionality and extend the amount of provided Indicators of the Application, we introduced a new data source called Scraper. This Scraper asynchronously extracts data from a desired webpage using the provided selector and URL, or a custom script. Periodically, the Scraper downloads the latest configuration and scrapes the specified webpage. The extracted data is then processed and stored in files.

The core script of the Application Database loads data from a predefined location, ensures data correctness, and stores the validated data in the database. Otherwise, it tries to resolve problems using rules or creates tickets to resolve them in the future. Rules can automatically resolve problematic data sources, which makes the system less demanding on maintenance. Subsequently, it exports the data to a format compatible with Flowmon ADS.

The resulting software solution, including the Application Database and Scraper, is a collection of multiple containers designed for easy deployment. The Application Database was well tested, including component and unit tests covering all advanced functionalities. Additionally, the entire codebase was analyzed using the static analysis tool Sonarqube, ensuring its safety and reliability.

Application Database will be extended by API for the operator to handle all use cases mentioned in Section 5.4, and these functionalities should be covered in the front end of the project. The Application Database was tested on a clone of the production server. Application Database will be deployed on a production server, providing data to all Flowmon products. Application Database deployment on the production server is expected in autumn 2024.

A Electronic attachments

Electronic attachments are located in the thesis archive. The attachment contains one folder called diploma thesis. This folder contains the source code of Scraper and Application Database located in the app-db directory and persistent storage used in docker called volumes. Volumes consist of input directories called external data, scraper data, and output data stored in the output directory. There is also a folder called Postgres, where the database persists. The source code is documented using markdowns README files.

B Diagrams

The following chapter includes Entity relation diagram and Analysis class diagram.

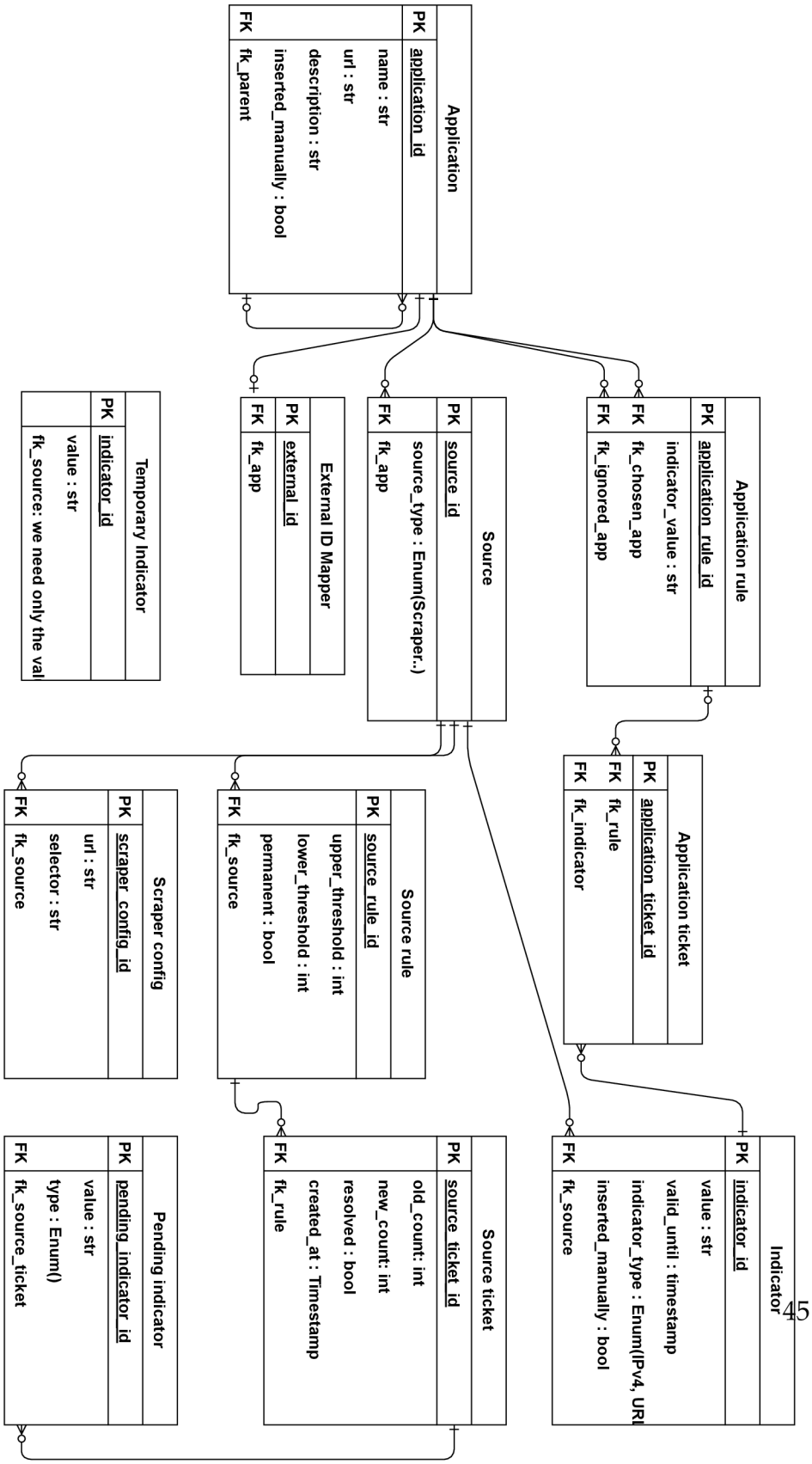

Figure B.1: Entity relation diagram

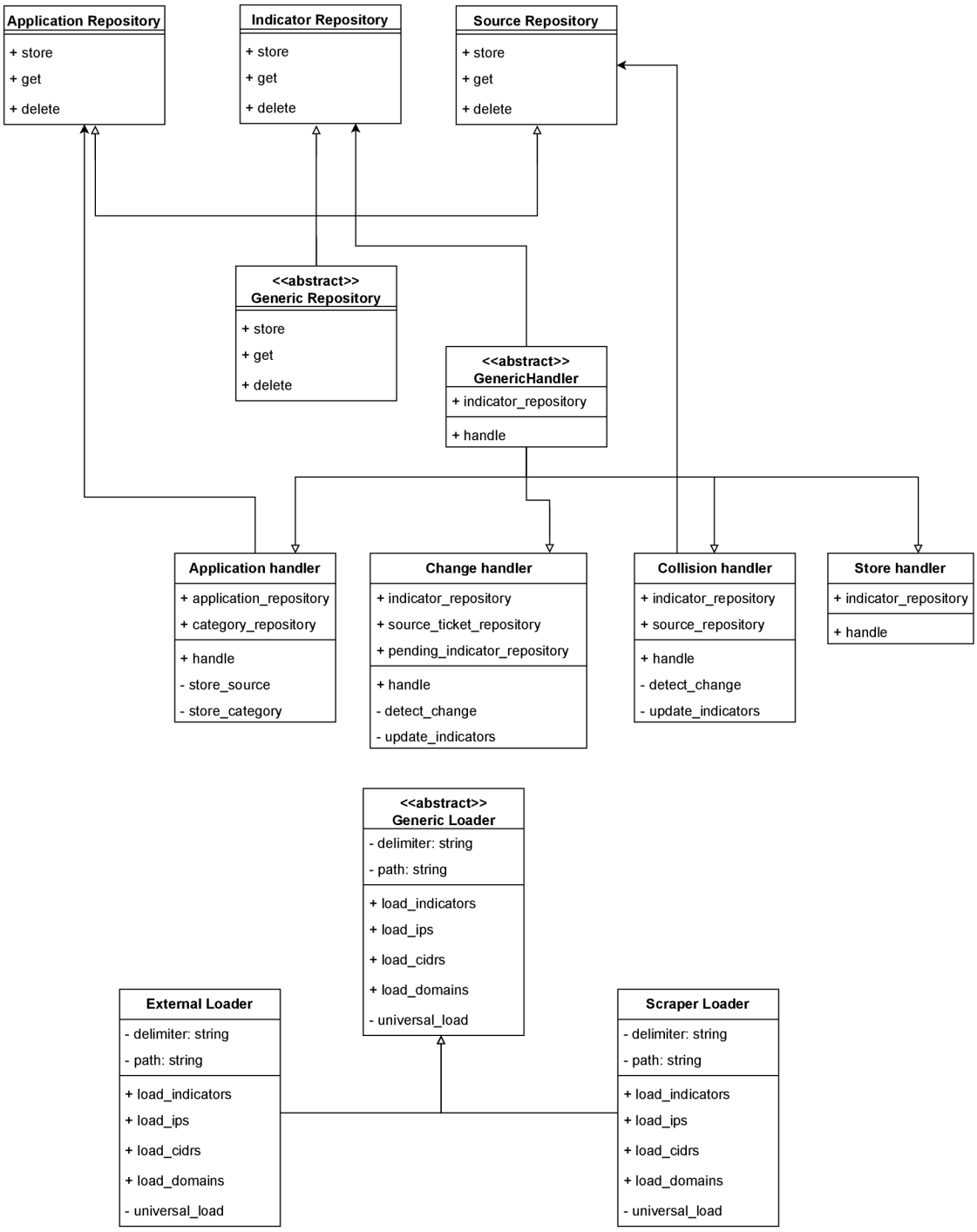

Figure B.2: Analysis class diagram

Bibliography

1. HOFSTEDE, Rick; ČELEDA, Pavel; TRAMMELL, Brian; DRAGO, Idilio; SADRE, Ramin; SPEROTTO, Anna; PRAS, Aiko. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys & Tutorials*. 2014, vol. 16, no. 4, pp. 2037–2064. Available from DOI: 10.1109/COMST.2014.2321898.
2. STEINBERGER, Jessica; SCHEHLMANN, Lisa; ABT, Sebastian; BAIER, Harald. Anomaly Detection and mitigation at Internet scale: A survey. In: *Emerging Management Mechanisms for the Future Internet: 7th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2013, Barcelona, Spain, June 25-28, 2013. Proceedings 7*. Springer, 2013, pp. 49–60.
3. AITKEN, Paul; CLAISE, Benoît; TRAMMELL, Brian. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* [RFC 7011]. RFC Editor, 2013. Request for Comments, no. 7011. Available from DOI: 10.17487/RFC7011.
4. MICHEEL, Jörg; DONNELLY, Stephen; GRAHAM, Ian. Precision Timestamping of Network Packets. In: *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*. San Francisco, California, USA: Association for Computing Machinery, 2001, pp. 273–277. IMW '01. ISBN 1581134355. Available from DOI: 10.1145/505202.505236.
5. DUFFIELD, Dr. Nick; MOLINA, Maurizio; RASPALL, Fredric; NICCOLINI, Saverio; ZSEBY, Tanja. *Sampling and Filtering Techniques for IP Packet Selection* [RFC 5475]. RFC Editor, 2009. Request for Comments, no. 5475. Available from DOI: 10.17487/RFC5475.
6. CLAISE, Benoît. *Cisco Systems NetFlow Services Export Version 9* [RFC 3954]. RFC Editor, 2004. Request for Comments, no. 3954. Available from DOI: 10.17487/RFC3954.
7. MARK, Lutz; WAGNER, Arno; BOSCHI, Elisa; ZSEBY, Tanja; TRAMMELL, Brian. *Specification of the IP Flow Information Export (IPFIX) File Format* [RFC 5655]. RFC Editor, 2009. Request for Comments, no. 5655. Available from DOI: 10.17487/RFC5655.

8. INTERNET DATA ANALYSIS (CAIDA), Cooperative Association for. *Summary of Anonymization Best Practice Techniques*. 2008. Available also from: `http://www.caida.org/projects/predict/anonymization/`. [Online].

9. TRAMMELL, Brian; WAGNER, Arno; CLAISE, Benoît. *Flow Aggregation for the IP Flow Information Export (IPFIX) Protocol* [RFC 7015]. RFC Editor, 2013. Request for Comments, no. 7015. Available from DOI: `10.17487/RFC7015`.

10. LIAO, Hung-Jen; RICHARD LIN, Chun-Hung; LIN, Ying-Chih; TUNG, Kuang-Yuan. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*. 2013, vol. 36, no. 1, pp. 16–24. ISSN 1084-8045. Available from DOI: `https://doi.org/10.1016/j.jnca.2012.09.004`.

11. SCARFONE, K A; MELL, P M. *Guide to Intrusion Detection and Prevention Systems (IDPS)*. 2007. Available from DOI: `10.6028/nist.sp.800-94`.

12. PROGRESS SOFTWARE. *Flowmon User Guide* [`https://docs.progress.com/bundle/progress-flowmon-os-12-3/page/topics/user-guide/Device-Description.html`]. [N.d.]. [Accessed 07-02-2024].

13. BRAUN, T; GÜNTER, M; KASUMI, M; KHALIL, I. Virtual private network architecture. *Charging and Accounting Technology for the Internet (Aug. 1, 1999)(VPNA)*. 1999.

14. LUOTONEN, Ari; ALTIS, Kevin. World-Wide Web proxies. *Computer Networks and ISDN Systems*. 1994, vol. 27, no. 2, pp. 147–154. ISSN 0169-7552. Available from DOI: `https://doi.org/10.1016/0169-7552(94)90128-7`. Selected Papers of the First World-Wide Web Conference.

15. LEWENBERG, Yoad; BACHRACH, Yoram; SOMPOLINSKY, Yonatan; ZOHAR, Aviv; ROSENSCHEIN, Jeffrey S. Bitcoin mining pools: A cooperative game theoretic analysis. In: *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*. 2015, pp. 919–927.

16. USSATH, Martin; CHENG, Feng; MEINEL, Christoph. Insights into Encrypted Network Connections: Analyzing Remote Desktop Protocol Traffic. In: *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing* (*PDP*). 2016, pp. 585–589. Available from DOI: 10.1109/PDP.2016.38.

17. KAUSAR, Md Abu; DHAKA, VS; SINGH, Sanjeev Kumar. Web crawler: a review. *International Journal of Computer Applications*. 2013, vol. 63, no. 2, pp. 31–36.

18. WANG, Yien; YANG, Jianhua. Ethical Hacking and Network Defense: Choose Your Best Network Vulnerability Scanning Tool. In: *2017 31st International Conference on Advanced Information Networking and Applications Workshops* (*WAINA*). 2017, pp. 110–113. Available from DOI: 10.1109/WAINA.2017.39.

19. CISCO SYSTEMS. *Cisco Firepower Release Notes, Version 6.5.0 - Upgrade the Software* [*Cisco Secure Firewall Management Center*] [https://www.cisco.com/c/en/us/td/docs/security/firepower/650/relnotes/firepower-release-notes-650/upgrade.html]. 2024. [Accessed 14-05-2024].

20. SEZNAM.CZ. *SeznamBot crawler* [https://napoveda.seznam.cz/en/seznambot-crawler/]. 2024. [Accessed 12-05-2024].

21. AMAZON WEB SERVICES. *Amazon Web Services IP Address Ranges* [https://ip-ranges.amazonaws.com/ip-ranges.json]. 2024. [Accessed May 12, 2024].

22. TEAM, IPFire.org - IPFire Development. *How to block Shodan scanners - The IPFire Documentation — ipfire.org* [https://www.ipfire.org/docs/configuration/firewall/blockshodan]. 2024. [Accessed 12-05-2024].

23. ZHAO, Bo. Web Scraping. In: *Encyclopedia of Big Data*. Ed. by SCHINTLER, Laurie A.; MCNEELY, Connie L. Cham: Springer International Publishing, 2017, pp. 1–3. ISBN 978-3-319-32001-4. Available from DOI: 10.1007/978-3-319-32001-4_483-1.

24. MAHTO, Deepak Kumar; SINGH, Lisha. A dive into Web Scraper world. In: *2016 3rd International Conference on Computing for Sustainable Global Development* (*INDIACom*). 2016, pp. 689–693.

25. KHDER, Moaiad. Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. *International Journal of Advances in Soft Computing and its Applications*. 2021, vol. 13, pp. 145–168. Available from DOI: 10.15849/IJASCA.211128.11.

26. SIRISURIYA, De S et al. A comparative study on web scraping. 2015. Available also from: https://api.semanticscholar.org/CorpusID:196204953.

27. PARVEZ, Momin Saniya; TASNEEM, Khan Shaista Agah; RAJENDRA, Shivankar Sneha; BODKE, Kalpana R. Analysis Of Different Web Data Extraction Techniques. In: *2018 International Conference on Smart City and Emerging Technology (ICSCET)*. 2018, pp. 1–7. Available from DOI: 10.1109/ICSCET.2018.8537333.

28. CLARK, James; DEROSE, Steve, et al. *XML path language (XPath)*. 1999. Available also from: http://new-design.renderx.com/files/demos/xmlspec/xpath/REC-xpath-19991116.pdf.

29. DARMAWAN, Irfan; MAULANA, Muhamad; GUNAWAN, Rohmat; WIDIYASONO, Nur. Evaluating Web Scraping Performance Using XPath, CSS Selector, Regular Expression, and HTML DOM With Multiprocessing Technical Applications. *JOIV: International Journal on Informatics Visualization*. 2022, vol. 6, no. 4, pp. 904–910.

30. KOSTER, Martijn; ILLYES, Gary; ZELLER, Henner; SASSMAN, Lizzi. *Robots Exclusion Protocol* [RFC 9309]. RFC Editor, 2022. Request for Comments, no. 9309. Available from DOI: 10.17487/RFC9309.

31. GLINZ, Martin. On non-functional requirements. In: *15th IEEE international requirements engineering conference (RE 2007)*. IEEE, 2007, pp. 21–26.

32. O'NEIL, Elizabeth J. Object/relational mapping 2008: hibernate and the entity data model (edm). In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. Vancouver, Canada: Association for Computing Machinery, 2008, pp. 1351–1356. SIGMOD '08. ISBN 9781605581026. Available from DOI: 10.1145/1376616.1376773.

33. SCRAPY. *Scrapy Documentation*. Available also from: https://docs.scrapy.org/en/latest/.