TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Archil Kristinashvili 194436IVSB

Creating Custom Solution for Visualizing Vulnerability Management Data at Pipedrive

Bachelor Thesis

Supervisor:	Kristian Kivimägi MsC
Co-Supervisor:	Kieren Nicôlas Lovell Lt CDR

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Archil Kristinashvili 194436IVSB

Turvanõrkuste visualiseerimise lahenduse implementeerimine Pipedrive'is

bakalaureusetöö

Juhendaja:	Kristian Kivimägi
	MsC
Kaasjuhendaja:	Kieren Niĉolas Lovell
	Lt CDR

Tallinn 2022

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Archil Kristinashvili

16.05.2022

Abstract

There are various vulnerability management tools available, and all of them offer different features and functionality. However, to meet the needs of all clients, these tools provide quite general approaches. The data accumulated in these tools scale along with the organization; Therefore, relatively big organizations can be overwhelmed and find it difficult to contextualize this amount of data. This research will focus on the example of Pipedrive, which uses, Tenable.io as the vulnerability management tool.

The primary goal of this thesis is to create an alternate exporting and visualization solution for data available from the well-known vulnerability management tool - Tenable.io. This solution will enable distinguishing visualizations related to vulnerabilities that affect asset groups of different business contexts. In addition, it would be possible to see the historical development of events. These features will allow information security teams quickly identify and contact units responsible for the vulnerability remediation. Furthermore, it would enable other units to track and evaluate their performance over time.

The result of this work is the combination of different services, tools, and dashboards. Some of these components, specifically the ones used to store and visualize this data, will be already existing, publicly available services. Meanwhile, the service that exports and categorizes the information is developed throughout this research.

Though the presented ideas and solutions are created and designed primarily for Pipedrive, they still might serve as a sample for other businesses.

The thesis is written in English and is 46 pages long, including 6 chapters, 20 figures and 2 tables.

List of abbreviations and terms

API	Application Programming Interface
CPE	Common Platform Enumeration
CRM	Customer Relationship Management
CVSS	Common Vulnerability Scoring System
DB	Database
ER	Entity Relationship
FQDN	Fully Qualified Domain Name
HTTP	Hypertext Transfer Protocol
IaaC	Infrastructure as a Code
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technologies
JSON	JavaScript Object Notation
MITRE	Massachusetts Institute of Technology Research and Engineering
NIST	National Institute for Standards and Technology
NRDB	Non-Relational Database
OS	Operating System
PCI DSS	Payment Card Industry Data Security Standard
QL	Query Language
RDB	Relational Database
REST	Representational State Transfer
SLA	Service Level Agreement
SOC	System and Organization Controls
SQL	Structured Query Language
TCP	Transmission Control Protocol
TSDB	Time-Series Database
URL	Uniform Resource Locator
UUID	Universal Unique Identifier
VPR	Vulnerability Priority Rating

Table of Contents

List of Figures

List of Tables

1 Introduction

Many of the recent happenings in the world have proven that cyber-attacks pose a severe threat to organizations. Nowadays, it has become especially simple to carry out those due to various publicly and privately available tools with incredibly low usage complexity. Cyber-attacks can have many different consequences, including financial losses, reputational damage, and loss of customer trust. In some cases, they might also lead to regulatory investigations and fines. One of the many reasons for cyber-attacks is unmitigated vulnerabilities. The word vulnerability can have many different meanings. By the broadest definition, vulnerability refers to the state of being exposed. From a security perspective, vulnerability resembles something that can be exploited by outside forces or actors to sidestep defenses, be it a breach in stronghold walls allowing the enemy in or a door left unlocked allowing a criminal inside [1]. While in the field of information technology, vulnerability means "A flaw or weakness in a computer system, its security procedures, internal controls, or design and implementation, which could be exploited to violate the system security policy." [2]

Almost any business today is either directly or indirectly affected by these vulnerabilities that carry a risk for normal day-to-day operations. One of the recent studies suggests that in 2019, 60 percent of data breaches could have been prevented by timely patching of vulnerabilities [3]. Due to the constantly increasing number of vulnerabilities nowadays, it has become impossible to patch all of them. To address this issue, a separate field of vulnerability management has been developed. Vulnerability management is not a standalone technology but is a combination of processes and practices aiming to identify, classify and mitigate risks connected to existing vulnerabilities. Due to the complexity of this approach, it is mainly aimed towards organizations and governmental bodies. [1]

The main focus of this paper is a contribution to the development of vulnerability management at Pipedrive by solving problems at a relatively lower technological layer. As the topic is quite complex and requires understanding several different fields, research is divided into two distinct but related studies. This paper focuses on the visualization aspect of vulnerability-related information and aims to automate all manual parts of this process. Meanwhile, another paper concentrates on establishing criteria for evaluating the performance of vulnerability checking solutions on critical infrastructure and comparing different approaches to vulnerability checking.

As a first step, a data categorization model will be constructed according to the vulnerability management priorities in Pipedrive. Based on these points, reasonable visualization goals will be defined. Afterward, multiple technologies related to storing and visualizing information will be studied. Once the most suitable ones are selected, a custom Tenable.io exporter will be developed. Finally, according to the previously defined visualization goals, multiple panels will be created.

Though principles and solutions provided in the research are mainly developed and designed for Pipedrive, they still can be used as an example for companies with similar vulnerability management priorities and IT infrastructure.

2 Problem Statement

The transitional period from a start-up to an enterprise organization is associated with a vast amount of challenges along with legal and financial responsibilities. The year 2021 was the beginning of Pipedrive's transition into an enterprise, which led to the restructuring of the information security team. The main result of those changes was a set of new requirements for the information security team. Pipedrive has already been certified with System and Organization Controls (SOC) 2 and 3 along with ISO27001:2013 [4][5]. The effect of these certifications is not permanent, these mandate to be maintained over time, which involves continuous improvement of the information security process. In addition, as these certifications do not necessarily fulfill their requirements, some refinements were requested by third parties, such as cyber insurance. One of the recent discussion subjects was the SLA for vulnerability remediation. Due to these circumstances, there has been a great emphasis on this topic from the stakeholders' side.

After several meetings with the information security team, visualization of vulnerabilities was determined to be one of the main limitations of the existing system. Firstly, the current vulnerability management tool, Tenable.io, does not support visualizations oriented on Pipedrive's IT infrastructure. Secondly, time-oriented visualizations in Tenable.io are very limited. Hence, there is no accurate representation of the historical development of events. To address these issues, the information security team developed a procedure for visualizing vulnerability-related information using spreadsheet tools. Because this procedure involves excessive manual and repetitive work, it results in inefficient use of the information security team's man-hours and shifts their attention away from security-related duties. Another issue with this approach is the intermittent state of visibility which slows down the team's response speed. In addition, it makes the process dependent on one person's efforts, which might become a blocker for the entire team. Finally, like every other manual process, the risk of human error is raised.

There is a significant relationship between the effectiveness of visualization and vulnerability management. Clear visualization enables organizations to understand their risks and vulnerabilities better, thus assisting them with the identification and mitigation of those. Additionally, effective visualization can also help organizations with a better understanding of how their systems are interacting with each other. There have been many real-life examples where visualization played crucial roles in vulnerability management. Widely spread critical vulnerabilities like Log4J or Heartbleed affected various system components in the different organizations. Though not all components of these systems held sensitive data, responsible teams were still overwhelmed by the amount of information received by vulnerability scanners. Therefore visualization was crucial in identifying which servers were vulnerable and which data was potentially at risk. This information allowed the companies to quickly patch the vulnerability on the vulnerable servers, and protect their users' data.

The primary goal of this paper is to develop an alternative, automated method for processing and visualizing vulnerability management data from Tenable.io in a way that vulnerabilities of different domains are easily distinguishable. The architecture of this solution must be suitable for Pipedrive's IT infrastructure and should leverage existing tools upon possibility. The dashboards provided by the solution should be capable of displaying the current and historical position in vulnerability management. Visualizations should be continuous to enable Pipedrive's information security team to respond immediately. Finally, the components should be modifiable and modular for future development.

In addition, if the solution is successfully developed and the results are reasonable, the thesis should answer the following questions:

- What are the differences between the developed solution and Tenable.io?
- What benefits does the developed solution bring?
- Who can make use of the solution?
- What are the limitations, and what could be improved?

3 Background Information

3.1 Data Visualization

We live in times when technological advancements occur at unprecedented speeds. Already back in 1965, it was observed by Gordon Moore that the number of transistors in a dense integrated circuit doubled about every two years [6]. In reality, this rule is not only applicable to the number of transistors but reveals the general development of the technological field. Thanks to the increased computational powers and popularization of the internet, the amount of produced data has grown exponentially. It was estimated that by 2020 about 1.7 megabytes of new information was being created every second for every human being on the planet [7]. Yet most of this information does not represent anything meaningful as it is in raw form. To retrieve any practical knowledge from this data, it should be a subject for an analysis. Since the amounts of data have grown and the cognitive abilities of humans have remained almost the same, a challenge has been formed. In addition, various sources and forms of this data complicated the processing even further. Therefore, the need for data visualization has been significantly up-trending.

Data visualization is the representation of data through the use of common graphics, such as charts, plots, info-graphics, and even animations. These easy-to-understand visual representations reveal complex data relationships and data-driven insights [8]. To emphasize how effective visualization is, several examples can be demonstrated. One research suggests that visuals are processed 60,000 times faster than text by the human brain. In addition, 90 percent of information transmitted to the brain is visual, as humans evolved over millennia to respond to visual information long before developing the ability to read the text. [9]

This becomes especially important in the business world. According to another study, usage of visual language shortened business meetings by 24 percent and increased productivity within the organization [10]. In fact, increased productivity was not the only benefit, as visualized data impacted the decision-making process as well. According to the same study, there was a 21 percent increase in the chance of reaching a consensus for the group that used visuals in comparison to one that did not [10].

3.2 Pipedrive

Pipedrive is an IT company founded in 2010 in Estonia. The company follows the software as a service monetization model, offering a CRM solution. The main principle of the software has always been simplicity and ease of use. Its intuitive design and kanban-based approach to sales make Pipedrive a very compelling product for small to medium businesses and startups. Currently, the company serves more than 100 000 different customers worldwide [11]. Each subscribed customer can have many users. Most of the clientele are other businesses that have some or all employees using the tool, making the total number of software users even greater.

3.2.1 Overview of IT Infrastructure Practices

To understand vulnerability management at Pipedrive, it is necessary to have a basic knowledge of the IT infrastructure at the company. Only this way, it is possible to contextualize the IT assets of the business and adequately categorize them. Said shortly, there are three main characteristics that would describe Pipedrives infrastructure:

1. Infrastructure as a Code approach
2. Containerization
3. Cloud computing

The first one is the Infrastructure as a code (IaaC) approach. IaaC is a technology for automating resource provisioning along with the deployment and configuration of services. In addition, it centralizes the management of those so that there are no inconsistencies between the work of different engineers. Finally, it is considerably fast and highly scalable in comparison to the manual approach. In the case of Pipedrive, primary IaaC tools are Ansible, Chef, and Terraform

The second characteristic is containerization, which due to a large number of services, also requires an orchestration technology. Pipedrive's product is available through a vast amount of micro-services. These micro-services undergo constant improvements and implementation of new features. It would not be feasible to run these services directly on machines due to many reasons. Binding services to the machine environment and network might not always work as intended and might cause interference between services. In addition, it would be overly expensive. Therefore, Pipedrive uses Docker and Kubernetes. Docker is the de-facto standard of containerization, while Kubernetes is a container orchestration technology. Within this approach, Kubernetes runs on multiple nodes (machines), which are used to manifest various services. Apart from that, Kubernetes also provides these services with load-balancing, health-checking, networking, and many other features.

The third characteristic relates to the usage of the cloud for provisioning computational and storage resources. This eliminates the supplementary cost and creates options for scalability by allowing more rapid provisioning of these resources. In addition, it enables the distribution of services in multiple regions, hence providing less product latency to the customers.

To summarize, Pipedrive's IT infrastructure consists of various nodes provisioned in multiple regions. In each region, an equivalent set of services is operating. These services exist as pods (smallest deployable units of Kubernetes) and are deployed using IaaC tools. The role of the service determines the sensitivity of stored data and the exposure to the public network. Therefore, throughout this research, each service will be considered a separate informational asset.

3.2.2 Overview of Vulnerability Management Tools

Vulnerability management at Pipedrive consists of multiple processes and practices. Since this paper only aims to solve problems at the technological layer, the discussion will only go around used tools. The main tools used by Pipedrive's information security team are:

- Snyk
- Dependabot
- Tenable.io

In this research, the main focus will be on Tenable.io, a cloud-based vulnerability management solution. Tenable.io leverages the power of Nessus, which is a well-known network vulnerability scanner. Nessus allows scans for software misconfigurations and identifies open ports along with the version of software associated with them. In addition, it searches for vulnerabilities that could allow manipulation or access to sensitive data on a system. Nessus also identifies possible vectors for denial of service attacks against the TCP/IP stack. Finally, it has the capability of PCI DSS audits. [12] Apart from vulnerability scanning, Tenable.io also offers various dashboards for visibility. Tenable.io resources are available either through a graphical web user interface or API.

Withing the current implementation of this tool at Pipedrive, Tenable.io is used to discover and scan assets. Scanning is done in two distinct manners. The first is the traditional network scan, while the second is the agent-based scan. According to Tenable.io's documentation, the difference between these two is the following:

- The traditional scan is done by an external Nessus scanner that reaches the target instance through the network.
- Agent scan is done directly on the instance regardless of network connectivity. Afterward, results are reported to Tenable.io.

Since both of these scans have their benefits and limitations, they have different use cases. According to the current configuration at Pipedrive, Agents are mainly used for scanning vulnerabilities on workstations (employees' laptops), while network scans are used for servers. Though configuration might change based on the results discovered in the research conducted on a parallel basis.

3.3 Tenable.io

3.3.1 API Background

Before developing an exporter, it is necessary to consider the specifications of the API that will provide the data. In the current scenario, data will be collected from Tenable.io API, which, according to Tenable.io, has all the vulnerability management capabilities that they offer [13]. The API is organized around resources, which are individual objects that can be manipulated. Each resource has its URL and is accessed using the standard HTTP verbs. As the available data is the property of the organization, it requires authentication through API keys.

The Tenable.io API documentation is very comprehensive and provides detailed information on how to use the API to access Tenable.io data and services. The documentation includes a theoretical guide and a reference guide. The theoretical guide covers the basics of using the API, including how to authenticate and make requests. The reference guide, which is based on OpenAPI 3 specification, provides detailed information on the API endpoints and query parameters [13]. In addition, the reference guide provides code snippets written in more than ten programming languages together with all examples of response models and their descriptions.

Like most solutions, Tenable.io API has several limitations that should be taken into account when using it. Firstly, as it is a cloud solution, it is rate-limited, meaning that only a certain number of requests can be made per minute [14]. This can be a problem if trying to do a large number of scans or pull a lot of data at once. Secondly, the API might not always be 100 percent accurate, and some data may be missing or incorrect, especially when it comes to exporting large chunks of data. Finally, the API can be slow at times, and users may have to wait a while for their data to be returned.

3.3.2 API Data Structure

Tenable.io API has various endpoints for accessing and modifying resources such as agents, assets, vulnerabilities, audit logs, plugins, policies, scans, et cetera. As this research only focuses on the visualization of vulnerabilities and affected assets, these two will be the only ones used. Details regarding both resources are available in the most common RESTful API format - JSON.

Vulnerability

The figure 1 demonstrates the JSON object corresponding to the vulnerability discovered using Tenable.io. Since not all the key-value pairs provide relevant information, most of them are not shown in this example.

```
{
  "asset": {
    "hostname": "abc.pipedrive.com",
    "uuid": "39461369-79da-455b-9edd-2c479fe0526e"
  },
  "output": "scan output",
  "plugin": {
    "cpe": ["cpe:/a:apache:log4j"],
    "name": "Apache Log4j 1.x Multiple Vulnerabilities"
  },
  "severity": "critical",
  "severity_modification_type": "NONE",
  "state": "OPEN"
}
```

Figure 1. *JSON of vulnerability response object from Tenable.io (only relevant fields shown)*

- Fields "hostname" and "UUID" (Universal Unique Identifier) will be used to refer to the asset object.
- Field "CPE" stands for Common Platform Enumeration. It is a project formerly developed by MITRE and later transferred under the responsibility of the U.S. National Institute for Standards and Technology (NIST) [15]. CPE standardizes the method of describing and identifying classes of applications, operating systems, and hardware devices present among an enterprise's computing assets [16]. It will be used to determine the affected application on the workstations. Though, in many cases, CPE can not accurately represent the vulnerable component.

- Field "output" contains the output from the vulnerability scanner and will be used to select the correct component if there are multiple CPEs.
- Fields "name" and "severity" will be used directly.
- Field "severity_modification_type" denotes whether the vulnerability was accepted and will be used to filter out accepted vulnerabilities.
- Field "state" refers to the state of vulnerability and will be used to filter out remediated vulnerabilities.

Asset

In the case of assets, the only relevant information is the categories assigned to them. As the IT infrastructure of every company is different, Tenable.io is not capable of classifying the discovered assets according to their use case. In addition, Tenable.io does not know whether the asset is exposed to the public or if it contains any sensitive information. Therefore, the classification process should be done manually. One way to solve this problem is to write complex regular expressions to identify the assets by their fully qualified domain name. Though FQDNs are not always similar and consistent and regular expressions will add an unnecessary layer of complexity to the solution. Consequently, to achieve a more maintainable and scalable solution, asset categorization logic should be located maximally close and ideally within the source of the data.

In fact, Tenable.io already offers the functionality of adding business context to the assets by using custom tags. Tags are key-value pairs that can be manually or automatically assigned based on the asset's hostname, operating system, network, or any other property of the asset.

Combining the data categorization model discussed in the upcoming chapter 4.1 with Tenable.io's tagging functionality creates the possibility of fetching a similar JSON response, as displayed in the following example.

- Field "ID" is identical to the "UUID" field in the vulnerability JSON model and, as already mentioned, will be used for uniquely referring to assets.
- Field "tags" is an array of key-value objects that are pre-set by the information security team.

```
{
  "id": "83413052-7f9c-42f7-b06f-359127824fa7",
  "tags": [
    {
      "key": "Type",
      "value": "Workstations"
    },
    {
      "key": "Workstations",
      "value": "macOS"
    }
  ]
}
```

Figure 2. *JSON of asset response object from Tenable.io (only relevant fields shown)*

3.4 Databases

Exporting and reformatting vulnerability and asset-related data is the first step toward visualizing it. As Tenable.io visualization only focuses on the current condition of the vulnerability management position, it becomes impossible to create dashboards related to the number of vulnerabilities affecting assets at a certain point in time. Tracking the changes in the relatively complex systems requires visualizations that are time-oriented, hence need time-series data. According to the classical definition, a time series is simply a sequence of numbers collected at regular intervals over a period of time. More generally, a time series is a sequence of data points (not necessarily numbers). And typically, time series consist of successive measurements made over a time interval [17]. Though it is possible to store data directly in the exporting service, it is not a feasible solution due to various reasons. First, delivering this data to visualization technology would become more complex or impossible. Second, the performance would be favorably inferior to any already existing solution. Last but not least, there will be a chance of losing data while implementing new features. All reasons combined, it is more reasonable to store time-series externally. There are multiple methods to store time-series data, such as relational and non-relational databases, which, in the best-case scenario, can also be a dedicated to time-series measurements.

3.4.1 Relational Databases

The first approach toward storing time series is using a relational database. There is a vast amount of free and paid RDB products. The most common examples are MySQL, PostgreSQL, Microsoft SQL Server, or even a cloud-based solution available directly from the cloud provider. RDBs are capable of storing different data types, such as numbers, text, binary, and many more. Since RDBs also support timestamps, it is possible to create tables where one of the columns is a timestamp, and hence rows turn into the measurements throughout the time [17]. Since additional features like performance, high availability, and scalability are not relevant due to an insignificant volume of data (insignificant only in terms of storing, not visualizing), a thorough comparison of these solutions is outside the scope of this paper. Instead, for simplicity MySQL is chosen as a representative of RDB options.

3.4.2 Non-Relational Databases

Different data structure within a database provides various features, limitations, and use cases. In many scenarios, SQL was not the best fit, and a need for different database technologies emerged. As these technologies were significantly dissimilar to the traditional RDBs, they are referred to as NRDBs or NoSQL, which correspondingly are abbreviations for "Non-Relational Databases" And "Not Only SQL." [18] Some of the primary reasons to use NRDBs are [19]:

- Avoidance of unnecessary complexity
- High throughput
- Scaling data
- Avoidance of expensive object-relational mapping
- Requirements for cloud computing

Unlike SQL, NoSQL databases are more distinct from each other. Some of them can have a wide use case scenarios, while others have their scope narrowed down. The type of these databases is defined by the data model they store. Four primary categories of NoSQL databases are

1. Key-value store
2. Column store
3. Document store
4. Graph database

In addition to these four types of databases, lately (from 2020 to 2022), the popularity of time-series databases (TSDBs) has shown rapid growth. Figure 3 illustrates the trend in database technologies. The chart is provided by DB-Engines, a project that collects and presents information on database management systems. [20]

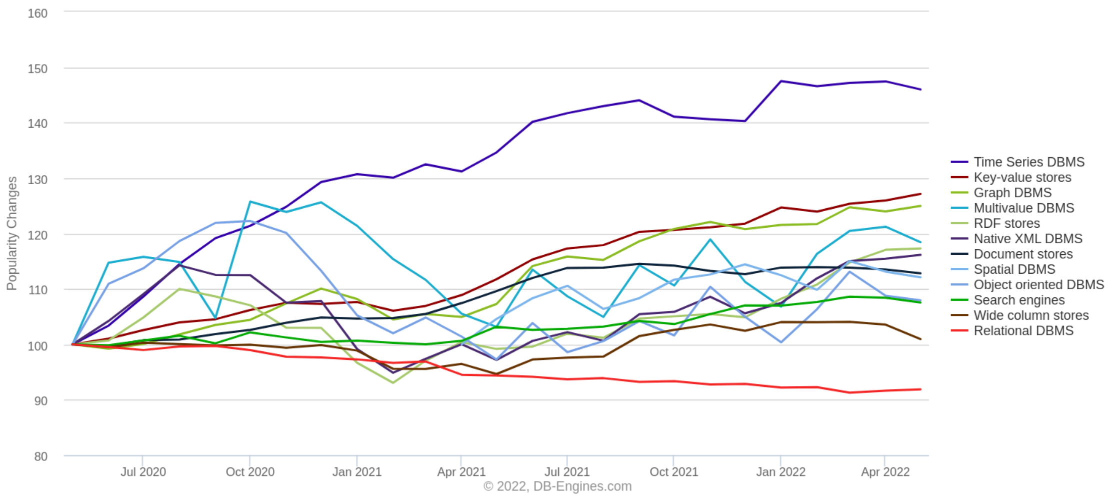


Figure 3. *Historical trend of the database categories' popularity according to mentions on websites, job positions, and professional networks*

A time-series database is a database that has been designed to operate with time-series data. There are numerous database management systems that include time-series data storage, processing, querying, and analytical capabilities. [19] This chapter will briefly describe two well-known databases that are also currently used by Pipedrive.

Prometheus

Prometheus is an open-source system monitoring and alerting toolkit originally built at SoundCloud. Its architecture is a combination of different components that retrieve, store, and access data. For retrieving data, Prometheus uses a scraping module. This module requests data and retrieves metrics from in-house developed or third-party applications via client libraries and exporters. For storing metrics, Prometheus uses a multi-dimensional data model, where each Time Series is defined by a metric name and a set of key-value dimensions called labels. As for the values of the measurements, Prometheus only supports numeric data types. These time series are accessible through PromQL, which is a flexible query language for selecting, filtering, and aggregating the time series data. [21][19]

InfluxDB

InfluxDB is an open-source time-series database developed by InfluxData. Written in Go and optimized for fast, high-availability storage and retrieval of time series data in fields such as; operations monitoring, application metrics, Internet of Things sensor data, and real-time analytics. [22]

It uses a Time-Structured Merge Tree to store sorted and compressed Time Series data in a columnar format. Subscriber endpoints are used to write data to InfluxDB. Every record in the database has a time column defined. The remaining attributes of a record are referred to as fields. A field key identifies each field, and the field value represents the actual data associated with a timestamp. The field value has a data type, such as string, float, integer, etc. All the pairs of field keys and field values make up a fieldset. Besides fields, InfluxDB also has tags representing metadata about the records stored in the database. InfluxDB's data model does not support the JOIN operation. [19]

4 Methodology

4.1 Data Categorization Model

The current implementation of Tenable.io does not provide separate dashboards for visualizing vulnerabilities affecting different types of assets. This creates the difficulty of reporting the uncovered information to the appropriate team. Within the present team structure of Pipedrive, vulnerabilities discovered on workstations (laptops) are forwarded to the IT Operations team, while vulnerabilities discovered on servers are communicated with the IT Infrastructure team. Subsequently, based on the workstation operating system or the purpose of the server, the right sub-team or specialists are selected to resolve the issue. Therefore, based on these needs, affected assets are proposed to be categorized into two main groups - servers and workstations. Each group is divided into sub-groups by the provided service and the operating system.

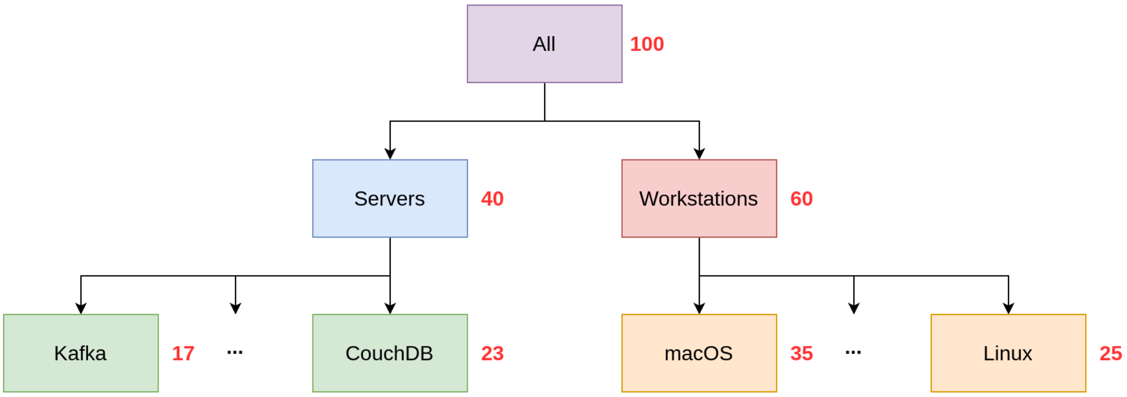


Figure 4. *Affected asset categorization model according to Pipedrive's information security teams needs*

Unlike servers which are Kubernetes pods and provide only one service, workstations come with multiple applications installed. Hence, it is reasonable to divide each OS into a deeper sub-group of affected applications.

Categorizing vulnerabilities is relatively simple and is done only according to severity level, which is equal to one from the following from the list:

- Critical
- High
- Medium

- Low

4.2 Defining Visualization Goals

It should be clear that just transforming data into graphics will not directly make it evident and usable. The first step of the data visualization is setting the goals. There can be two major goals for the visualization: data exploration and data explanation [23]. Exploration is used when there are no clear hints about what data represents. In this scenario, visualization might uncover the true meaning behind the data. Meanwhile, the explanation is used when the meaning of data is already known and should be presented to others in an understandable form [23]. This thesis will mainly focus on the data explanation as its structure is precisely described by the Tenable.io documentation. In addition, there is a set of certain practices that need to be considered to achieve effective visualization. In this work, one of the main guidances will be “Principles of Effective Data Visualization” by Stephen R. Midway [24].

The primary assignment is to provide automated visualization for the Tenable.io data. The visualization should be reasonable and capable of providing all the necessary details to the information security team. However, it is up to the information security team to decide which factors are most important for their needs.

Stephen R. Midway, in his research Principles of Effective Data Visualization, suggests the ten most influential principles to consider [24]. Not all of those can be relevant to this individual scenario, though the rest provide valuable guidance. Here is the list of the considered principles throughout the research:

1. Diagram First
2. Use an Effective Geometry and Show Data
3. Colors Always mean Something
4. Simple visuals, Detailed Captions

The first and the most important principle states to prioritize the information that needs to be shared before creating visuals. It is necessary to focus on the end-message first so that no software limitations occur in the future. Though this does not necessarily mean thinking of geometries, the main goal remains the definition of the final visual objective. This could be achieved either by the imagination or by using relatively primitive tools/software [24].

The second (on this list) suggests that picking the correct geometry is the key to making the thinking process happen naturally. As there are many scenarios when multiple geometries can be used to visualize the same data-set. Therefore it is necessary to pick the most suitable one, or additionally, in many cases, it makes more sense to use multiple ones to highlight the different details of the same data. [24]

As suggested by other principles, it is also essential to have detailed captions and an intuitive color scheme. Therefore, all of the above-mentioned visuals will include descriptions. As for color, red, orange, yellow, and green will be mapped to the vulnerabilities in correspondence to critical, high, medium, and low severities.

The case covered by this paper will include assets of many types and groups affected by various vulnerabilities of different severity. Currently, there are two asset types: servers and workstations. To clarify, servers are the assets taking part in providing service to customers, while workstations are laptops utilized by Pipedrive employees. These two are of a completely different domain and should be handled in distinct manners. For example, servers are parts of separate asset groups based on the specific services they provide. Depending on the roles of these services, the information security team can decide whether they may contain sensitive data. As for workstations, they are grouped according to the operating system they use. These are macOS, Linux (Ubuntu), and Windows.

- The first visual to have is the time-series graph that would indicate the total number of vulnerabilities that can be split into different severities such as critical, high, medium, and low. This would be the only graph that combines both - server and workstations.
- As it is complicated to understand a graph with many different plots, there also is a need for a relatively simple visualization that will exhibit affected server groups. To simplify the view, time-series data can be reduced and sent into pie chart visualization form. This pie chart will represent the current number of vulnerabilities per server group based on certain severity. As critical vulnerabilities are the highest priority, only this severity should have a separate pie chart. For the state vulnerabilities of all other severities a separate pie chart can be created. This pie chart will represent the combined number of vulnerabilities per asset group.
- Since there is a fixed number of operating systems, a stacked bar chart can be used for the number of workstation vulnerabilities. In this chart. One bar will be dedicated to each operating system. Each bar will be split into three parts that will represent the number of critical, high, and medium vulnerabilities.
- A table will be used to provide something similar for the servers. In this table, server groups will represent the titles of rows, while severities will define the columns.
- The servers will have one more table that will display the number of most common vulnerabilities and their severity.

- For a more detailed overview of workstations state, another table and bar chart will be provided. In this table, the list of affected applications will represent the titles of rows, while severities will be titles for columns. The bar chart will visualize the same information to decrease inspection time.

4.3 Selecting Databases

When it comes to choosing between databases, there are several things to consider. Specifically, the database should be capable of storing data both in textual and numeric format. In addition, it would be a great advantage if the database's data structure and query language were to be optimized for the time-series data.

Table 1. *Satisfaction of general requirements by the discussed databases*

Requirement	**SQL**	**Prometheus**	**InfluxDB**
Optimized for time-series	N	Y	Y
Measurements in numerics	Y	Y	Y
Measurements in strings	Y	N	Y

This paper mainly aims to develop a solution that is suitable for Pipedrive. Therefore, apart from general requirements, there are two additional considerable factors. Firstly, there must be multiple people with a good knowledge of this database within the observability team. Secondly, the teams responsible for remediating vulnerabilities should be sufficiently familiar with the corresponding querying language.

Table 2. *Satisfaction of Pipedrive's requirements by the discussed databases*

Pipedrive's Requirement	**SQL**	**Prometheus**	**InfluxDB**
Supported by Observability	Y	Y	N
Famial QL	Y	Y	N

According to discussion along with table 1, it became evident that, generally, InfluxDB was the most suitable solution. Mainly because InfluxDB can support both - numeric and string measurements. In addition, unlike Prometheus, InfluxDB is a push-based system; therefore, there is no need to create extra metrics for the points in time when no new data was exported.

Regardless, due to the requirements from Pipedrive described in table 2, it was decided to continue with more acquainted technologies. And to cover the complete range of specified general requirements Prometheus and MySQL are used simultaneously.

4.4 Selecting Visualisation Technology

In fact, there is one more exceptionally important principle of effective visualization that has not been discussed in chapter 4.2. This principle describes the importance of the right software choice. It should be no surprise, as creating effective visuals requires good command and understanding of particular software[24]. Since this paper focuses on finding the solution focused on Pipedrive, there is only one candidate - Grafana.

Regardless, this does not leave this research in a problematic situation as Grafana supports all mentioned databases and the vast amount of visualization forms. In addition, it is likely to be utilized by any organization with cloud-native infrastructure, or since it is free and open-source, it can be easily implemented. Yet, it is to be verified that Grafana is capable of visualizing geometries defined in chapter 4.2. The geometries are relatively simple as they only include time-series graphs, pie charts, and stacked bar charts and tables. According to the documentation, Grafana provides a Time-series panel which is the default and primary way to visualize time series data. In addition, it has several rendering options, such as lines, the path of dots, or a series of bars. Besides, it states that it has the capability of displaying almost any time-series data [25]. Bar charts and Pie charts are also mentioned in Grafana documentation, and according to it, both are capable of displaying reduced time series[26][27]. Though it does not explicitly mention in the documentation that bar charts can be stacked, there have been various community discussions stating the possibility of this option.

4.5 Developing Exporter

4.5.1 Querying Data

Tenable.io API has various endpoints for accessing and modifying resources. Since it was determined that the scope of this exporter is to show historical data for vulnerabilities and respective assets, these two are the only data types that will be queried. Retrieving vulnerabilities or assets lists from Tenable.io is not a very straightforward process. Big organizations have a vast amount of assets that can be affected by even more vulnerabilities. Numbers can grow so big that it becomes impossible for Tenable.io to send all of this information in one API call. To address this issue, Tenable uses the approach of exporting data into several chunks and making them available for download separately. Though this improves performance on Tenable's side, it complicates the process for developers and severely increases the required time to access these resources. According to the documentation, this data can be accessed in the following sequence [28]:

1. Export file Generation
2. Querying for the export generation status and chunk identification information
3. Downloading completed export chunks

Though, since API does not directly respond with the progress status, there is a need to implement the following logic into the code.

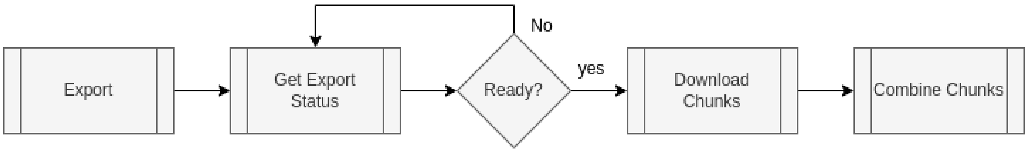


Figure 5. *Sequence of actions to export asset and vulnerability data from Tenable.io*

When exporting vulnerabilities, two parameters must be taken into account - state and severity. Since the information security team is only interested in active vulnerabilities, only they are selected. Tenable.io uses Common Vulnerability Scoring System (CVSS) and a dynamic Tenable-calculated Vulnerability Priority Rating (VPR) to quantify the risk and urgency of vulnerability [29]. As vulnerability management in organizations mainly targets critical, high, and medium severity vulnerabilities, a corresponding filter is applied. On the other hand, assets do not need any special filters.

4.5.2 Exporting data

Working directly with JSON responses is immensely inconvenient and does not allow fast and logical information processing. Therefore, there is a need for an additional processing structure of data. According to the analysis of the response data structure in chapter 3.3.2, an entity-relationship diagram is created (see figure 6).

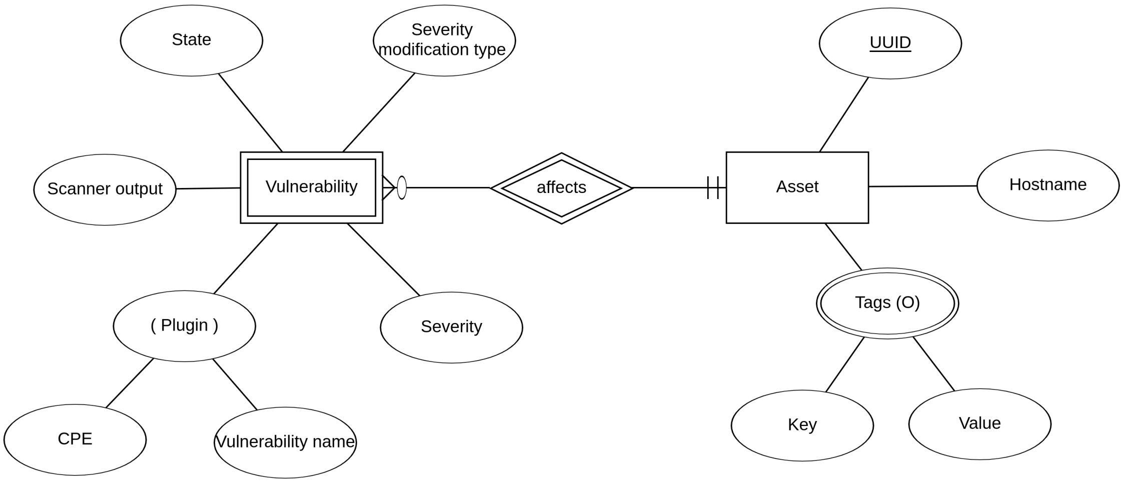


Figure 6. *ER Diagram of Data that is Exported from Tenable.io*

The following step after structuring information is to calculate the metrics. This solution will include two types of metrics. One will be the number of server vulnerabilities, and the second will be the number of workstation vulnerabilities. As the metrics are numeric, the Prometheus database is preferred.

Prometheus supports four types of metrics: counter, gauge, histogram, and summary [30]. The counter is a constantly increasing value, and its primary purpose is to retain the data in case the source restarts. Gauge, on the other hand, can both decrease and increase. Histogram and Summary provide several occurrences within a quantile for metrics that would not make sense to be exported as a gauge or counter. In addition, every Prometheus metric supports labels that can be used for providing information like a tag of an affected asset or a vulnerability severity. The only metric available throughout this research is the number of vulnerabilities; Therefore, only the gauge metric is used.

For each vulnerable server, separate measurements will be taken. This measurement will represent the number of vulnerabilities and would have a unique set of the following labels:

- Severity
- Instance (hostname of the server)
- Service

For workstations, similar logic will apply, but unique set of labels will be:

- Severity
- Instance (hostname of the workstation)
- OS
- App

Figures 7 and 8 represent the vulnerabilities in the form they are available for and from Prometheus.

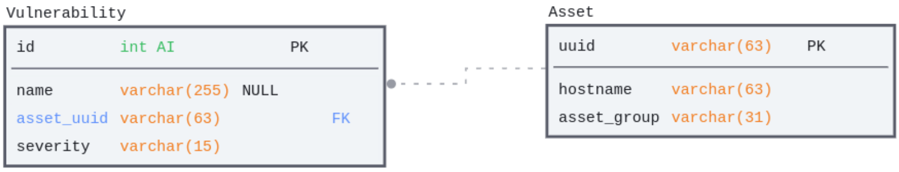
```
tenable_server_vuln_count{
  severity="high",
  instance="something.pipedrive.com",
  service="nginx"
} 1
```

Figure 7. *Prometheus metric for affected servers*

```
tenable_workstation_vuln_count{
  severity="high",
  instance="someones-macbook",
  os="macOS",
  app="mozilla firefox"
} 2
```

Figure 8. *Prometheus metric for affected workstations*

In addition to these metrics, two tables exist in the MySQL database. One is for the vulnerabilities, while the second is for assets. The primary purpose of these tables is to store the names of vulnerabilities.



Vulnerability

id	int AI	PK	
name	varchar(255)	NULL	
asset_uuid	varchar(63)		FK
severity	varchar(15)		

Asset

uuid	varchar(63)	PK
hostname	varchar(63)	
asset_group	varchar(31)	

Figure 9. *SQL Diagram of vulnerability and asset data*

4.6 Final Architecture

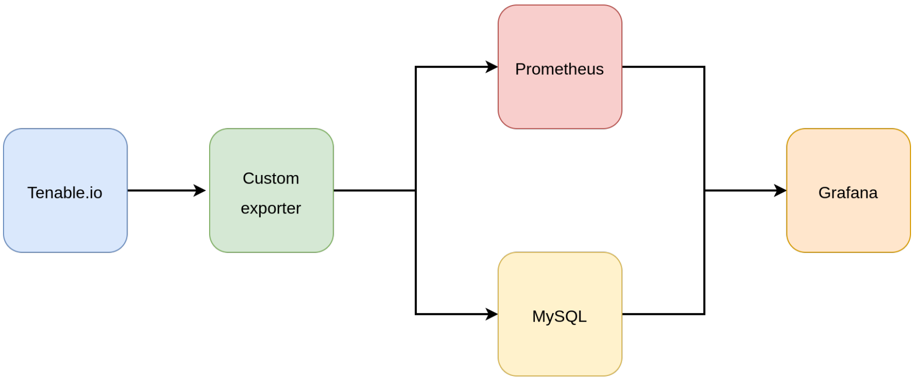


Figure 10. *Solution architecture*

The final architecture of the solution is illustrated in the figure 10 and works in the following way:

1. Exporter makes the set of API calls to generate and retrieve required data.
2. Once the data is available, it is processed by the exporter, and corresponding metrics are generated.
3. Metrics showing the number of vulnerabilities on particular services, operating systems, and applications are pulled by Prometheus. Meanwhile, the name, severity, and affected asset group of vulnerabilities are stored in MySQL.
4. Grafana uses both of these databases as sources of data. It queries them and according to the provided responses, generates dashboards.

5 Results

Two primary products of this research are the Tenable.io exporter and Grafana dashboards that visualize the exported data. The exporter is written in Golang, the same language used in Prometheus, Grafana, and other popular observability products. Though the exporter is capable of categorizing data according to specifications indicated in chapter 4.1, it requires two different database technologies to store them. This is mainly because Pipedrive widely uses no single technology that supports numeric, textual, and time-series data types. On the other hand, visualizations in Grafana are almost identical to the specifications defined in chapter 4.2.

5.1 Exporter

The exporter is written in a manner that there is a possibility of implementing new metrics. Precisely, it works in the way that it first gathers all the necessary data and transforms it into a more processable format. After that, this information is transmitted to the independent functions, where each generates measurements for one specific metric. Therefore, there is an opportunity to extend the exporter's functionality by adding new functions that calculate metrics. While testing the exporter, it became clear that Tenable.io's API has some restrictions. Specifically, it does not allow exporting the required data at any desired interval of time due to the large load on its server. The exact allowed duration between exports has not been determined. Though, during the tests, 3-hour intervals appeared to be on the safe side.

5.2 Dashboards

Even though the back-end part of the solution has many different limitations, they almost are not reflected in the visualizations. To precisely describe created panels and their use cases, this section explains multiple figures. Since the actual names and numbers of affected services and applications are confidential information, dashboards are filled with dummy data. Regardless, this information should be enough to present the accomplishments understandably.

Another great advantage of visualizing these metrics in Grafana is the ease of creation and customization of panels. In Pipedrive, each department has its own Grafana workspace and user accounts (referred to as org in Grafana). Therefore, in addition to relatively general-purpose panels presented in the upcoming figures, users from different departments can create visuals dedicated to their services. Furthermore, they can leverage Grafana's explore functionality for quickly generating single-time visualizations. This eliminates the need for constant unnecessary communication with the information security team, hence saving both teams' time.

A time-series line shown in figure 11 displays the current and historical number of vulnerabilities of each severity along with the total number. There are two different samples of this panel. One is for server vulnerabilities, while the second is for workstation vulnerabilities. These historical measurements can be utilized by the information security team to separately evaluate their overall position in two distinct domains. In addition, these graphs can directly be used by the IT Operations and IT Infrastructure teams to have an overview of their progress.

The server dashboard contains two pie chart panels. The first pie chart panel shown in figure 12 displays the total number of vulnerabilities affecting certain services. Meanwhile, the second pie chart in the figure 13 has a filtering option and can display vulnerability only of a certain severity. This panel can be used to identify the most affected servers and concentrate on remediating vulnerabilities of the highest severity.

The first table on the server dashboard on figure 14 shows the number of critical, high, medium, and low severity vulnerabilities present in each asset group, while the second table on the server dashboard on figure 15 shows the number of each vulnerability type. As the vulnerability names are long and complex strings, the Prometheus database does not support them. Thus, the second table is the only table that pulls the stored vulnerability name information from the MySQL database.

Two similar pie charts exist on the workstation dashboard. In comparison to the server dashboards, these pie charts display the vulnerabilities of the applications rather than services. One primary difference is that vulnerabilities on workstations can affect various applications. Therefore the number of affected applications is very high. To keep the dashboard less bloated only the top 20 entries are displayed. Similarly to the server's dashboard, figure 16 displays the total number, while figure 17 can be filtered according to severity.

Three stacked bar charts exist on the workstation dashboard. The first three bar charts in figure 18 visualize the distribution of vulnerabilities of different severities across different operating systems. The last stacked bar chart in figure 19 provides the ratio of vulnerabilities by different severities present in each type of operating system. These dashboards allow contacting the right group of employees.

The single table on the workstation dashboard in figure 20 shows the number of critical, high, medium, and low severity vulnerabilities present in each application. This table is an addition to the pie chart panels and provides more detailed analytics of the most problematic applications.

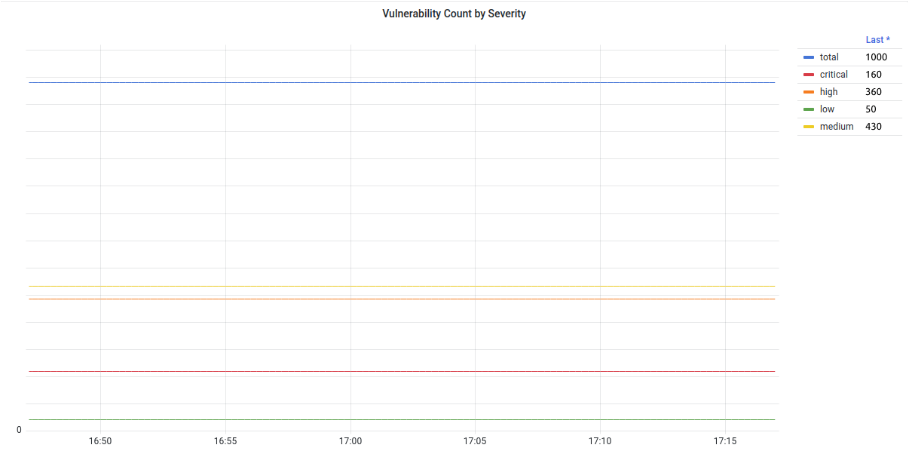


Figure 11. *Time-series line graph of number of vulnerabilities*

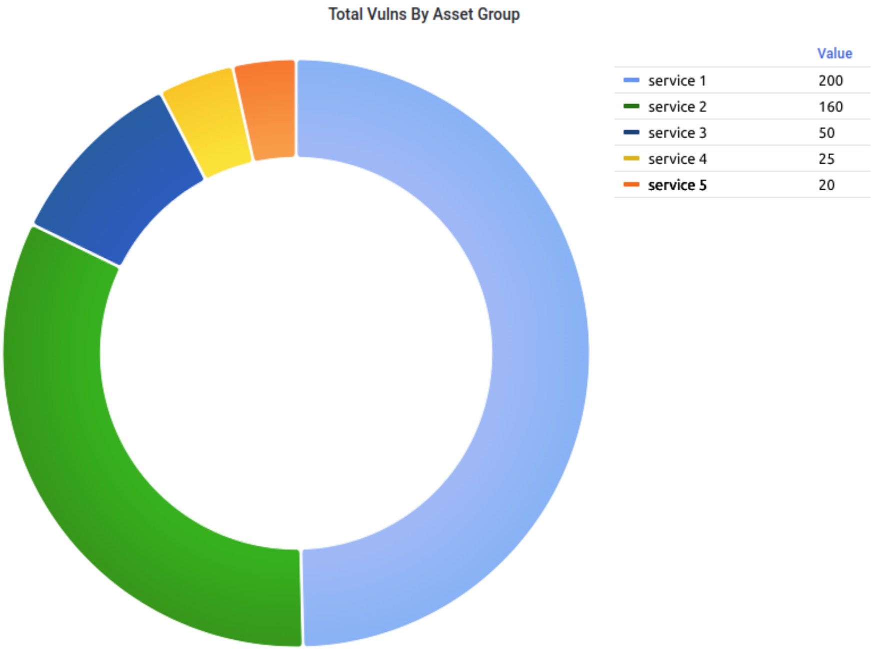


Figure 12. *Pie chart of total number of vulnerabilities by asset group*

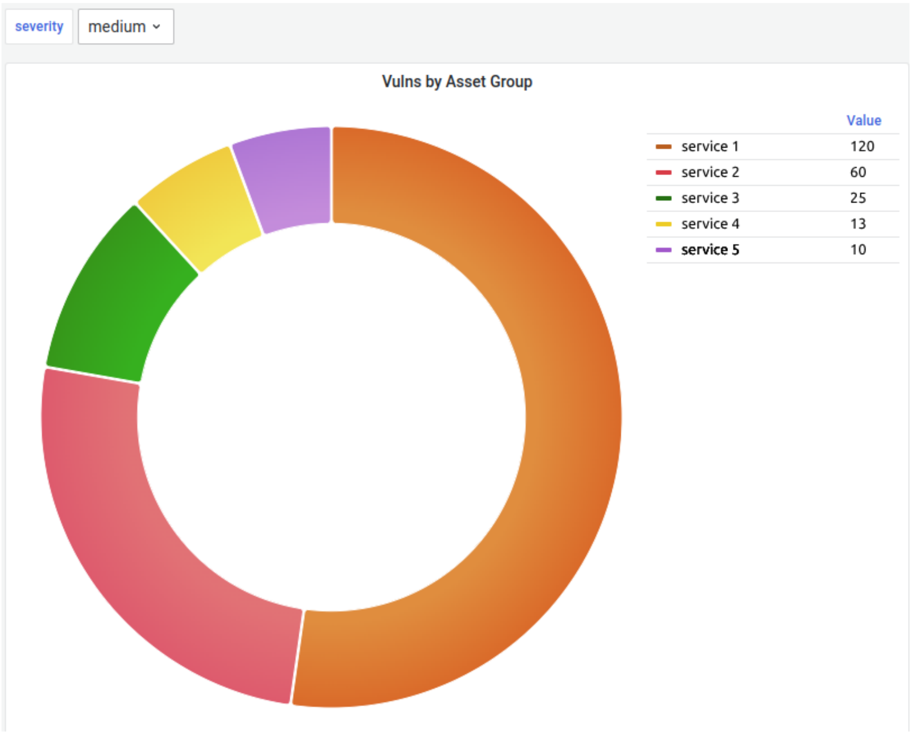
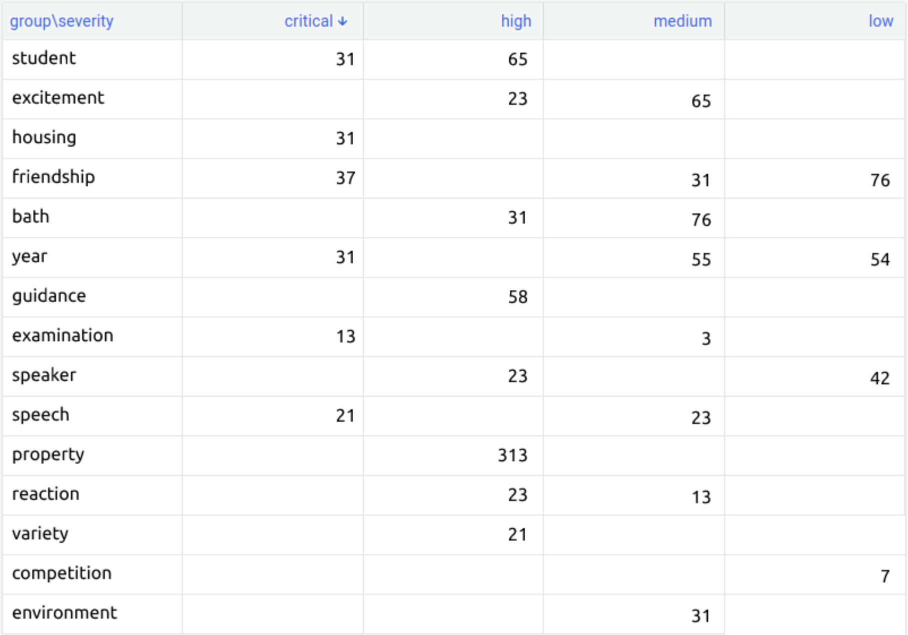
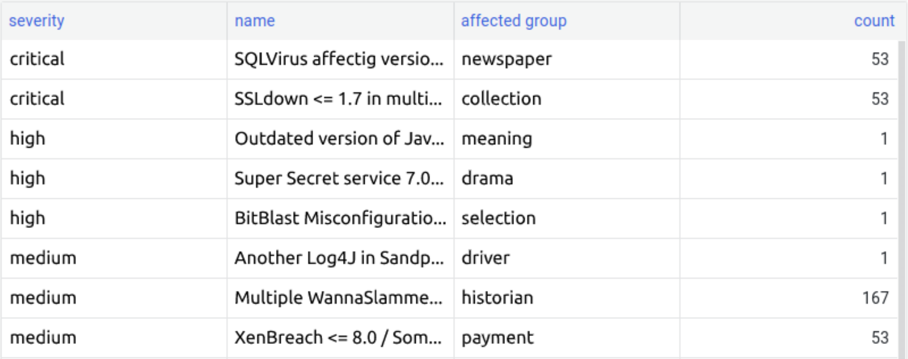


Figure 13. *Pie chart of total number of vulnerabilities by asset group filtered by severity*



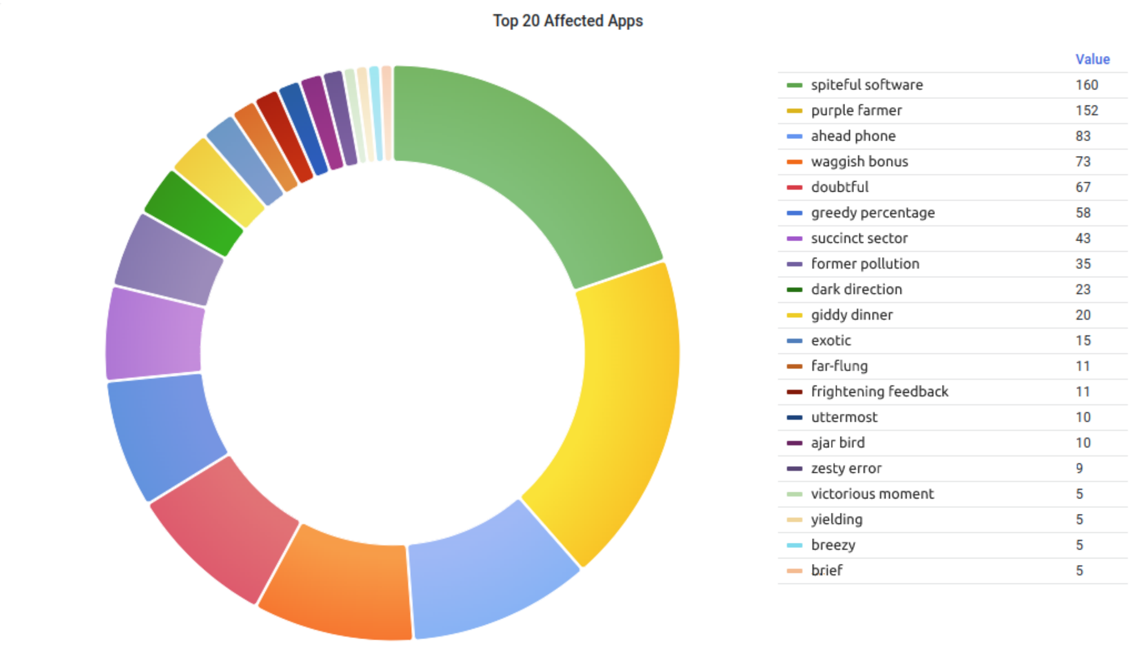
group\severity	critical ↓	high	medium	low
student	31	65		
excitement		23	65	
housing	31			
friendship	37		31	76
bath		31	76	
year	31		55	54
guidance		58		
examination	13		3	
speaker		23		42
speech	21		23	
property		313		
reaction		23	13	
variety		21		
competition				7
environment			31	

Figure 14. *Table of total number of each severity category affecting asset groups*



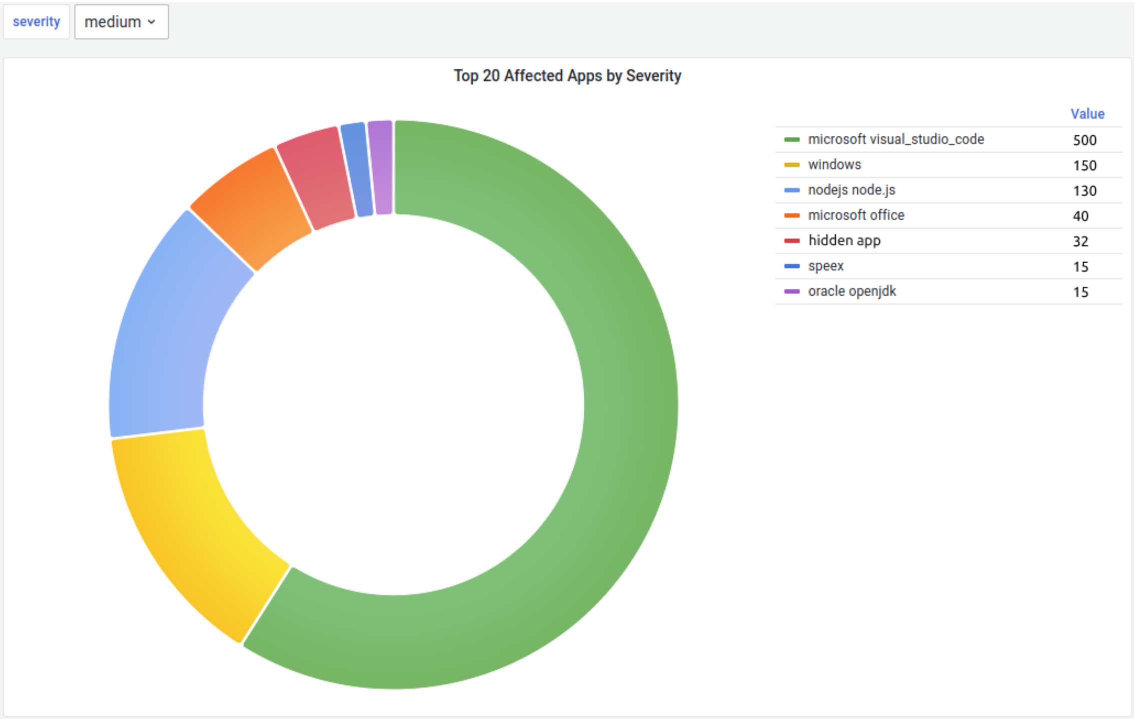
severity	name	affected group	count
critical	SQLVirus affectig versio...	newspaper	53
critical	SSLdown <= 1.7 in multi...	collection	53
high	Outdated version of Jav...	meaning	1
high	Super Secret service 7.0...	drama	1
high	BitBlast Misconfiguratio...	selection	1
medium	Another Log4J in Sandp...	driver	1
medium	Multiple WannaSlamme...	historian	167
medium	XenBreach <= 8.0 / Som...	payment	53

Figure 15. *Table of total number of each vulnerability affecting asset group*



	Value
spiteful software	160
purple farmer	152
ahead phone	83
waggish bonus	73
doubtful	67
greedy percentage	58
succinct sector	43
former pollution	35
dark direction	23
giddy dinner	20
exotic	15
far-flung	11
frightening feedback	11
uttermost	10
ajar bird	10
zesty error	9
victorious moment	5
yielding	5
breezy	5
brief	5

Figure 16. *Pie chart of total number of vulnerabilities by top 20 popular applications*


severity: medium

	Value
microsoft visual_studio_code	500
windows	150
nodejs node.js	130
microsoft office	40
hidden app	32
speex	15
oracle openjdk	15

Figure 17. *Pie chart of total number of vulnerabilities by top 20 popular applications filtered by severity*

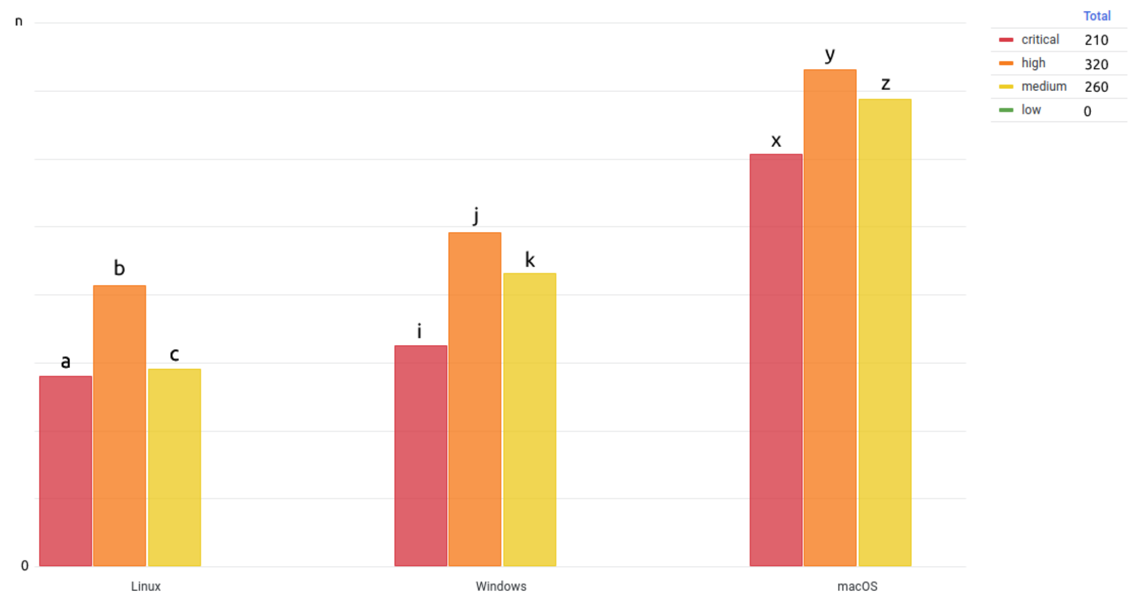


Figure 18. *Stacked bar chart of total number of vulnerability from each category on different operating systems*

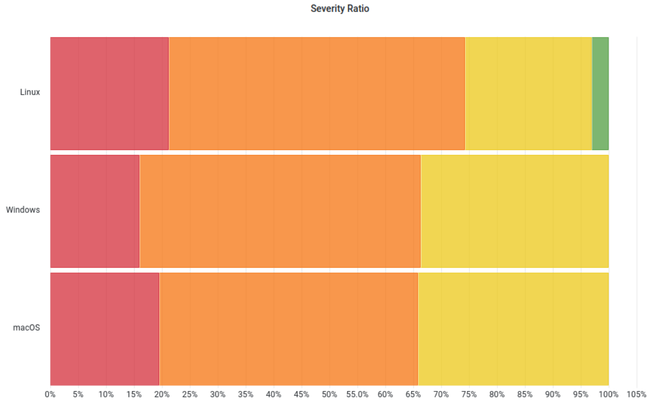
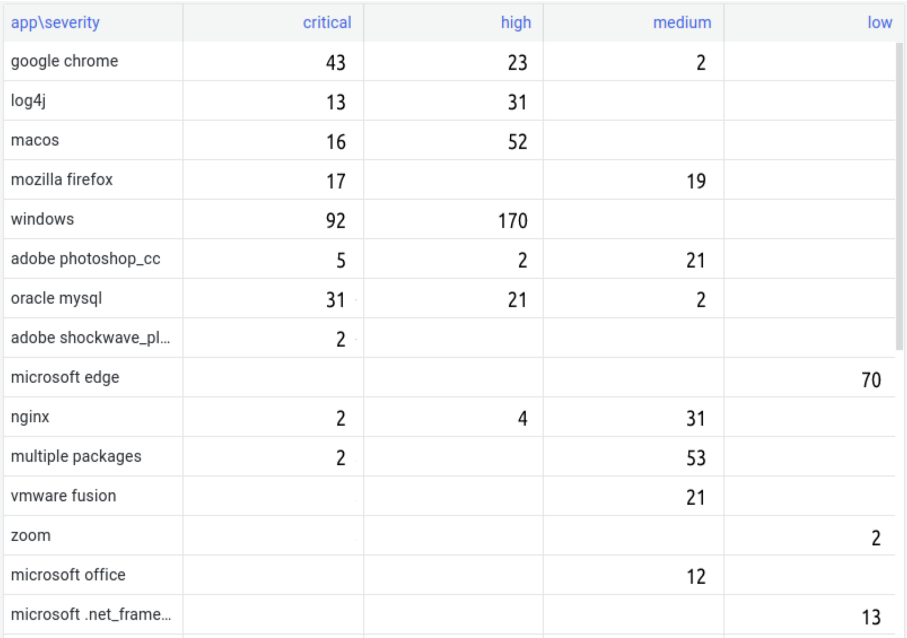


Figure 19. *Stacked bar chart of total number of vulnerability from each category on all operating systems*



app\severity	critical	high	medium	low
google chrome	43	23	2	
log4j	13	31		
macos	16	52		
mozilla firefox	17		19	
windows	92	170		
adobe photoshop_cc	5	2	21	
oracle mysql	31	21	2	
adobe shockwave_pl...	2			
microsoft edge				70
nginx	2	4	31	
multiple packages	2		53	
vmware fusion			21	
zoom				2
microsoft office			12	
microsoft .net_frame...				13

Figure 20. *Table of total number of each severity category affecting all applications*

6 Summary

As a result of this research, a tool referred to as "tenable exporter" was developed. The primary purpose of this tool is to collect, process, and export data related to assets and vulnerabilities. The tool systematically retrieves asset and vulnerability-related data from Tenable.io by making a set of API calls. Retrieved vulnerabilities are categorized according to tags defined in Tenable.io. Tenable exporter generates two separate sets of metrics, one for the servers and the second for workstations. In the case of servers, these metrics are the number and name of vulnerabilities affecting each service group. Meanwhile, for workstations, metrics are numbers of vulnerabilities grouped dimensionally, where one dimension is the operating system while another is the affected application. To store these metrics, tenable exporter uses two distinct databases. The first and the main one is Prometheus, which pulls all numeric time-series data. The second one is MySQL, which stores only the current state of the existing vulnerabilities. Visualizing accumulated information is achieved through Grafana, which obtains this data by querying both data sources.

The developed solution does not intend to at any point replace the visualization provided by Tenable.io. It is more specific and has a more narrow scope, thus offering relatively different functionality and features that are complementary to features of Tenable.io. Specifically, the solution is more focused on the time-series representation of data. It allows users to see not only the increase in the number of total existing vulnerabilities but also the decrease. This allows the information security team to have more visibility of their performance and creates an opportunity to track their effectiveness over time. In addition, all provided metrics are, by their design, either already split or easily splittable into different groups and sub-groups. Leveraging these features enables the information security team to measure their position in particular domains associated with separate departments.

The developed solution brings many benefits to the information security team. Firstly, the power of automation eliminates the need for reoccurring manual calculations and brings attention back to security rather than statistics and analytics. Secondly, a continuous process of visualization with a remarkably lower delay allows the information security team to see the most up-to-date data. This significantly increases their operational awareness, thus allowing more rapid response. Another advantage of this solution is that departments responsible for the remediation of vulnerabilities can directly access and visualize data related to them.

All tools and technologies (except for Tenable.io) used to compose this solution are free and open-source. Practices, ideas, and instruments provided in this solution can be used by any organization or entity that uses Tenable.io and wishes to extend and contextualize their options of visualization. Precisely, this solution can provide guidance on how to separate data associated with different teams, services, operating systems, applications, or more. The only requirement from organizations is to design either hierarchical, dimensional, or hybrid models for grouping vulnerability management data according to their needs. One good example of the target audience is the organizations with IT infrastructure of modern practices, where everything is divided into micro-services and is containerized.

The solution has several limitations and shortcomings, which might be a good point for further improvements and research. The first shortcoming is that solution relies on two separate databases and hence, requires provisioning and maintaining extra services. This is due to the reason that this paper discussed only time-series databases used in Pipedrive. More detailed research can be conducted around the other various time-series databases, which might be better candidates for this assignment. Another topic is that visualizations offered within the current solution do not have any integrations for creating tickets or the tasks for other teams.

Bibliography

[1] P. Foreman, *Vulnerability Management*. Auerbach Publications, 2010.

[2] C. C. Editor, *Vulnerability - glossary*. [Online]. Available: https://csrc.nist.gov/glossary/term/vulnerability (visited on 05/16/2022).

[3] "Costs and consequences of gaps in vulnerability response," Ponemon Institute, Tech. Rep. [Online]. Available: https://media.bitpipe.com/io_15x/io_152272/item_2184126/ponemon-state-of-vulnerability-response-.pdf (visited on 05/16/2022).

[4] M. A. Ben Osbrach, "Report on pipedrive's description of sales management system and on the sustainability of its controls relevant to security, availability, confidentiality and privacy throughout the period october 1, 2020 - september 30, 2021," Marcum LLP, Tech. Rep. [Online]. Available: https://www-cms.pipedriveassets.com/documents/Pipedrive-2021-SOC-3-Final-Report-1.pdf (visited on 04/27/2022).

[5] L. Marcum RAS, *Certificate of registration*, 2019. [Online]. Available: https://www-cms.pipedriveassets.com/documents/ISO-IEC-27001-2013-Certificate-MRAS-Pipedrive-v12.22.2021-1.pdf (visited on 04/27/2022).

[6] R. Schaller, "Moore's law: Past, present and future," *IEEE Spectrum*, vol. 34, no. 6, pp. 52–59, Jun. 1997, ISSN: 0018-9235. DOI: 10.1109/6.591665. [Online]. Available: http://ieeexplore.ieee.org/document/591665/ (visited on 05/16/2022).

[7] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Analyze the future*, vol. 2007, no. 2012, pp. 1–16, 2012. [Online]. Available: https://www.speicherguide.de/download/dokus/IDC-Digital-Universe-Studie-iView-11.12.pdf (visited on 05/16/2022).

[8] *What is data visualization?* [Online]. Available: https://www.ibm.com/cloud/learn/data-visualization (visited on 05/16/2022).

[9] E. Walter and J. Gioglio, *The power of visual storytelling: how to use visuals, videos, and social media to market your brand*, 1 Edition. New York: McGraw-Hill, 2014, ISBN: 9780071823937.

[10] R. E. Horn, *Visual language: global communication for the 21st century*. Bainbridge Island, Wash: MacroVU, Inc, 1998, ISBN: 9781892637093.

[11] Pipedrive, *Pipedrive reaches 100,000 customers: The story in numbers*, Pipedrive. [Online]. Available: `https://www.pipedrive.com/en/blog/the-story-behind-100000-customers` (visited on 05/16/2022).

[12] N. I. Daud, K. A. Abu Bakar, and M. S. Md Hasan, “A case study on web application vulnerability scanning tools,” in *2014 Science and Information Conference*, IEEE, Aug. 2014, pp. 595–600, ISBN: 9780989319331. DOI: `10.1109/SAI.2014.6918247`. [Online]. Available: `https://ieeexplore.ieee.org/document/6918247/` (visited on 05/16/2022).

[13] *Tenable.io API: Get started*, Tenable Holdings, Inc, 2020. [Online]. Available: `https://developer.tenable.com/docs/get-started` (visited on 05/16/2022).

[14] *Tenable.io API: Rate limiting*, Tenable Holdings, Inc, 2020. [Online]. Available: `https://developer.tenable.com/docs/rate-limiting` (visited on 05/16/2022).

[15] *Common platform enumeration*. [Online]. Available: `https://cpe.mitre.org/` (visited on 05/16/2022).

[16] I. T. L. Computer Security Division, *Common platform enumeration (cpe) - security content automation protocol: Csrc*. [Online]. Available: `https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/cpe` (visited on 05/16/2022).

[17] D. Namiot, “Time series databases,” in *DAMDID/RCDL*, 2015, pp. 132, 133.

[18] J. Han, H. E, G. Le, and J. Du, “Survey on nosql database,” in *2011 6th International Conference on Pervasive Computing and Applications*, Oct. 2011, pp. 363–366. DOI: `10.1109/ICPCA.2011.6106531`.

[19] C.-O. Truică and E. S. Apostol, “Nosql environments and big data analytics for time series,” in Jun. 2021, pp. 108–138, ISBN: 9780367814397. DOI: `10.1201/9780367814397-6`.

[20] *Dbms popularity broken down by database model*. [Online]. Available: `https://db-engines.com/en/ranking_categories` (visited on 05/16/2022).

[21] P. Authors, *Overview*, The Linux Foundation. [Online]. Available: `https://prometheus.io/docs/introduction/overview/` (visited on 05/16/2022).

[22] InfluxData, *Influxdb*, InfluxData Inc. [Online]. Available: https://docs.influxdata.com/influxdb/v2.2/reference/glossary/#influxdb (visited on 05/16/2022).

[23] N. Iliinsky and J. Steele, *Designing data visualizations: intentional communication from data to display*, eng. Beijing Köln: O'Reilly, 2011, ISBN: 9781449312282.

[24] S. R. Midway, "Principles of effective data visualization," *Patterns*, vol. 1, no. 9, p. 100 141, 2020, ISSN: 2666-3899. DOI: https://doi.org/10.1016/j.patter.2020.100141. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666389920301896 (visited on 05/16/2022).

[25] G. Labs, *Time series*, Grafana Labs. [Online]. Available: https://grafana.com/docs/grafana/latest/visualizations/time-series/ (visited on 04/27/2022).

[26] ——, *Bar chart*, Grafana Labs. [Online]. Available: https://grafana.com/docs/grafana/latest/visualizations/bar-chart/ (visited on 04/27/2022).

[27] ——, *Pie chart*, Grafana Labs. [Online]. Available: https://grafana.com/docs/grafana/latest/visualizations/pie-chart-panel/ (visited on 04/27/2022).

[28] Tenable.io, *Retrieve asset data from tenable.io*, Tenable Inc. [Online]. Available: https://developer.tenable.com/docs/retrieve-asset-data-from-tenableio (visited on 04/27/2022).

[29] ——, *Cvss vs vpr*, Tenable Inc. [Online]. Available: https://docs.tenable.com/tenablesc/Content/RiskMetrics.htm (visited on 04/27/2022).

[30] P. Authors, *Metric types*, The Linux Foundation. [Online]. Available: https://prometheus.io/docs/concepts/metric_types/ (visited on 05/16/2022).

Appendix 1 – Non-exclusive license for reproduction and publication of a graduation thesis[1]

I, Archil Kristinashvili

1. Grant Tallinn University of Technology free license (non-exclusive license) for my thesis "Creating Custom Solution for Visualizing Vulnerability Management Data at Pipedrive", supervised by Kristian Kivimägi
   - 1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   - 1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive license.
3. I confirm that granting the non-exclusive license does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

16.05.2022

[1] The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 - Disclaimer

Some parts of this thesis are done in collaboration with Giorgi Zeikidze. As both works solve the same general problem, the introduction, background, and problem statement chapters contain some identical text.