New Zealand Wide Internet Scanning Data Analysis for Interesting Trends

Shruti Raja

***Abstract*—In our increasingly digital society, understanding cyber behaviour and trends in New Zealand is vital for individuals and organisations to enhance their cybersecurity practices and protect themselves effectively. However, there is currently a limited amount of publicly available information regarding cyber behaviour and trends specific to New Zealand. This project aims to address this issue by developing an effective tool that will visualise the data to facilitate identifying interesting trends found in the data collected by ZX Security. Given the size of the dataset, we conducted research to identify the most effective data handling and processing methods, and ultimately chose indexing and projection for efficiency. The tool extracts relevant information from the data and generates results in the form of graphs, which will be analysed and compiled into a publicly accessible report. By providing insights into New Zealand's cybersecurity landscape, this project seeks to contribute to a safer digital society, promote better cyber practices, and bridge data gaps, ultimately fostering a more secure online environment.**

I. Introduction

IN today's interconnected world, cybersecurity has emerged as a critical concern for organisations and individuals. The widespread use of devices and the increasing presence of individuals online have heightened the importance of safeguarding digital systems and personal information. In New Zealand, there is a lack of easily accessible data and statistics on cybersecurity behaviour and trends. This shortage of information is problematic because, without clear data on cybersecurity behaviour and trends, individuals may remain unaware of the importance of their online habits, potentially engaging in practices that inadvertently expose them to cyber threats. This lack of awareness can lead to an increase in breaches that could otherwise be avoided. When people know more about cybersecurity specifically in the New Zealand context, they're more likely to pay attention, share what they've learned, and take steps to protect themselves and others. By addressing the problem of limited accessibility to cybersecurity data and statistics, this project aims to raise awareness and provide insights into cyber behaviour, promoting safer practices in New Zealand. The importance of this project stems from the potential implications of cybersecurity breaches and incidents, which can disrupt businesses, compromise personal information, and undermine trust in digital systems. The project will address this issue by using a dataset collected by ZX Security. This dataset, in JavaScript Object Notation (JSON) format, contains details from various New Zealand devices such as HTTP content, open ports, server software, and internet service providers (ISPs). Access to this data is vital for understanding the cybersecurity landscape and developing effective risk mitigation strategies. The overall goal of this project is to utilise the available dataset to gain insights that can improve cyber behaviour and ultimately help contribute to a safer digital society. Currently, while there are numerous data analysis tools available, they fall short in handling the extensive dataset provided by ZX Security, as its size exceeds their processing capabilities. Additionally, accessing all the features of these existing tools is often locked behind paywalls. To overcome these limitations, I will be developing a tool specifically for extracting and visualising information from the ZX dataset. This custom tool will have the capacity to manage the data's size and will also avoid the financial barriers associated with existing approaches. The visualisations generated by this tool will assist in identifying cybersecurity trends within the New Zealand digital landscape. These insights will then be consolidated and presented in an in-depth report that will be made available to the public. By doing so, the project will not only raise awareness but also educate the wider community about common cybersecurity habits and why it's important to practice strong online safety.

To achieve the goals mentioned, the project aims to produce two deliverables: 1. A tool that efficiently extracts information from the dataset and visualises the results. 2. A report that uses the results of the tool to visualise trends found in the data and areas for improving cyber behaviour.

By achieving this goal, the project aligns with Goal 4 of the United Nations' 17 Sustainable Development Goals, which aims to ensure inclusive and equitable quality education and promote lifelong learning opportunities for all [1]. The project's deliverables will make valuable educational resources accessible to the public. This accessibility promotes lifelong learning by providing individuals with the opportunity to enhance their knowledge and understanding of cybersecurity, fostering a culture of digital security and responsible online practices among all users.

One of the main challenges involved in the process of making this project successful is effectively handling the substantial amount of data. The dataset is extensive, requiring the tool to efficiently search and extract the correct information requested by the user. By implementing efficient data searching and extraction methods, the tool will overcome this challenge and enable meaningful analysis of the dataset. Specifically, to evaluate the efficiency of our proposed method, we considered performance metrics, with one metric being the system's response time to user requests. In particular, we focused on the time it takes for the system to generate a graph based on user input, such as when they click the button to plot the graph. Through the incorporation of query optimisation techniques, we were able to enhance efficiency.

Our optimisations have successfully reduced the time, allowing us to retrieve a years worth of data in just 4-5 minutes. This achievement demonstrates and effectiveness of our approach in efficiently handling and processing the extensive dataset from ZX Security.

To ensure the accuracy and correctness of the graphs generated by the tool, an essential aspect of our project involved functionality testing. Functionality testing is a performance metric we used to verify whether the tool performs its intended functions correctly. In our case, it was important to confirm that the graphs accurately represented the data they were intended to visualise. Functionality testing was conducted by comparing the results of the graphs generated with the raw data from the database. The process involved querying the MongoDB database to extract the same data that the tool used to generate graphs. By ensuring that both sets of results aligned, we could confidently confirm that the tool effectively achieves its objective of providing accurate and meaningful visualisations.

Throughout the project, considerations for environmental sustainability and resource optimisation will be taken into account to minimise resource consumption and enhance computational efficiency. Additionally, we recognise the sensitivity of the dataset, which contains information related to real individuals in New Zealand. Therefore, measures have been implemented to safeguard their privacy and protect other confidential details.

The remainder of this report is organised as follows. Section 2 covers the literature review, and discusses existing tools with similar functionality. Section 3 looks at the different tools and methods tried and used throughout the project. Section 4 presents the conceptual design and implementation of the proposed method and also discusses sustainability aspects. Section 5 covers the different performance metrics used to evaluate the effectiveness and efficiency of the proposed method. Section 6 presents the results and parts of the report produced for ZX Security. Section 7 discusses the limitations of the proposed method. Lastly, sections 8 and 9 will conclude the report and suggest possible next steps for continuing this project in the future.

II. Literature Review

To ensure that the tool developed as a result of this project is unique and provides advantages over existing tools, research has been conducted to analyse the features and functionality offered by current tools. There are numerous other tools available in the market that offer similar functionality to the tool being developed in this project, with slight differences and purposes. While it is impractical to provide a review of every tool, this literature review will focus on key popular tools that are widely recognised in the field of data visualisation and analysis. Each of these tools brings its own strengths and limitations. Tableau, Excel, and Elastic Stack are just a few examples of the tools analysed.

A. Tableau

Tableau is a widely recognised data visualisation tool known for its powerful data exploration and interactive visualisations [2]. It offers a user-friendly drag-and-drop interface, allowing users to create visually appealing charts, graphs, and dashboards. Tableau supports a wide range of data sources and provides advanced analytics capabilities. However, the main limitation of Tableau is its pricing model, as it requires a monthly subscription cost for full access to its features, which can pose a barrier for some users.

B. Excel

Excel is another popular tool. Though its primary purpose is to create and manage spreadsheets, it also allows users to input JSON data. This data is then organised neatly into columns and rows. [3]. Additionally, Excel provides the capability to create graphs directly from this organised data. Excel uses its built-in engine to create graphs and charts from data. This engine is an integral part of Microsoft Excel itself and does not rely on external software. Excel is designed to handle a wide range of chart types and customisation options, allowing users to visualise data directly within the Excel application. Excel, while a versatile tool, comes with its set of limitations. Firstly, the graphs and charts it produces are static and lack interactivity, which can be a drawback because they limit the depth of understanding that can be gained from them. On the other hand, dynamic graphs offer a more engaging experience, allowing users easier access to details like exact values at specific points in time. These details will be beneficial when compiling the report that analyses the trends found. Being able to interact with the data and see precise values at particular moments will help to trace patterns back to their origins, identify correlations, or spot anomalies. This will help improve the analysis and ensure the conclusions in the report are accurate. Another constraint of Excel is that it has maximum row limit of 1,048,576 rows per worksheet. While this capacity suffices for most purposes, users dealing with large datasets will encounter performance issues and might need to explore alternatives. Additionally, Excel's ability to import and process JSON data relies on the data being well-structured and clean; if the JSON data contains intricate nested structures, it may not be parsed correctly. Furthermore, Excel is not a free software, and licensing fees are required for its use. These limitations underscore the importance of considering specific data requirements and constraints when choosing Excel as a data analysis tool, as it may not always be the most suitable choice for tasks involving interactivity, very large datasets, or complex data structures.

C. Elastic Stack

Elastic Stack is a powerful suite of open-source tools designed for searching, analysing, and visualising data in real-time [4]. While the primary use case of Elastic Stack revolves around log and data analytics, its capabilities extend beyond this core function. It allows users to ingest and index data from various sources, making it accessible for search and analysis. Elastic Stack provides fast search and querying capabilities. Kibana, another component of Elastic Stack, offers a user-friendly interface for creating interactive visualisations from the indexed data. Elasticsearch differs

from relational databases by using a schema-less approach. Instead of SQL, it employs a Query DSL (Domain Specific Language) based on JSON syntax for querying. The Query DSL allows users to create versatile queries for full-text search, filtering, and aggregations [5]. It is important to note that Elasticsearch does have a maximum document size limitation of 100MB. Elastic Stack's scalability and versatility make it a valuable tool for security analytics, and business intelligence however, it can also be complex to set up and configure, which adds extra time to get past the learning curve. Effective deployment often requires a good understanding of the various components and their interactions. Elasticsearch also has limitations when it comes to handling certain data types, such as nested arrays or deeply nested JSON structures. Users may need to flatten or preprocess data to fit the schema.

While these tools share common features and functions, they have their respective strengths and limitations as discussed above. One key advantage of the tool developed as part of this project is that it will be made open source, ensuring costless accessibility to analyse the data. Additionally, by focusing specifically on analysing New Zealand-wide internet scanning data, the tool will help in producing a report that will address a gap in the existing landscape, as no publicly available reports analysing such data currently exist. Additionally, the report will be made publicly available on the ZX Security website, promoting the open sharing of insights derived from the New Zealand context.

III. Tools and Methodolgy

To ensure the successful development of the proposed solution, critical factors were evaluated to address the challenges related to the process of accessing and displaying the data. Key considerations involved identifying suitable tools and methods for data access and storage, selecting an appropriate programming language for tool development, researching efficient strategies for data processing, and determining effective display methods for the graphical user interface (GUI) of the tool, graphs, and report

A. Data Access and Storage

The data collected by ZX Security consists of approximately 370 million JSON documents in a file that is 142 GB. To access this data, we had the option of either storing the data locally via a USB drive containing the file or accessing it remotely through secure shell (SSH). After consideration, SSH was chosen as the preferred option due to its benefits compared to using a USB drive. SSH provides secure remote access to the server hosting the data. This ensures data integrity and confidentiality during the transfer and retrieval processes. Using a USB drive introduces potential risks such as data loss, theft, or unauthorised access if the drive is misplaced. With SSH, I can remotely interact with the server's command-line interface, allowing for easy data retrieval and subsequent processing. This eliminates the risk of physical damage or possible loss of the USB device. For data storage, the decision was made to use MongoDB by installing it on the ZX Security server. This choice proved to be advantageous for several reasons. Firstly, MongoDB is well-suited for handling JSON data, this simplifies the data storage process, allowing for transformation and compatibility with JSON-based data [6]. Another benefit of using MongoDB is its ease of access and flexibility in data retrieval. MongoDB provides a wide range of query capabilities, allowing for efficient searching, filtering, and extraction of information from the stored data. Its query language and indexing mechanisms enable quick data retrieval based on various criteria, enhancing the overall data analysis process. Additionally, MongoDB offers a scalable and distributed architecture, making it suitable for handling large volumes of data. Since this project involves an extensive dataset, the scalability of MongoDB ensures that the system can accommodate the growing dataset without sacrificing performance. While MongoDB was chosen as the preferred option for data storage, there are alternative solutions available. One option is traditional relational databases such as MySQL [7] or PostgreSQL [8]. These databases offer structured data storage and have robust querying capabilities. However, they require additional effort and schema definition to accommodate Binary Javascript Object Notation data (BSON). Additionally, there are other NoSQL that are designed for handling large datasets and distributed systems. These databases excel in scalability and fault tolerance, making them suitable for large-scale data storage and processing. While these databases offer their own unique features and strengths, MongoDB was chosen as the preferred option for this project. One key factor in selecting MongoDB was its extensive documentation and community support. Having prior experience and familiarity with MongoDB allowed for a smoother and more efficient development process, as it eliminated the need to invest time in learning and adapting to a different database technology.

B. Programming language

When selecting the programming language for developing the proposed method, Python was chosen as the preferred option. Python provides a range of useful libraries and frameworks that can facilitate the development process, particularly in the domain of data analysis and visualisation [9]. One notable library is Pygal, which offers a set of tools for creating various types of graphs and visualisations [10]. Python has support for interacting with different database systems, including MongoDB, through libraries such as PyMongo. This compatibility allows for easy integration between the tool and the MongoDB database used for storing the collected data.

While there are alternative programming languages available for this project, Python seemed more advantageous due to these specific benefits. Other languages, such as Java, also offer data analysis capabilities and visualisation libraries, but Python stood out for its simplicity, readability, and extensive community support. Additionally, my familiarity and proficiency with Python further contributed to the decision, as it allowed for faster development and reduced the learning curve associated with a new language.

C. Agile Methodology

For this project, we will be adopting an Agile methodology to ensure flexibility, adaptability, and continuous improvement. Agile methodology emphasises iterative development, collaboration, and frequent feedback from stakeholders. By incorporating weekly iterations, we will be able to gather regular feedback from clients and stakeholders, allowing for the incorporation of improvements and addressing any unforeseen changes or requirements. Unlike the traditional Waterfall method, where changes are difficult to accommodate once a phase is completed, Agile methodology embraces changes as an inherent part of the development process. This iterative approach allows for more responsive and dynamic project management, enabling us to deliver a solution that can handle changes and ensure a higher level of client satisfaction.

D. Methods of handling large datasets

One of the main challenges encountered during this project is effectively handling the large volume of data. The dataset provided is very large in size, presenting difficulties in efficiently retrieving and processing the data. Working with large files requires special considerations to ensure optimal performance and resource utilisation. It has been observed that traditional approaches are not suitable for handling such data due to limitations in memory capacity and processing capabilities. As a result, different techniques and methods have been explored to tackle this challenge effectively. The following sections cover the several different approaches that were explored and evaluated to identify the most effective method for obtaining this information.

1) Hashing: One of the methods considered to handle the large amounts of data involved using hashing. This approach includes reading the BSON data from a file, processing it using a hash function, and then performing further analysis on the hashed representation. The hashing method proved to be fast and efficient, requiring only a small amount of memory to store the hash table and individual documents. However, hashing does not provide the original data but rather a hashed representation of it. To make the output human-readable, a lookup table would need to be created to map each hash to its corresponding original data. Given the massive size of the BSON file, managing and creating such a lookup table could pose challenges and potentially become too difficult to handle. Considering the complexities associated with hashing and managing a lookup table, it was determined that this method was not the most suitable for the project's objectives and timeline.

2) Grep: Next, we explored the Grep command-line tool, which is commonly used to search for specific text patterns in files. However, while experimenting with Grep, we quickly observed that Grep operates by reading and processing one line at a time, resulting in slower performance and increased memory usage. The inefficiency of this method can hinder the progress of the project because it is not suitable for extracting meaningful insights from the extensive dataset. Due to these limitations, grep was not deemed suitable for the project as it was not efficient, and an alternative approach will be needed to ensure efficient data processing and analysis.

3) Sharding: Another potential method for handling large datasets in MongoDB is sharding. Sharding is a horizontal scaling technique that distributes data across multiple servers or shards [11]. Each shard is responsible for managing a subset of the data, allowing the database to handle larger volumes of information and higher throughput compared to traditional, non-sharded databases that might struggle with performance and scalability issues under heavy loads. Sharding works by partitioning data based on a shard key, which determines how data is distributed across shards. While sharding can largely enhance database performance and capacity, it was not a feasible option for this project. The project's infrastructure provided by ZX Security consisted of a single server, which lacked the necessary multiple servers required for sharding. Therefore, sharding as a solution was not viable in this particular context.

4) Lazy loading: Another method explored for handling large datasets was lazy loading. Lazy loading is an approach that involves retrieving and processing data in small, more manageable chunks as needed, rather than loading the entire dataset into memory at once. This technique leverages the cursor object in MongoDB, which allows for efficient iteration through large result sets without loading everything into memory simultaneously. With lazy loading, data is fetched and processed incrementally, reducing memory and enhancing performance. To test its effectiveness, we used this method to identify the most common HTTP proxy versions in the data as an example. The lazy loading approach was able to retrieve and process the results in seven minutes, which was an improvement in response time compared to the other methods. While this performance was okay, further optimisations could potentially enhance it even further, demonstrating the effectiveness of the lazy loading technique for managing large datasets.

5) Indexing and projection: To enhance the performance of the proposed method, we combined multiple techniques to handle the substantial dataset efficiently. Alongside implementing lazy loading, MongoDB's indexing and projection features were also integrated into the Python code. Indexing involves the creation of optimised data structures within the database engine, efficiently creating a roadmap for swift data retrieval, and largely speeding up query processes. Projection, on the other hand, is a query optimisation technique that allows specifying the exact data fields you wish to retrieve, reducing data transfer and resource consumption. By combining indexing, projection, and lazy loading, we streamlined data retrieval and minimised resource usage, resulting in improvements in query execution speed. Testing this on the same previous example of fetching the most common HTTP proxies, we were able to retrieve results in two minutes which is a 71% improvement in performance compared to lazy loading. This optimisation was essential in enhancing the functionality of

the proposed method.

E. Methods of Displaying Graphs

One of the requirements of this project was to create a report that visualises trends found in the data. To achieve this, we had several options for selecting a tool to create informative and visually appealing graphs. Among the tools considered were Matplotlib [12], Vega [13], and Pygal [10]. Matplotlib, a widely used Python library, is known for its graphing capabilities [12]. While Matplotlib produced satisfactory graphs, they lacked interactivity. Since the primary purpose of developing this tool was to analyse the graphed data, interactivity became crucial as it can be challenging to make accurate claims when dealing with large datasets without the ability to inspect specific data points on a graph. Consequently, we explored Vega, a powerful visualisation tool. However, we found it challenging and time-consuming to set up [13]. Pygal is a user-friendly Python library for creating interactive and visually appealing graphs, we found that it met the requirements for generating graphs [10]. Pygal's ease of use, combined with its visually appealing and interactive graph features, made it the ideal choice for closely examining the data and highlighting the interesting insights in the report.

F. GUI Libraries

The tool required a GUI with its purpose being to query the database by letting the user specify the variables to be graphed. The selected variables will later be passed to Pygal to graph. We aimed for a straightforward GUI design as it is a component part of the pipeline to generate the graphs. To create this GUI, the libraries considered were Tkinter [14], PyQt [15], and WxPython [16].

1) Tkinter: Tkinter is known for its simplicity and user-friendly interface [14]. It excels at creating straightforward GUIs, which makes it a good choice for the requirements of this study. Tkinter offers a variety of widgets like sliders, list boxes, check boxes, and radio buttons, which are well-suited for variable selection tasks. With basic styling options, Tkinter allows creating a plain and simple GUI that aligns with the project's needs. Additionally, Tkinter easily integrates with Python, allowing for efficient workflow.

2) PyQt: PyQt, on the other hand, provides a rich set of widgets and controls, which can be advantageous for creating sophisticated GUIs [15]. However, for this project's, simple variable selection and data visualisation needs, these features will be too much. PyQt does offer extensive customisation options, enabling users to create highly tailored and visually appealing interfaces, but this would introduce complexity that is unnecessary for this project. One notable complexity when working with PyQt is its signal and slot mechanism for event handling. While this approach offers a powerful way to handle user-interface events and customize behavior, for a simple project that only requires basic variable selection and button-click functionalities, setting up and managing these connections can be overkill.

3) WxPython: WxPython is another option we considered. It offers a diverse range of widgets and native controls, including those suitable for variable selection tasks [16]. Much like Tkinter, it provides a native look to different platforms, which can make the GUI feel familiar to users on Windows, Mac OS, and Linux. WxPython has an active community and extensive documentation, which can be a valuable resource for assistance or additional information.

In conclusion, for a simple and plain GUI primarily focused on variable selection and data visualisation, Tkinter was a suitable choice. Its simplicity, ease of use, and straightforward approach aligned well with this project's requirements. Tkinter ensures that we can quickly and easily create a user interface that will allow us to select variables to query and graph without introducing unnecessary complexity. While PyQt and WxPython are powerful for more complex GUI applications, they weren't the most efficient choices for this project. Fig. 1 shows the end result of the GUI made with Tkinter and its visual components.

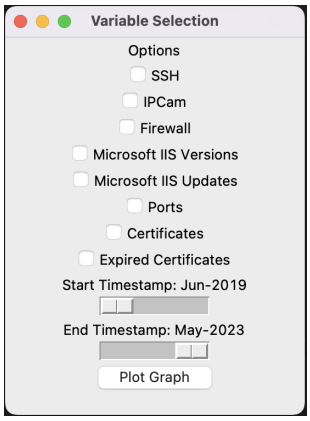

Fig. 1. The interface of the proposed method.

G. Report Display

In terms of the creation of the report, including its formatting and visual aspects, the initial plan was to write and design it using Google Docs. However, as we began working on it, it was quickly realised that the resulting document appeared plain, and lacked elements that could capture the reader's attention effectively. Needing a more visually appealing approach, we considered Canva. Canva is an online design tool that offers a wide selection of eye-catching report templates, enabling the selection of a layout that best suits the project's needs. Canva's user-friendly interface and customisation options made it easy to design the report, from formatting to selecting fonts and creating attention-grabbing visual elements [17]. Ultimately, Canva's design capabilities helped in transforming the report into a visually engaging and professional document that effectively conveyed the project's findings.

IV. Design and Implementation

A. Conceptual Design

For the system architecture, the Model-View-Controller (MVC) design is used to create a well-structured and efficient framework, where each component plays a different role in the system's functionality. The Model represents the database which acts as the central storage for the data that the system retrieves and uses from the database. This includes managing the relevant information and making it readily accessible for manipulation. The Controller component is responsible for the GUI. Users interact with the system through this interface, allowing them to select which variables they want to visualise in graphs. The Controller takes user inputs and constructs queries based on the user's selections and passes them to the Model (database) for data retrieval. The View component corresponds to the graphical representation of data. Once the Controller has retrieved the data from the Model, it is passed to the View, which then transforms the data into graphical representations for the user to interpret. The interfaces and interactions between these components are key to the system's functionality. The Controller communicates with both the Model and the View, serving as the intermediary. The controller sends requests and data to the Model for database operations and receives results that it subsequently passes to the View for visualisation.

The choice of the Model-View-Controller architectural pattern for this project is appropriate for several reasons. MVC offers a clear and organised structure that aligns well with the specific objectives of the tool developed. The following section highlights reasons why MVC was the right choice for this project and an alternative structure.

1) Separation of concerns: MVC enforces a clear separation of concerns, dividing the application into three distinct components: Model, View, and Controller. This separation allows for modular development and maintenance. In the context of this project, where data retrieval, user interface, and data visualisation are distinct functionalities, MVC's separation of concerns is highly beneficial. It ensures that changes or updates to one component can be made without affecting the others, leading to more straightforward code maintenance.

2) User-centric Interface: The Controller component in MVC is responsible for handling user interactions and providing a user-friendly interface. Given the primary function of the proposed method is to allow users to query and visualise data, the Controller's role in managing the GUI is crucial. It enables users to interact with the system intuitively, select variables of interest, and trigger data retrieval and visualisation processes. MVC's emphasis on user-centric design aligns perfectly with this tool's purpose.

3) Flexibility and Maintainability: MVC architecture offers flexibility in terms of adapting to evolving requirements. As data visualisation needs or user interface enhancements arise, the MVC structure can accommodate these changes without major disruptions to the existing codebase. This adaptability is valuable in ensuring the tool remains relevant over time.

4) Efficient Data Flow: In MVC, data flows seamlessly between the Model, View, and Controller components. This efficient data flow is critical for a tool that retrieves data from a MongoDB database, processes it, and presents it as graphical representations. The Controller collects the GUI state, constructs queries, and interacts with the Model to retrieve data, which is then efficiently passed to the View for rendering. This streamlined data flow minimises bottlenecks and ensures a smoother user experience.

5) Alternative architecture: While the architecture was a suitable choice for this project, an alternative structure I considered was Model-View-ViewModel (MVVM). MVVM is an architectural pattern that is similar to MVC but is often associated with applications that have rich and dynamic user interfaces. In MVVM, the ViewModel is responsible for preparing the data that is presented in the View. While MVVM can be beneficial for complex user interfaces with extensive interactivity, it might introduce redundant complexity for a tool with a primary focus on data visualisation. Since the GUI requirements for this project were relatively simple, the MVC pattern provided a more straightforward and manageable solution. Fig. 2 illustrates the system architecture and the flow of each component functioning.

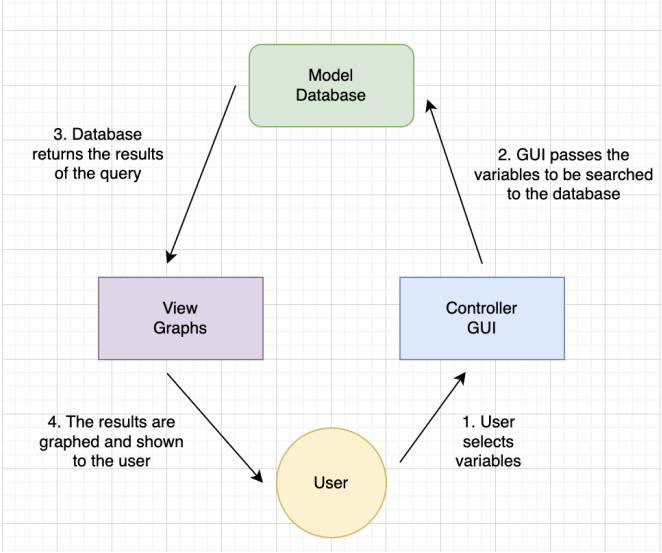

Fig. 2. MVC System Architecture of the proposed method.

B. Implementation

The implementation revolves around creating an interface for querying and visualising data from a MongoDB database. The system integrates three key components: the GUI, the database, and the graph visualisation functions.

1) Graphical User Interface (GUI) with Tkinter: The Tkinter library is used to create a simple GUI that provides users with options to select the variables they would like to see graphed. Users can select these variables by clicking the checkboxes which will be dynamically updated by the GUI and stored in memory. To select the date range of the data users want to see, two sliders are provided, as an interactive way to define the start and end period of data retrieval. The graphing process is triggered when the "Plot Graph" button is pressed, this will fetch data from the database and generate a graphical representation.

2) Database Interaction: The system is designed to interact with the MongoDB database that stores the required data. The Database class is responsible for establishing and maintaining this connection. Key functions of this module include connection setup, query building, and data processing. First, the connect method initialises the connection to the MongoDB database and the specific collection. It takes user-defined host, database, and collection parameters, providing flexibility in working with different database setups. Next, the query method constructs database queries based on user selections, translating their choices from the Tkinter GUI into precise queries for data retrieval. Once data is retrieved, the query method processes it so that the results can be stored. The data retrieval results in numerous JSON documents, each containing the variables selected by the user. In this processing step, the documents are gathered by aggregating and summarising the data. This includes tasks such as counting occurrences, calculating percentages, and transforming the raw data into a structured format. The processed data is then organised and stored in a dictionary, simplifying the subsequent graphing process.

3) Graph Visualisation Functions: Graphing is the final component of the system and it presents the data in a visual and understandable format. The Pygal library is used to create informative line graphs, pie charts, stacked bar graphs, and time series graphs. Each type of graph has its own function that will be used when a variable that uses that graph type is chosen. The generated graphs are saved as Scalable Vector Graphics (SVG) files so they can be easily viewed and embedded in reports.

In conclusion, the implementation of the system was guided by several key considerations, including the choice of Tkinter for GUI development due to its simplicity and Python compatibility, the use of MongoDB for efficient data storage and retrieval, and the selection of Pygal for creating informative and visually appealing graphs. These implementation choices were made to ensure a user-centric, efficient, and effective tool for querying and from a MongoDB database. The system's implementation ensures an efficient flow from user interactions through the Tkinter-based GUI to data retrieval from the database and finally to graph visualisation. Fig. 3 shows a flow chart that outlines the key methods used in each MVC component in order of their functionality.

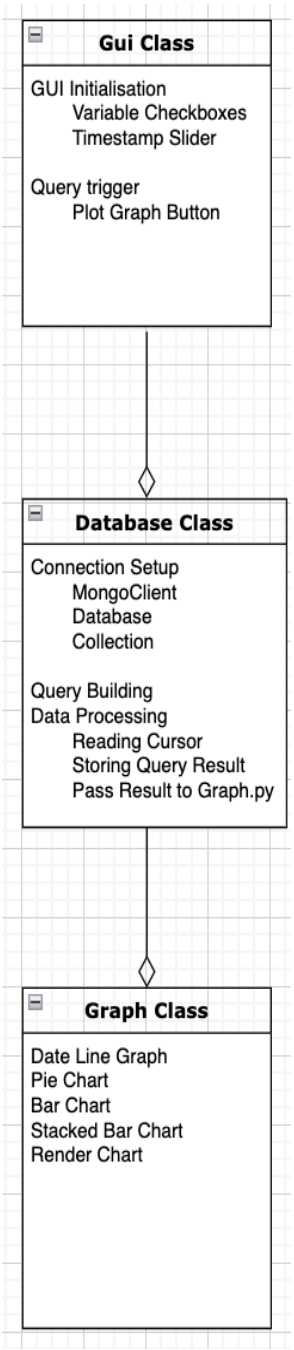

Fig. 3. Implementation flowchart of the proposed method.

C. Sustainability Consideration

In terms of sustainability considerations, this project upholds social sustainability by specifically safeguarding users' right to privacy. Recognising the sensitive nature of user data, our proposed method has been designed with data protection measures to uphold privacy rights of people whose data is being represented in this study. While the intention behind the proposed method is to strengthen cybersecurity practices in New Zealand, it's important to acknowledge the potential risks associated with making such information publicly accessible. Providing too much information can inadvertently aid malicious actors in understanding the digital landscape of

New Zealand, including potential vulnerabilities and common trends. Attackers often seek out detailed data about systems, prevalent security protocols, or common weaknesses within a specific demographic or geographical location. If the proposed method were to provide insights that are too detailed without proper access control, it could potentially serve as a roadmap for bad actors. Measures have been implemented for this project to address the inherent risks of sensitive data exposure. Firstly, all data presented in the report has been anonymised; specific details such as actual IP addresses, host names, and other potentially sensitive information are not disclosed. This approach ensures that while the information remains valuable for understanding cybersecurity patterns, it does not provide exploitable details to potential malicious actors. Furthermore, although the code for the tool will be made open source to promote transparency and collaboration, it's designed to operate as a standalone application without any embedded data. The code is structured to connect to a database, but access to the actual database is strictly controlled and limited to specific authorised team members. This approach to data access further safeguards sensitive information while still supporting the proposed methods objective to enhance cybersecurity awareness and practices.

V. Evaluation

A. Efficiency

One of the performance metrics used to evaluate the proposed method was efficiency. Efficiency refers to the duration it takes for the system to react to users requests, specifically in the context of generating a graph based on user input after they click plot graph. Efficiency enhances the user experience and reduces the response time of the system. Initially, without any query optimisation techniques in place, this process of querying the data and transforming it into a graph was time-consuming due to the substantial amount of data the system needed to process to retrieve relevant information. However, by implementing optimisation techniques like indexing, projection, and aggregation, The system was largely improved in terms of response time. The actual response time depends on two factors. First, it depends on the time period, i.e. date range, chosen by the user as longer time periods result in more data to process and therefore longer response times. Secondly, it relies on the specific months selected for the query. Some months contain a high volume of documents, potentially up to 50,000, while others may have as few as 1,000 or even fewer documents. For example, in a scenario where a user wants to analyse SSH versions used in the year 2021 and determine the percentage of users for each version, with optimisation techniques in place, this task takes approximately 5 minutes to query the database, compile the relevant documents, and create a graph. This estimate assumes a stable internet connection. However, without the benefits of indexing and projection, the response time would be much longer, up to two hours.

B. Functionality Testing

Another important performance metric employed to assess the proposed method was functionality testing. Functionality testing verifies whether the proposed method is operating as intended. Specifically, it involves confirming that the graphs generated by the tool are accurate and correct. Functionality testing is important for this project as the results of the proposed tool directly influence the reliability and trustworthiness of the information presented to users. To test the functionality of the system, the MongoDB database was queried using Mongosh, a MongoDB shell, to retrieve the raw data. Next, the results obtained from these direct database queries are compared to the data presented in the generated graphs. If the data extracted via Mongosh queries matches the data depicted in the graphs, it signifies that the proposed method accurately represents the dataset. MongoDB queries via Mongosh are inherently accurate because Mongosh allows direct access to the MongoDB database, ensuring that the data retrieved is an accurate representation of the database contents without intermediaries or potential data transformation issues such as data format alterations or unintended modifications. The subsequent figures show how the database was queried for functionality testing. Fig. 4 presents the specific query used to determine the top six most commonly used HTTP proxy versions within the dataset. It additionally showcases the counts associated with each of these versions. The query results display the following:

- 'F5 BIG-IP load balancer http proxy': 776
- 'HAProxy http proxy 1.3.1 or later': 244
- 'EZproxy web proxy': 62
- 'SonicWALL SSL-VPN http proxy': 21
- 'Squid http proxy': 14
- 'bad gateway': 9

Fig. 5 shows a pie chart representation of the data derived from fig. 4. By comparing the percentages and counts displayed on the pie chart with the results from the query, it's evident that they match. This alignment between the query results and the graphical representation assures that the tool is extracting, processing, and presenting data in the intended manner, highlighting its effectiveness in functionality testing.

```
mydb> db.mycollection.aggregate([{$match: {"openports": {"$elemMatch": {"port": 80, "service":
 "http-proxy", "version": {"$exists": true}}}}}, {$unwind: "$openports"}, {$match: {"openports
.port": 80, "openports.service": "http-proxy", "openports.version": {"$exists": true}}}, {$gro
up: {_id: "$openports.version", count: {$sum: 1}}}, {$sort: {count: -1}}, {$limit: 6}])

[
  { _id: 'F5 BIG-IP load balancer http proxy ', count: 776 },
  { _id: 'HAProxy http proxy 1.3.1 or later', count: 244 },
  { _id: 'EZproxy web proxy ', count: 62 },
  { _id: 'SonicWALL SSL-VPN http proxy ', count: 21 },
  { _id: 'Squid http proxy ', count: 14 },
  { _id: '(bad gateway)', count: 9 }
]
```

Fig. 4. Query and results for the top six HTTP proxy versions used.

VI. Results

After implementing the proposed method and using its capabilities, we put together a report for ZX Security; this is the project's final outcome. In the report, the focus was on various key aspects that are crucial for understanding the security and configuration trends. These categories include Microsoft IIS web server patching rates over time, monitoring the web server versions in use, and how they have changed. This information is useful to know because outdated or unsupported

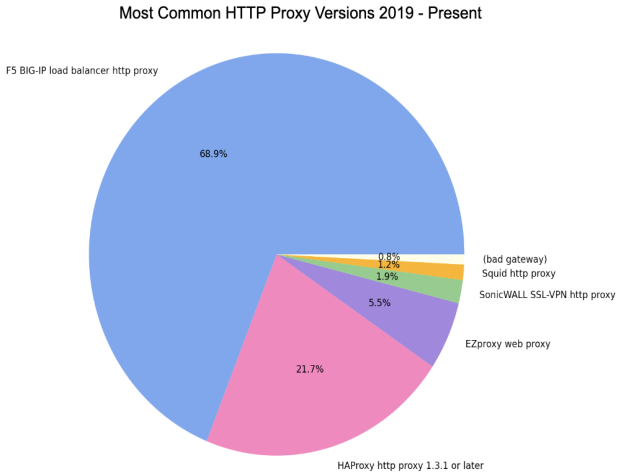

Fig. 5. Pie chart generated by the proposed method of the top six HTTP proxy versions used.

versions may contain vulnerabilities that could be exploited by malicious actors. Additionally, the report includes the frequency of users relying on certificates signed by Certificate Authorities (CAs) as opposed to self-signed certificates. It also highlights the popular CAs in use and the ratio of expired to valid certificates. The report explores open ports on devices including what those ports are used for, vulnerabilities associated with them, and how they have changed over time, this information can help in uncovering potential entry points for cyberattacks. Another section included in the report is about the adoption of firewalls and IP cameras during different time periods and how they contribute to security. Adoption trends of firewalls and IP cameras across different time periods is important for assessing network security infrastructure because their presence or absence can impact the network's strength against threats and unauthorised access. Lastly, the report looks into the different versions of SSH being used. Analysing the versions of SSH in use helps in assessing adherence to the latest security standards as outdated SSH versions may expose the network to known vulnerabilities. By examining these areas, the report helps to gain insights into potential vulnerabilities, security practices, and technology adoption patterns of people in New Zealand. The following sections, derived from the report, demonstrate the insights obtained using the graphs produced by the proposed method.

A. Self-signed and CA-signed certificates in 2020 and 2022

Fig. 6 presents a stacked bar chart produced by the tool developed in this study, depicting the percentage of certificates that were self-signed and CA-signed in 2020 and 2022. Of those that were CA-signed, it shows which certificate authority signed it. From these graphs, we can observe that in 2022, there has been an increase in the use of self-signed certificates compared to 2020. Self-signed certificates lack the validation provided by trusted authorities, leading to warning messages for users and potentially eroding site security confidence. These certificates are also vulnerable to man-in-the-middle

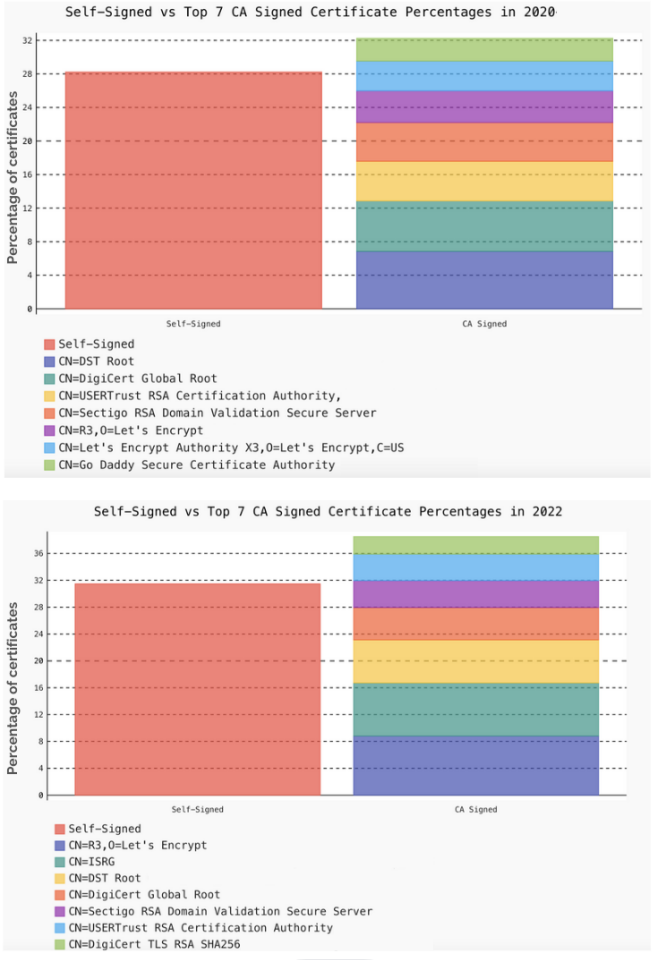

Fig. 6. Self signed and CA-signed certificates in 2020 and 2022.

attacks, where attackers exploit the absence of validation. While self-signed certificates offer quick encryption, the shift emphasises the trade-off between ease, cost, and security. Organisations should use them cautiously, as obtaining certificates from reputable authorities remains a safer choice for ensuring user trust and strong security. In both 2020 and 2022, Let's Encrypt stood out as the predominant certificate authority, reflecting its wide acceptance and favor among users. This preference can largely be attributed to its popularity, as well as the distinctive nature of Let's Encrypt as a nonprofit organisation with a mission rooted in enhancing web security and privacy [18]. By advocating for the widespread adoption of HTTPS, Let's Encrypt aims to foster a more secure and private online environment. Their services are characterised by accessibility, being both free and user-friendly, allowing websites of all sizes to effortlessly implement HTTPS and strengthen their security protocols. Digicert emerges as the next popular choice, despite its relatively high cost of around $300 annually [19]. This popularity can be attributed to its exceptional offerings, including 24/7 customer support, which has earned it the reputation of being the highest-rated Certificate Authority for customer service globally [20]. Digicert's provision of free reissues and replacements for the entire certificate's lifespan adds to its popularity. Though pricier than other options,

organisations prioritising data protection might find Digicert appealing for its comprehensive support and assurance. The increase in self-signed certificates in 2022 compared to 2020 can be attributed to factors such as cost considerations and ease of implementation. Obtaining certificates from trusted authorities often comes with a cost. In an effort to reduce expenses, some organisations and individuals may opt for self-signed certificates, especially if their websites are primarily for internal use or testing. Self-signed certificates are relatively easy to create and implement, making them a quick solution for securing web traffic. This convenience may have led to their increased use, especially among smaller websites and personal projects.

B. Microsoft IIS Updates

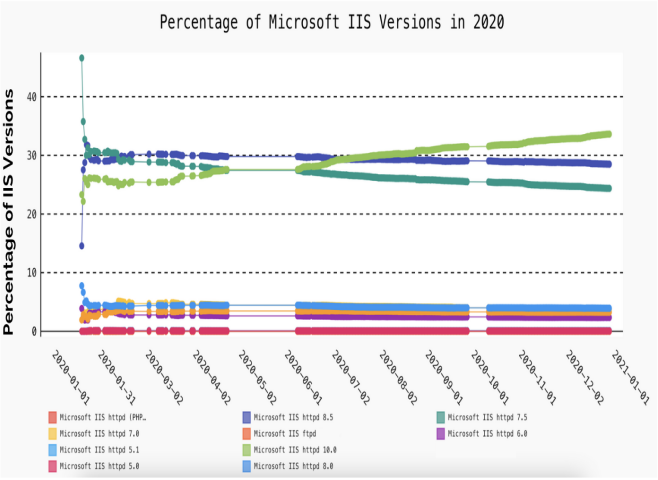

Fig. 7. Percentage of Microsoft IIS Updates in 2020.

Fig. 7 provides insight into the different versions of Microsoft IIS being used by hosts in New Zealand in 2020. From the graph, we can see the majority of users are adopting more recent versions, specifically 7.5, 8.5, and 10, and the usage for these versions shows an upward trajectory over time. This indicates a positive response to security updates and improvements. In late June, there was a notable increase in the number of hosts updating to version 10. This spike coincided with Microsoft issuing an urgent alert about a vulnerability within the Internet Information Service (IIS) and recommending immediate server patching due to a surge in cyberattacks exploiting this vulnerability. The updates during this period could be attributed to heightened awareness and the immediate need to address potential security vulnerabilities. However, a small subset of users continues to employ versions below 7.5. This practice is problematic as Microsoft has discontinued support for these older versions, leaving them vulnerable to known security issues. In light of these findings, it's important for organisations to prioritise migrating to supported and more secure IIS versions to safeguard their systems from potential vulnerabilities.

VII. Limitations

The project had certain limitations in the dataset collected by ZX Security. Firstly, the scans conducted to gather documents lacked consistency. These scans were not performed at regular time intervals, making it challenging to establish a uniform baseline for analysis. Additionally, the number of documents scanned during each session exhibited large variability, ranging from as few as 1,000 documents per month to over 50,000 documents per month. A notable gap in the dataset occurred in May 2020 when ZX's server was temporarily switched off, resulting in the absence of any data representing activities during that month. These limitations can largely affect the reliability and interpretability of the data. Inconsistencies in the timing and volume of document scans introduce uncertainty into the dataset. For instance, the lack of consistent time intervals between scans makes it challenging to track changes or trends over specific time frames accurately. Additionally, the wide variability in the number of documents collected in each scan session can skew the data and potentially lead to misinterpretations. To address these challenges, converting the data into percentages helped make it more comparable and useful. Although converting data into percentages helps in making comparisons more consistent, it cannot completely mitigate the effects of irregular data collection practices. The underlying inconsistencies in timing and document counts remain, which can still introduce potential biases into the analysis. Therefore, while percentage-based representation improves data comparability, it is still important to acknowledge that the dataset's limitations exist and are taken into account when interpreting the findings in the report.

VIII. Conclusion

In conclusion, this project aimed to develop a tool to visualise trends from ZX Security's data and analyse these trends in a report. The project involved several key components, including query optimisation, data processing and aggregation, graph generation, and a report containing the findings. The proposed method successfully met all its objectives, providing valuable insights into various aspects of network security. Through the use of graph visualisations, the proposed method allowed users to analyse trends in Microsoft IIS web server patching rates and certificate usage. These insights are critical for identifying potential vulnerabilities and areas needing improvement. The tool's performance was evaluated in terms of efficiency and functionality. Optimisation techniques, such as indexing and projection, largely improved efficiency, reducing the time required to generate graphs. Functionality testing confirmed that the proposed method accurately represented the dataset, enhancing its reliability and trustworthiness. Despite the limitations of the dataset, the method provided valuable insights and contributed to a comprehensive report that highlighted trends in various areas of cyber security from the dataset provided by ZX Security.

IX. Future Work

A. Enhanced GUI

Introducing a feature to save and load queries could save users' time when revisiting past graphs. This feature would allow users to store their preferred query configurations for future use.

Enabling users to export graph data in various formats (e.g., CSV, Excel) can facilitate further analysis or reporting outside of the proposed method.

By providing the capability to display multiple graphs simultaneously, users can compare trends across different variables or time periods, allowing for more comprehensive insights.

Creating a customisable dashboard where users can arrange and view multiple graphs, charts, and data tables in a single layout. This feature simplifies data interpretation and trend analysis.

B. Predictive Analytics and Machine Learning

Combining predictive analytics and machine learning into the tool can unlock powerful capabilities for understanding and addressing cyber trends. Predictive analytics can help forecast future trends based on historical data, while machine learning can detect anomalies and patterns that may not be apparent through traditional analysis methods. Predictive analytics can foresee the next surge in patching activity or forecast new security technology adoption rates. Machine learning algorithms can detect unusual patterns in network behaviour that might indicate a security breach or the presence of malware. For example, machine learning can identify patterns of unusual network traffic. Additionally, machine learning models can be trained to classify security events and incidents, allowing for automated alerting and response. For instance, a machine learning model could classify network events in the data as normal, suspicious, or malicious, helping to prioritise response efforts. By combining these techniques, the tool can offer a proactive approach to network security, it can help organisations stay ahead of emerging threats and vulnerabilities. Additionally, it can provide valuable insights into network performance, and user behaviour.

In summary, an enhanced GUI and the integration of predictive analytics and machine learning can make the proposed method more user-friendly, insightful, and proactive.

Acknowledgments

I want to express my thanks to Harith Al-Sahaf, Tomais Williamson, and Stephen Shkardoon for their support and guidance throughout the year on this project.

References

[1] *Goal 4 — Department of Economic and Social Affairs*. United Nations, 2022. [Online]. Available: https://sdgs.un.org/goals/goal4 (accessed May 24, 2023).

[2] *Tableau Software Review: Pros and Cons of a BI Solution for Data Visualization*. SAM Solutions. 2019, July 30. Retrieved March 25, 2023, from https://www.samsolutions.com/blog/tableau-software-review-pros-and-cons-of-a-bi-solution-for-data-visualization/

[3] *JSON Connector Documentation*. Microsoft. Retrieved March 25, 2023, from https://learn.microsoft.com/en-us/power-query/connectors/json

[4] *Elasticsearch Query DSL Documentation*. Elastic. Retrieved March 25, 2023, from https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html

[5] *Elasticsearch Features*. Elasticsearch Features. Retrieved October 13, 2023, from https://www.elastic.co/elasticsearch/features

[6] *JSON and BSON*. MongoDB. Retrieved March 25, 2023, from https://www.mongodb.com/json-and-bson

[7] *mysql*. *MySQL Development*. MySQL Development. Retrieved October 13, 2023, from https://dev.mysql.com/

[8] *postgressql*. *PostgreSQL*. PostgreSQL. Retrieved October 13, 2023, from https://www.postgresql.org/

[9] *Python*. Python.org. Retrieved March 25, 2023, from https://www.python.org/

[10] *Pygal Documentation*. Pygal Documentation. Retrieved March 25, 2023, from https://www.pygal.org/en/stable/documentation/index.html

[11] *Sharding: A Database Optimization Technique*. TechTalks by Anvita. Retrieved March 29, 2023, from https://www.techtalksbyanvita.com/post/sharding-a-database-optimization-technique

[12] *Matplotlib*. Matplotlib.org. Retrieved March 25, 2023, from https://matplotlib.org/

[13] *Vega*. Vega GitHub Pages. Retrieved June 28, 2023, from https://vega.github.io/vega/

[14] *Python Tkinter Documentation*. Python Tkinter Documentation. Retrieved July 18, 2023, from https://docs.python.org/3/library/tkinter.html

[15] *Qt for Python Quick Start Guide*. Qt for Python Quick Start Guide. Retrieved July 18, 2023, from https://doc.qt.io/qtforpython-6/quickstart.html#quick-start

[16] *wxPython Documentation*. wxPython Documentation. Retrieved July 18, 2023, from https://docs.wxpython.org/

[17] *Canva*. Canva. Retrieved August 1, 2023, from https://www.canva.com/

[18] *Let's encrypt*. *Let's Encrypt FAQ*. Let's Encrypt FAQ. Retrieved October 13, 2023, from https://letsencrypt.org/docs/faq/

[19] *Digicert price*. *DigiCert Order*. DigiCert Order. Retrieved October 13, 2023, from https://order.digicert.com/

[20] *Digicert support*. *DigiCert Contact Us*. DigiCert Contact Us. Retrieved October 13, 2023, from https://www.digicert.com/contact-us