Surveying .NZ TLS

Lucas Sarten

***Abstract*—The Internet is a critical platform for exchanging sensitive information, encompassing businesses such as banks, online stores, and government agencies that routinely transmit and receive sensitive personal data over the Internet. Ensuring the security of this information is of paramount importance.**

The security protocols Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), play a vital role in securing the web. These protocols implement various security measures to ensure the confidentiality, integrity, and authenticity of the data exchanged between web browsers and web servers. Nonetheless, the improper configuration and implementation of SSL/TLS protocols on web servers can introduce security vulnerabilities, potentially compromising user privacy and security.

This report contains the results and conclusions derived from a comprehensive survey of SSL/TLS configurations among websites registered under the .nz top-level domain name. We evaluated the current SSL/TLS configurations in place and compared them with the latest web security standards and best practices. Our findings provide valuable insights into the security posture of websites within the .nz domain and highlight potential risks, vulnerabilities, and areas of concern. By doing so, we contribute to the broader understanding of web security best practices and help businesses, policy-makers, and individuals make informed decisions to enhance the security of sensitive online data.

I. Introduction

THE security of sensitive user data on the web is a critical concern in the digital age. The widespread exchange of personal information with e-commerce, online banking, and other sensitive services underscores the urgency of safeguarding this data against cyber threats. However, many websites fail to implement recommended security standards, jeopardising user data privacy and security.

This project addresses a pressing issue in web security: configuring the SSL/TLS protocols on websites within the .nz domains. TLS and its predecessor SSL are the fundamental protocols designed to ensure the privacy, integrity, and authenticity of data exchanges between web browsers and web servers. However, when misconfigured, they can introduce security vulnerabilities that expose users to risks.

The primary goal of this project is to assess the state of SSL/TLS configurations across .nz websites and determine whether they meet or exceed recommended security standards. We aim to quantitatively evaluate these configurations, setting clear and measurable specifications that inform every aspect of our study.

To accomplish this, we will use specialised tools to scan and collect detailed information on the SSL/TLS configurations of approximately 200,000 .nz websites. The data will include details such as supported SSL/TLS protocol versions, cypher suites, public key certificates, Hypertext Transfer Protocol Secure (HTTPS) adoption, and proper redirections. This analysis must make special considerations to ensure that all network scans are conducted non-intrusively with explicit bandwidth, frequency, and concurrency considerations. Furthermore, all data collected from scanning externally facing web servers must be public information, ensuring strict adherence to privacy laws and regulations.

The analysis will identify vulnerabilities and weaknesses resulting from implementing SSL/TLS on each website. This project aims to provide website owners, web users, and the broader community with an in-depth understanding of the current security posture within the .nz domain.

The project's deliverables include a summarised database containing the collected SSL/TLS data, security vulnerability probe results, and HTTPS adoption statistics.

Key metrics of evaluation will include the strength of supported cypher suites, which are cryptographic algorithms responsible for authenticating and encrypting SSL/TLS traffic, ensuring that websites reject insecure and obsolete cypher suites. Additionally, we will examine the proper traffic redirection from unsecured Hypertext Transfer Protocol (HTTP) to secure HTTPS connections. Furthermore, we will explore common SSL/TLS vulnerabilities that may be present among .nz websites. This analysis includes vulnerabilities related to cypher suite support, SSL/TLS versions, and insecure or invalid certificates.

The scope of this survey includes all websites registered under the .nz top-level domain, signifying a broad cross-section of New Zealand's online entities. This project aims to assess whether an acceptable security standard is maintained across websites registered under the .nz top-level domain and promote web security best practices within New Zealand. The findings of this project will provide insights into the current state of web security in New Zealand and serve as a foundation for recommendations to enhance the security of .nz websites further.

In summary, this project seeks to clarify the state of SSL/TLS security configurations in the .nz domain, addressing a problem that affects both website owners and users alike. The results of this survey will empower stakeholders with valuable information to make informed decisions and improve web security, contributing to the broader effort to ensure the confidentiality and integrity of sensitive online data.

A. Background

SSL/TLS configurations are the crucial backbone of secure communication between our web browsers and web servers [1]. However, individual web servers can have varying and potentially insecure configurations, allowing insecure methods of communication which pose security risks. Understanding the SSL/TLS security configurations used by websites registered

This project was supervised by Dr Arman Khouzani.

in New Zealand is essential for assessing the overall state of internet security within the country. Newer versions of TLS provide many security benefits over previous versions, with TLS 1.3 being the current gold standard for best security practices. The Internet Engineering Task Force (IETF) officially deprecated TLS 1.0 and TLS 1.1 in March 2021, highlighting the vulnerable nature of older protocols [2].

The use and integrity of public key certificates for encryption is a critical aspect of web security that will also be analysed. Public key certificates play the vital role of verifying the identity of websites visited by users [3]. This process of verifying the identity of a web server uses asymmetric encryption with public and private keys. Websites verify their identity by presenting a public key certificate digitally signed by a trusted Certificate Authority (CA). The user's browser will then verify the certificate's validity with the third party CA, confirming that the web server is who it claims to be. Missing or invalid public key certificate configurations, such as expired certificates, can threaten data confidentiality. One typical example of a potential security risk posed by an invalid certificate is a man-in-the-middle attack [4]. Without a valid public key certificate, the user's web browser cannot verify the Web server's identity, which could allow attackers to "pretend" to be the web server and intercept and manipulate sensitive information sent between the two parties.

Furthermore, the use of HTTP and HTTPS protocols for serving web content is another crucial area of web security. HTTPS is HTTP over SSL/TLS and is paramount in ensuring data is transmitted securely between a user and a website. Lack of support for HTTPS or the usage of HTTP for any web content may introduce security and privacy concerns. If a web server uses HTTP instead of HTTPS and the user's browser allows this connection, any data sent between the two parties is unencrypted and unauthenticated [5]. This communication would allow attackers with access to these networks to intercept and manipulate any data sent over the unsecured connection, including potentially private and sensitive information.

II. Related Work

The security of web communications has been an ongoing area of research within the cybersecurity community. Numerous studies have examined the configurations of SSL/TLS protocols with various techniques and methodologies. Moreover, many tools currently exist in the field of SSL/TLS security assessment, specifically designed to extract low-level SSL/TLS configurations from web servers. This section will discuss some of these works, their main findings, and how they relate to our study.

A. SSL/TLS Configuration Assessments

1) F5 Labs: A report by cybersecurity company F5 Labs [6] presents detailed statistics on the current state of TLS usage across the top one million websites from the Tranco Top 1 Million list [7]. F5 Labs developed an open-source tool, Cryptonice, which scans websites and reports basic protocol and cypher support, HTTP redirects, Domain Name System (DNS) records, and certificate information [8].

The report, based on data collected as of August 2021, indicates that TLS version 1.3 is currently the most preferred TLS protocol with 63% of the one million websites scanned designating TLS 1.3 as the preferred protocol (Figure 1). Compare this to two years ago when the same study found that only 32% of web servers defaulted to TLS 1.3 [6]. The report continues to outline that 0.4% of websites still prefer TLS 1.0, and 0.002% still prefer SSL 3.0. The study also found no correlation between website popularity and SSL/TLS protocol support.

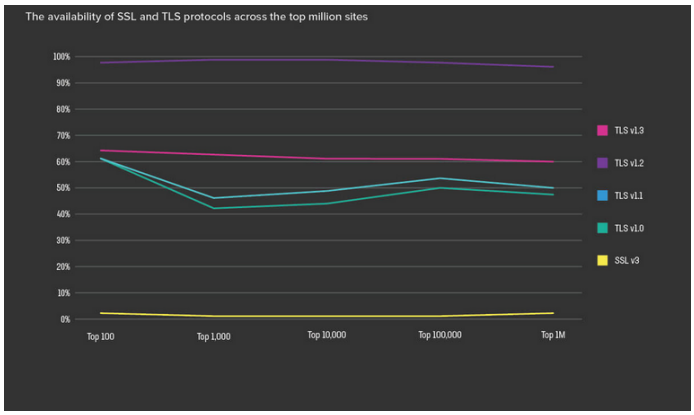

Fig. 1. Availability of SSL/TLS Versions Across the Top Million Sites F5 Labs [6].

Another crucial aspect of SSL/TLS configurations is the choice of handshake mechanisms. The report reveals that over 99.3% of servers now choose non-Rivest-Shamir-Adleman (RSA) handshakes when possible. However, the report also shows that 52% of web servers still allow RSA handshakes (Table I). RSA handshakes are not inherently insecure, but they are more susceptible to specific attacks and fail to provide forward secrecy [9].

TABLE I
The Most Popular Selected Cypher Suites Across the Top Million Sites F5 Labs [6].

Protocol	Portion	Cypher Suite Chosen by Web Server
TLS 1.3	56.8%	TLS_AES_256_GCM_SHA384
TLS 1.2	18.4%	ECDHE-RSA-AES256-GCM-SHA384
TLS 1.2	12.6%	ECDHE-RSA-AES128-GCM-SHA256
TLS 1.3	5.4%	TLS_AES_128_GCM_SHA256
TLS 1.2	1.9%	ECDHE-RSA-AES256-SHA384
TLS 1.2	1.4%	ECDHE-RSA-CHACHA20-POLY1305
TLS 1.3	0.45%	TLS_CHACHA20_POLY1305_SHA256
TLS 1.2	0.4%	ECDHE-ECDSA-AES256-GCM-SHA384
TLS 1.2	0.4%	ECDHE-ECDSA-CHACHA20-POLY1305
TLS 1.2	0.4%	DHE-RSA-AES256-GCM-SHA384
TLS 1.2	0.3%	ECDHE-ECDSA-AES128-GCM-SHA256
TLS 1.0	0.3%	DHE-RSA-AES256-SHA

The study found that 2.8% of websites were vulnerable to TLS 1.2 session renegotiation denial-of-service (DOS), 1% of websites were vulnerable to the CRIME exploit due to supporting TLS compression [10], and 0.2% of websites did not support session renegotiation [6]. Our study will compare the presence of these vulnerabilities with the .nz domain and

seek to capture data on additional common vulnerabilities not part of the original study.

While the F5 Labs study provides quite comprehensive results, it has limitations. The F5 Labs study concluded in August 2021, so it no longer accurately depicts the rapidly evolving web landscape. This historical information will serve as a comparison for our project, which aims to extend this research by showing how this data has evolved. Furthermore, the F5 Labs study only focused on the top one million most popular websites, which may not accurately represent the average website. F5 Labs compares the SSL/TLS versions of a broad sample of countries; however, New Zealand is not the focus of their study and needs representation in their findings. Our study solely examines websites with .nz top-level domains, enabling comparison with New Zealand.

2) SSL Pulse: Another report published monthly by Qualys SSL Labs [11] examines the state of SSL/TLS implementation across a dataset of approximately 135,000 of the top websites based on Alexa's list of the most popular sites in the world, which ranks website based on visits [12]. Similar to the F5 Labs study, the SSL Pulse report found that, as of September 3rd 2023, 64.8% of the top 135,000 most popular websites supported TLS 1.3, as shown in Figure 2. Additionally, the report found that an overwhelming 99.9% of websites supported at least TLS 1.2, showing that TLS 1.2 remains the most widely supported TLS protocol. The report continues to show that 32.5% of websites still support TLS 1.1, and a similar proportion, 30.1%, still support the long deprecated TLS 1.0.

The report also investigated common vulnerabilities of common SSL/TLS, finding small proportions of websites still vulnerable to well-known exploits. Specifically, the report mentions that 85 websites are still vulnerable to the ROBOT attack, 52 websites are vulnerable to OpenSSL ChangeCipherSpec (CCS) attacks, and 18 websites exhibit vulnerabilities associated with the Heartbleed bug as per Table II. Our research in the .nz domain will establish a basis for comparison, allowing us to assess the presence of these common SSL/TLS vulnerabilities compared to the global internet landscape.

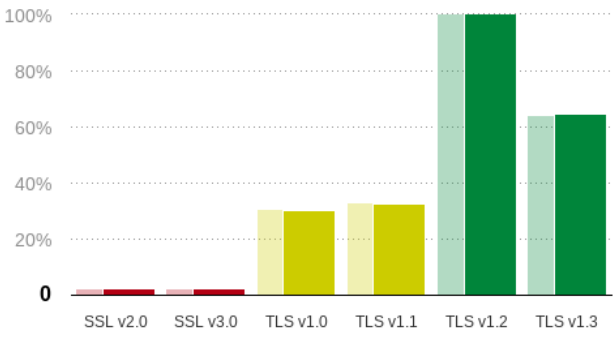

Fig. 2. SSL/TLS Protocol Support Across the Top 150,000 Sites SSL Pulse [11].

TABLE II
COMMON SSL/TLS VULNERABILITIES SSL PULSE [11].

Vulnerability	Number of Websites
OpenSSL CCS	52
ROBOT	85
Heartbleed	18

B. Certificates and HTTPS Usage

1) F5 Labs [6]: Regarding certificates, the F5 Labs study found that 42% of certificates surveyed had a lifespan of 91 days, while 46% of certificates had a lifespan of 361-450 days (Figure 3). Moreover, the study found that 2.45% of the top one million websites had expired certificates at the time of the survey. There is a clear pattern with certificate authorities issuing certificates with shorter lifespans.

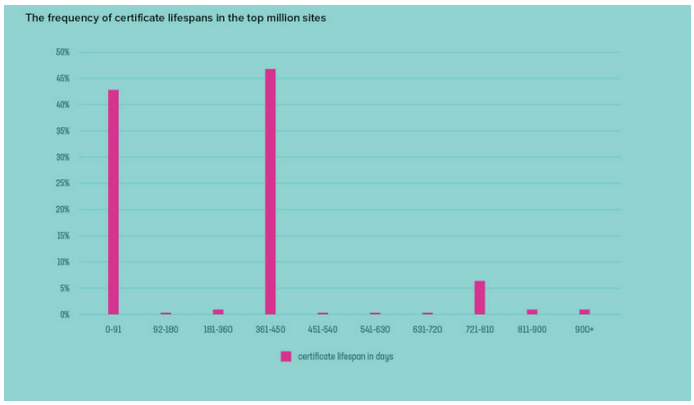

Fig. 3. Certificate Lifespans Across the Top Million Sites F5 Labs [6].

Furthermore, the study found that 25% of certificates are now signed with the Elliptic Curve Digital Signature Algorithm (ECDSA), while the remaining 75% still use traditional RSA signatures (Figure 4). ECDSA is considered a robust and efficient security algorithm, suggesting a growing commitment towards more secure cryptographic algorithms on the web.

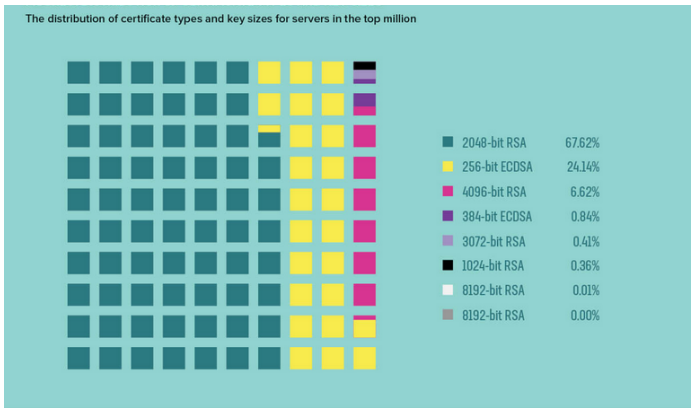

Fig. 4. Certificate Key Type/Size Across the Top Million Sites F5 Labs [6].

C. Existing Solutions

1) SSLyze: SSLyze is an open-source Python module that provides comprehensive SSL/TLS scanning capabilities [13].

It captures full SSL/TLS configuration details, including complete lists of supported cypher suites, certificate chains, and susceptibility to a limited number of known security vulnerabilities [13]. Vulnerabilities caused by using susceptible cypher suites can be detected, such as the Heartbleed or ROBOT vulnerabilities [14], [15]. SSLyze also provides a script-friendly output and a fully featured Python Application Programming Interface (API) for simple integration with external tools. SSLyze is freely available under the GNU Affero General Public License [16], allowing unconditional use of the software.

The main limitation of SSLyze is the lack of built-in concurrency [13]. SSLyze does not have native support for performing multiple scans concurrently, which is a crucial consideration when working with a large data set. Additionally, Python imposes significant limitations on concurrency due to the limitations of the Python Global Interpreter Lock (GIL). The GIL is a mechanism of the Python interpreter that only allows one thread to execute at any given time [17]. This restriction limits the scalability of SSLyze as Python can not fully utilise multiple CPU cores for parallel execution of CPU-bound tasks such as this.

SSLyze's comprehensive analysis and vulnerability scanning make it a good choice for thorough security assessments. Its Python API provides the flexibility to easily extend the functionality of the SSLyze and integrate it with external tools, further adding to its utility.

2) tls-scan: Another existing tool for SSL/TLS scanning is tls-scan, a command-line tool written in the C programming language [18]. Like SSLyze, tls-scan can extract SSL/TLS configurations from web servers. The main advantage of tls-scan over SSLyze is its ability to scan thousands of servers concurrently, making it a popular tool for large-scale assessments [18].

However, tls-scan captures significantly less information than SSLyze, providing a more general snapshot overview of a server's SSL/TLS configuration. It is also important to note that tls-scan does not have built-in security vulnerability testing, only focusing on capturing and analysing certificates and SSL/TLS configurations.

tls-scan is an attractive option with its efficient parallel scanning capability, allowing for quick assessment of SSL/TLS configurations across many servers.

3) crashtest-security: crashtest-security is a comprehensive web security testing platform designed to detect common vulnerabilities and assess the overall security of websites [19]. The platform can scan for various SSL/TLS and certificate vulnerabilities, including weak cypher suites, outdated protocols, and certificate issues. Furthermore, the service includes web security compliance scanning for various regulations and standards, such as The Health Insurance Portability and Accountability Act (HIPAA), General Data Protection Regulation (GDPR), and International Organization for Standardization (ISO) [20]–[22]. The platform can provide detailed reports on these findings aimed at commercial users and project-based web security testing.

While crashtest-security offers valuable features for web security testing, the platform has limitations which make it unsuitable for the use case of this study. The basic subscription level for the platform costs €79.00 per month, limited to two concurrent scans. This limitation makes the service unusable to scan the dataset of 200,000 .nz domains within the time frame for this study.

4) ssl-enum-ciphers: ssl-enum-ciphers is a Command Line Interface (CLI) tool written in the Lua programming language designed to collected SSL/TLS configuration information [23]. The tool repeatedly initiates SSL/TLS connections with differing cypher suites to determine if the web server accepts each option. The tool rates each cypher suite with a letter grade from A to F, indicating the cryptographic strength of the cypher suite. Ratings are based on the Qualys SSL Labs SSL Server Rating Guide [24].

The tool, however, does not contain any built-in concurrency or scalability support for scanning multiple web servers simultaneously. This is one of the most essential requirements for assessing a large dataset of domains efficiently, making this tool unsuitable. ssl-enum-ciphers also describes itself as "intrusive since it must initiate many connections to a server, and therefore is quite noisy", which violates our requirement that the scanning process must not be intrusive or cause disruption [23].

5) testssl: testssl is another open-source SSL/TLS configuration tool written in Bash capable of capturing SSL/TLS and certificate information [25]. The scans for SSL/TLS versions, ciphers, and some simple SSL/TLS vulnerabilities and misconfigurations.

However, much like ssl-enum-ciphers II-C4, the tool does not have built-in support for scanning multiple domains simultaneously. Therefore, due to the scale of the survey, the tool would not be suitable for this use case.

6) Cryptonice: Cryptonice is an open-source tool written in Python developed by the cybersecurity testing company F5 Labs [26]. Cryptonice is a comprehensive tool incorporating open-source tools such as SSLyze [13]. The tool is capable of collecting SSL/TLS information similar to SSLyze, and tests support for additional protocols such as HTTP2 and DNS. Cryptonice also collects information on certificates and web application headers.

While Cryptonice does perform additional tests that other tools do not, the information collected for each scan is limited compared to alternatives. Cryptonice is targeted at generating human-readable output for security engineers to test their web applications and does not include built-in concurrency. Furthermore, Cryptonice scans take substantially longer to complete, taking 70x longer on average compared to a single SSLyze scan. These drawbacks make the Cryptonice unsuitable for large-scale scanning, but the tool could still serve a purpose for in-depth targeted scans.

III. Design

A. Scanning Tools

To assess the security of websites within the .nz top-level domain, we meticulously designed our data collection process to extract and refine the relevant data promptly. Our survey evaluated three key aspects: SSL/TLS configurations, public

key certificate health, and the usage of HTTP protocols. Our approach aimed to provide an in-depth analysis of these three critical aspects while addressing the unique challenges of scanning many websites. When selecting tools and platforms for this project, we prioritised the depth of data collected and the extendability and scalability of the data collection tools. We employed a combination of automated tools, utilising both preexisting solutions and custom-made tools tailored to our specific needs.

For collecting SSL/TLS configurations, we used the open-source tool SSLyze over alternatives such as tls-scan to collect the relevant configuration from each server [13]. SSLyze provides more comprehensive data than tools like tls-scan and F5 Lab's Cryptonice. SSLyze boasts superior extendability compared to the alternatives, with a built-in Python API, making it the most suitable tool for multiprocessing. Multiprocessing was crucial to enhance efficiency and expedite data collection. The information SSLyze collects includes information on the supported versions of SSL/TLS, the cypher suites supported for each version, susceptibility to known vulnerabilities, and public key certificate chains, which include the issuer, expiration date, and cyphers.

For collecting HTTP usage data, custom tools would need to be created as existing solutions did not meet the cost and time requirements of the project. This data included identifying instances of partial HTTP usage for serving content and the implementation of HTTP-HTTPS redirects. Go was chosen as the most suitable programming language for this task due to its inherent support for multiprocessing, aligning with our goal of creating efficient tools for scanning large datasets.

1) SSLyze vs tls-scan: The choice of scanning tool plays a pivotal role in the efficacy of our large-scale security scans. Network performance is a critical aspect to consider when selecting a tool, as it directly impacts the performance of the scanning tools and the potential impact of the surveyed websites. Understanding how potential tools interact with web servers and their impact on network resources is essential to ensure efficient data collection and adhere to ethical standards. The two main tools of interest in this comparison are SSLyze and tls-scan [13], [18].

Figure 5 shows the network performance of SSLyze in terms of packets per second. The graph reveals the balanced and predictable operation of the tool, maintaining a constant rate of approximately 500 inbound and 500 outbound packets per second per scan. This steady packet flow aligns well with ethical considerations as a more distributed bandwidth reduces strain on surveyed websites. We can gather from this data that SSLyze makes sequential requests, performing each scan in sequence rather than parallel.

On the other hand, Figure 6 shows the network performance graph for tls-scan, which exhibits distinctly different behaviour. The network graph indicates an initial spike of approximately 600 packets, followed by a period of limited activity, and concludes with an additional spike of inbound packets. It appears that tls-scan adopts a different approach to scanning, sending a significant portion of requests at once before waiting for replies. This may lead to intermittent bursts of network activity that could potentially impact the websites being surveyed, with the longer delay in responses potentially indicating this pattern.

This investigation suggests that SSLyze is a more suitable tool as it aligns more closely with this study's efficiency and ethical requirements. The more balanced and predictable nature of SSLyze's scanning process will have a lower impact on surveyed websites, distinguishing it as the better option for this use case.

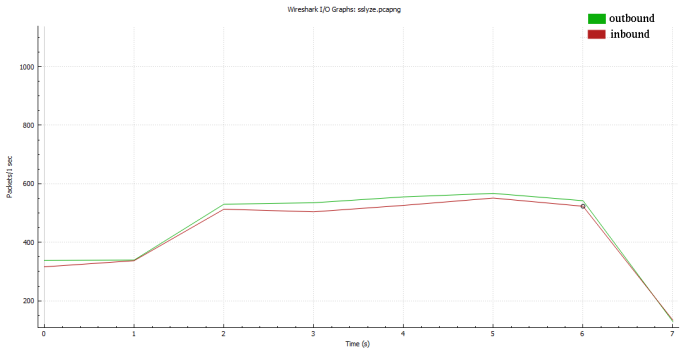

Fig. 5. SSLyze Network Bandwidth over Time.

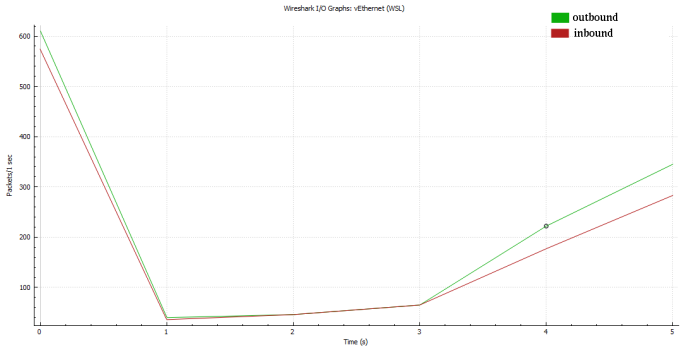

Fig. 6. tls-scan Network Bandwidth over Time.

B. Cypher Suite Grading

To assess the overall security status of websites within the .nz domains, we must evaluate the security of cypher suites utilised in websites SSL/TLS configurations. Cypher suites are pivotal components of secure SSL/TLS configurations as they dictate how data is encrypted and decrypted during transmission.

The data for cypher suite grading was sourced from ciphersuite.info, an online open-source dataset that provides comprehensive information about the cryptographic security of cypher suites [27]. The database grades cypher suites from insecure, weak, secure, and recommended based on their security strength. Strong cypher suites using Advanced Encryption Standard (AES) are rated secure, while those utilising deprecated algorithms are rated weak or insecure.

The choice of cypher suite can significantly impact a website's security. By examining the cypher suites used amongst .nz websites, we can gain valuable insights into the security postures of these websites.

C. Data Storage

We chose NoSQL as the base database type to store and manage large volumes of collected data effectively. A NoSQL database solution was a natural choice as it can accommodate unstructured data storage. It is anticipated that database records may contain missing or unstructured data throughout the data collection process. The diverse range of tools utilised meant dealing with multiple data formats and structures. NoSQL's flexibility ensured data structure discrepancies were supported, allowing for a unified platform for data storage from multiple sources.

IV. Implementation

A. SSL/TLS Scanning

The custom tools for running concurrent SSL/TLS scans have been developed. These tools have undergone testing and refinement to ensure their effectiveness, accuracy, and efficiency in collecting the required data. We built a custom wrapper using the SSLyze Python API to implement the multiprocessing functionality of the tool. Our solution performs numerous SSLyze scans concurrently and saves the information to the database backend as needed. The program uses multiple processes which communicate with the main dispatch process and a monitoring process to manage the queue of domains for scanning. Our solution performed over 100 concurrent TLS configuration scans simultaneously, with an average scan length of just 1 second.

1) Architecture: The custom SSL/TLS scanning tools architecture is designed to concurrently scan multiple domains using the SSLyze Python API. The tool's architecture is based on the Producer-Consumer pattern for Python multiprocessing. The Producer-Consumer pattern involves three main threads: the producer, consumer, and monitoring threads, each with a specific function.

1) **Producer Thread**: This thread supplies the Consumer threads with domains to process. It retrieves a set batch of domains from the MongoDB database where the SSL/TLS scans have not been completed. These domains are then added to the domain queue, accessible to each Consumer thread. The number of domains retrieved in each batch is customisable and larger, dependent on the read performance of the database.
2) **Consumer Thread**: These threads are the core scanning functionality of the tool. They run in parallel, independently of the Producer thread. Each consumer thread continuously retrieves and scans a domain from the shared queue. The tool uses SSLyze's Python API to scan each domain's SSL/TLS configuration information. The scan results are then processed and saved to the database. The Consumer threads also track their statistics, including the number of domains scanned and skipped and the average time taken per scan. These statistics are updated in a shared progress dictionary, which all Consumer threads can access.
3) **Monitoring Thread**: This separate process is responsible for monitoring and reporting on the progress of the current scanning operations. It periodically updates and displays the number of domains scanned, domains skipped, and overall progress towards completing all scans. The process also calculates the average time taken per scan and uses this data to estimate the time remaining for all scans to complete. This monitoring process is needed to understand the current state of scanning progress.

The architecture leverages parallel processing to scan multiple domains concurrently. The process of scanning a batch of domains involves five steps, each in parallel:

1) The Producer thread retrieves a batch of domains from the MongoDB database for scanning.
2) Each Consumer thread scans a domain for SSL/TLS configurations using SSLyze.
3) The scan results are processed and saved to the database.
4) Progress information is updated in the shared progress dictionary.
5) The monitoring thread displays the progress statistics.

This architecture allows over 100 concurrent SSLyze scans to be performed independently, with limited performance overhead. This extension of SSLyze allows for the large .nz top-level domain dataset to be scanned within the limited time constraints of this study, as set out in the requirements. Figure 7 shows the functional Unified Modeling Language (UML) diagram of the scanning tools system architecture:

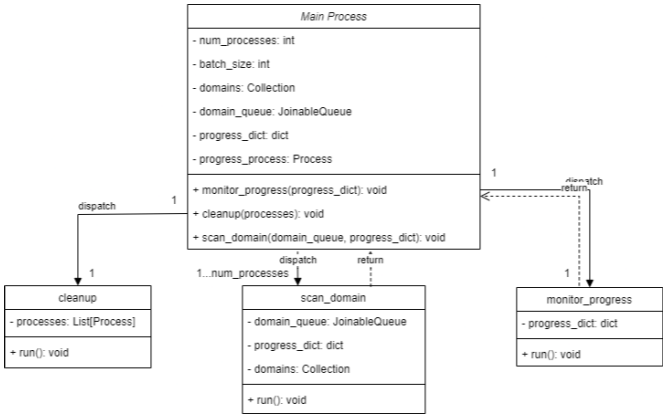

Fig. 7. SSL/TLS Scanning Tool System Architecture.

Figure 8 demonstrates the retrieval, processing, and storage of survey data with the scanner process.

2) Tools, Libraries, And Languages: The development of the custom SSL/TLS scanning tool leveraged several libraries and frameworks to implement its functionality.

1) **Python**: Python is the programming language used to develop the SSL/TLS scanning tools. Python has many built-in and third-party libraries for web scanning, database interactions, and parallel processing.
2) **SSLyze**: SSLyze is the open-source Python library that forms the core of the SSL/TLS scanning tool. It provides an extensive API with the ability to collect SSL/TLS configuration information such as SSL/TLS versions, cypher suites, certificate chains, and vulnerability assessments.

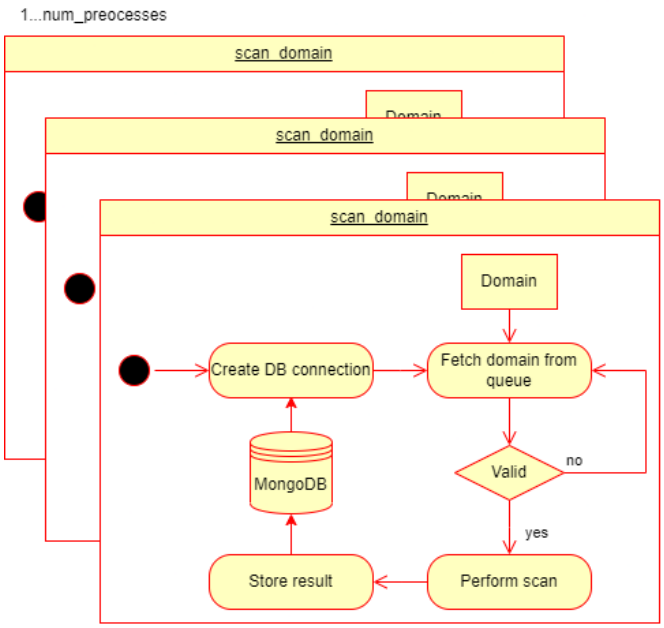

Fig. 8. Scanner Process Architecture.

3) **MongoDB**: MongoDB is the Database Management System (DBMS) of choice for storing domain information and scan results. Its flexibility, scalability, and ability to handle JavaScript Object Notation (JSON)-like documents make it suitable for efficiently storing SSL/TLS configuration data. MongoDB's integration with Python via the PyMongo library simplifies the database operations, allowing for easy programmatic integration.
4) **Multiprocessing**: The tool utilises Python's built-in multiprocessing capabilities to achieve parallelism during scanning. The multiprocessing library allows for multiple processes to run concurrently, enabling the simultaneous scanning of SSL/TLS configurations for multiple domains. This parallelism is crucial in improving the tool's speed and efficiency for the large dataset.
5) **json**: JSON is a lightweight data-interchange format for defining structured data. Python's built-in JSON library is used for parsing and manipulating JSON formatted data used by SSLyze and MongoDB through the Binary Javascript Object Notation (BSON) specification.

B. HTTPS Probing

The custom tool developed for HTTPS probing was created in Go due to Go's built-in concurrency system. The tool can scan hundreds of domains per second, completing the entire dataset of domains in hours. The tool consists of two distinct checks. The first determines whether a web server will allow HTTP traffic, while the second evaluates whether a server automatically upgrades HTTP connections to HTTPS. This is done by impersonating a standard web browser and requesting a web page over HTTP. The tool then analyses the web server's response, determining if the connection was blocked, allowed, or requested to be upgraded to the more secure HTTPS. From this data, we can conclude which websites allow insecure HTTP connections and which websites will attempt to upgrade HTTP connections automatically.

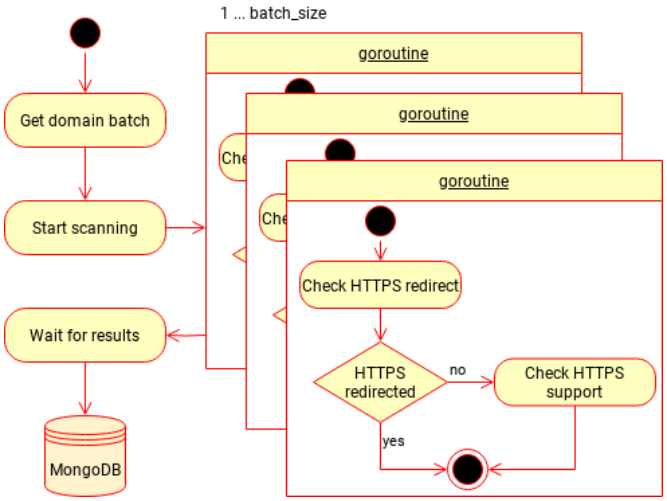

Fig. 9. HTTPS Probe System Architecture.

1) Architecture: The custom HTTPS probing tool is responsible for two distinct checks, each serving a specific purpose:

1) **HTTP Allowance**: The first check assesses whether a web server permits the usage of HTTP traffic. The tool initiates an HTTP GET request to the servers Uniform Resource Locator (URL). The request simulates the behaviour of a standard web browser, including headers such as User-Agent and Accept, that mimic typical browser user agents. Upon receiving the server's response, the tool determines if the server allowed the HTTP connection or disallowed an HTTPS connection. This is done by analysing the returned HTTP status code. HTTP status code 200 Okay indicates that the server allowed the connection.
2) **HTTP to HTTPS Upgrade**: The second component evaluates whether a web server automatically upgrades HTTP connections to HTTPS. Like the first check, the tool sends an HTTP GET request to the servers URL. Upon receiving the server's response, the tool determines if the server redirected the request to HTTPS or requested the connection be upgraded to HTTPS. By following the redirect chain, the tool can determine if the final redirection leads to an HTTPS URL, implying an attempt to automatically upgrade HTTP to HTTPS. The tool otherwise checks if the server's response returned HTTP status codes that indicate redirection. For example, HTTP status code 302 Found could be returned, along with a populated Location header, to inform the browser to use an HTTPS redirect specified in the Location header.

The custom HTTPS probing tool utilises Go's built-in concurrency features to probe multiple domains in parallel. Go's concurrency model is based on lightweight threads called goroutines, which probe each domain concurrently. This pattern allows for the concurrent processing of upwards of

1,000 domains, with the main limitation found to be DNS rate limiting. The process of probing HTTPS usage for a domain involves four steps, repeated for each batch:

1) A batch of domains is retrieved from the MongoDB database.
2) Each domain is probed for HTTPS and HTTP to HTTPS upgrade support in a separate goroutine.
3) The main process waits for all goroutines to finish execution, with a max request timeout of five seconds.
4) The full batch of results is saved to the database.

2) Tools, Libraries, And Languages: Implementing the HTTPS probing tool relies on several libraries and packages in the Go programming language.

1) **net/http** The net/http package is a built-in Go package which is essential for marking HTTP and HTTPS requests to the web servers. The package is used to create numerous individual HTTPS client instances for checking HTTP and HTTPS support for domains.
2) **go.mongodb.org/mongo-driver** The go.mongodb.org/mongo-driver package provides Go clients for connecting to the MongoDB database backend. This package is used for retrieving domains to scan from the database and updating the database with scan results.
3) **sync** The sync package is a built-in Go package which offers synchronisation primitives for the coordination of goroutines. The tool used the sync packages WaitGroup functionality to wait for each batch of goroutines to finish before resuming the main process.
4) **BSON** The BSON package is part of the mongo-driver group and is responsible for encoding and decoding BSON data used by the MongoDB database. This library is essential for parsing the BSON data retrieved from the database and storing the scan results back in a structured format.

These libraries collectively provide the necessary functionality for retrieving domain data, conducting HTTP probing, and updating the MongoDB database with the results. By leveraging these libraries, the tool streamlines the process of scanning a large dataset of domains and efficiently determining their HTTP and HTTPS support status.

C. Data Storage

For data storage, we chose to use a MongoDB database backend. MongoDB's NoSQL capabilities made it an ideal choice for the project's needs due to its flexibility and scalability. MongoDB has libraries for Python and Golang, the two main scanning tools, allowing easy programmatic integration. MongoDB's indexing and querying capabilities also allowed for the efficient retrieval and analysis of the collected data.

MongoDB's indexing system was critical in optimising data retrieval and analysis. By creating indexes on fields used for querying and filtering, we enhanced the performance of complex operations significantly. For example, we created indexes for fields such as the website's domain name, which substantially increased the performance of a single document lookup query. This was crucial for locating and updating existing records with new scanning information from different sources without causing a bottleneck. This indexing strategy significantly reduced the time required to access and manipulate the data.

The approach to database design was based on the interactive data collection process. We initialised the database with a core structure consisting of a document (record) for each domain. As the scans from various tools progressed and more data was gathered, we incrementally added information to each domain's document. This allowed for multiple tools to concurrently collect data and update the central database independently and without conflict. Each tool populated a status field when interacting with the database, marking the state of each scan. This status field played a pivotal role in ensuring data integrity.

Using this technique, we ensure that domains were scanned only once, and the tool could retry the scan if a scan fails. This approach helped to prevent unnecessary disruptions or intrusions on the websites being scanned by minimising the need for repeated scans. By maintaining a structured and organised database schema and applying efficient indexing, we handled large-scale data collection while maintaining data integrity and optimising retrieval for analysis and reporting.

D. Data Collected

The survey included many different data points to assess the security posture of the .nz top-level domain comprehensively. These data points were pivotal in understanding these websites' security configurations and vulnerabilities. The key elements collected included:

1) **Certificates**: Detailed information about the SSL/TLS certificates used by each website. This includes the full certificate chain, expiration dates, issuers, and subjects.
2) **SSL/TLS Versions**: Identification of the SSL/TLS versions supported by each website. This includes versions accepted and rejected by the website.
3) **Cypher suites**: Information about the cypher suites supported by each website for each SSL/TLS, including accepted and rejected cypher suites. Includes complete lists of accepted and rejected cypher suites across all SSL/TLS versions.
4) **Elliptic Curves**: Data regarding the elliptic curves used for encryption, including supports and rejected elliptic curves.
5) **TLS Compression**: Identification on whether TLS data compression was enabled on each web server.
6) **Session Resumption**: Data on session resumption capabilities, which allow for the reestablishment of secure connections. This includes full and partial support.

In addition to this, we also probed websites for various aspects related to their HTTP and HTTPS configurations:

1) **HTTP Usage**: Identification of whether a web server allowed HTTP traffic. This means an HTTP connection was accepted and not upgraded to HTTPS.
2) **HTTP Headers**: Collection of HTTP headers returned in a servers response to a HTTP probe.

3) **HTTPS Redirect Support**: Analysis of whether a website automatically redirected HTTP traffic to HTTPS via upgrade, redirect, or session renegotiation.

Finally, as part of the security assessment, we also scanned for specific vulnerabilities present in SSL/TLS implementations. This includes:

1) **Heartbleed**: Detection of the Heartbleed vulnerability, a critical security bug that could lead to data leakage from a web server's memory.
2) **ROBOT**: Detection of the ROBOT vulnerability, a security issue related to RSA encryption.
3) **OpenSSL CCS Injection**: Detection of potential vulnerabilities related to OpenSSL CCS Injection in the OpenSSL library commonly used in SSL/TLS implementations.
4) **Session Renegotiation DOS**: Detection of vulnerabilities associated with session renegotiation.

E. Infrastructure

In the early stages of implementation, we found that consumer hardware was struggling to meet the intensive processing and networking requirements of numerous concurrent scans. Therefore, infrastructure was designed and established to support data storage and scanning solutions. To address our large-scale data storage needs, we deployed a MongoDB Server instance on Azure's "always free" services [28]. Azure offers a highly scalable and reliable cloud environment with abundant network bandwidth. It is well-suited for handling the substantial volume of data involved in the SSL/TLS scanning and subsequent data storage processes. In addition to the MongoDB Server instance, a Virtual Machine (VM) was deployed to provide the computational resources to run large numbers of concurrent SSL/TLS scans across the entire dataset.

Our infrastructure's network security and efficiency were pivotal in ensuring the speed of security scans while meeting ethical standards. All infrastructure was established within private Virtual Network (VNET)s with strict firewall rules. These rules were configured to grant exclusive access to the MongoDB database from the scanner server over the Azure backbone while blocking unauthorised external access. The scanner process requires open internet access for conducting scans, while the MongoDB database requires no external network access. Moreover, additional security measures were put in place to fortify the overall security of the database. Secure Shell (SSH) certificates were used to control and authenticate access to the MongoDB database. We further mitigated common risks by running externally exposed services on non-default, ensuring these services were not easily discoverable to malicious probing.

The decision to utilise Azure as our infrastructure host was derived from its exceptional scalability and performance attributes. Azure's extensive network infrastructure ensured low latency and high bandwidth connectivity, crucial for large-scale scans. The utilisation of VMs for SSL/TLS scanning allowed us to seamlessly scale the computational resources available in response to our scanning requirements. Figure 10 illustrates the design of the infrastructure.

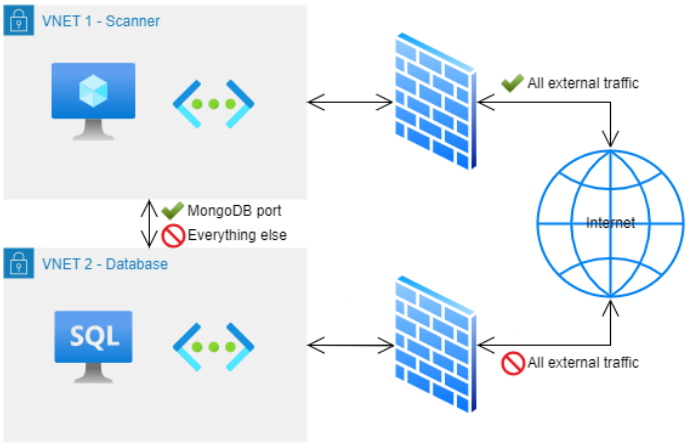

Fig. 10. Cloud Infrastructure Design.

F. Data Analysis

We aggregated and refined the collected dataset through a systematic data analysis approach to derive meaningful insights from the collected data. One of the fundamental steps in our data analysis process involves the utilisation of MongoDB aggregation pipelines. MongoDB offers powerful data aggregation capabilities through its flexible pipeline implementation. Pipelines can filter, transform, and restructure raw data programmatically. We utilised MongoDB pipelines to extract valuable information from the raw data to perform our analysis.

The aggregation and refinement process involved grouping data based on various attributes, such as SSL/TLS protocol versions, cypher suites, and vulnerabilities. By employing pipeline stages such as '$match', '$group', and '$project', we created refined datasets that served as the foundation to guide our analysis. Furthermore, we refactored and restructured the document tree of each domain record to increase the conciseness, clarity, and query speed for database lookups.

Data quality is another crucial aspect of our analysis, as guided by the requirements of this study. Our data underwent a meticulous data cleanup process to ensure the accuracy and integrity of our findings. This process involved identifying missing or incomplete data occurrences and recollecting the erroneous data where necessary. MongoDB pipelines were used to find incomplete or inconsistent data points, further investigated by hand.

V. Results and Evaluation

A. Performance Metrics

1) SSL/TLS Scanning: The TLS scans provided valuable insights into the SSL/TLS configurations of the .nz websites while conforming to the efficiency and reliability requirements of the project. The TLS scans were conducted with a concurrency of 100 scans, resulting in an average scan time of just 1.6 seconds. Figure 11 shows the number of scans completed over time for the default serial SSLyze solution vs our custom extension running in parallel. Our approach of extending SSLyze to implement concurrent scanning allowed for the scanning of a significant number of websites within a

reasonable time frame. This concurrent solution reduced the scan time significantly compared to the base scan time of 10.2 seconds on average for a single SSLyze scan, as shown in Figure 12.

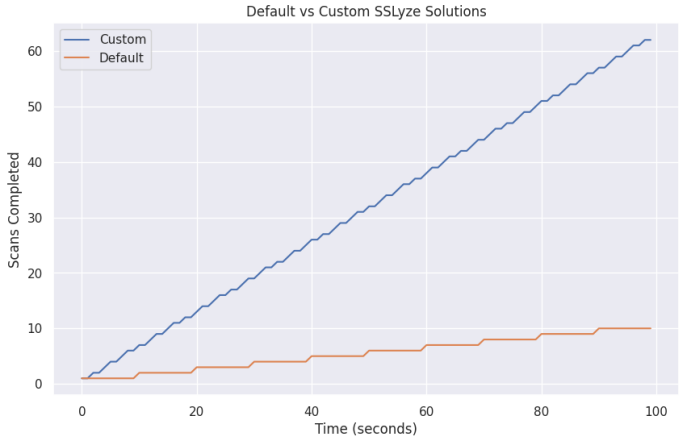

Fig. 11. TLS Scanning Performance Default Vs Custom.

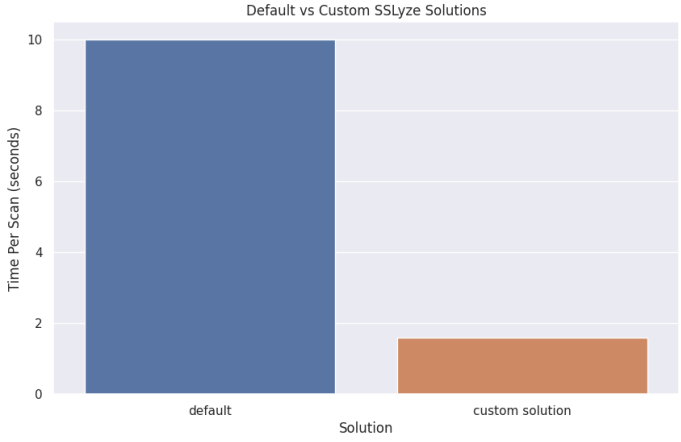

Fig. 12. TLS Time Per Scan Default Vs Custom.

2) HTTPS Probing: The custom tool developed in Go for probing HTTP/HTTPS usage data demonstrated impressive performance compared to alternative solutions. Our custom tool could collect the HTTP usage information for 50 web servers per second, with an average execution time of 20 milliseconds per scan, running 100 concurrent scans. This significantly reduced the time required for HTTP probing and showcased our custom's efficiency in real-world testing scenarios. The tool performs a minimum of two HTTP/HTTPS requests, which take on average between 0.1 and 1 second to perform per request. Our custom tool takes, on average, 400 milliseconds per scan to complete, 200 milliseconds per request. This is well within the window of an acceptable HTTPS response time, indicating that the tool adds little overhead compared to an ordinary HTTP request.

3) Network Performance and Ethical Considerations: The network performance of the scanning tools is a critical aspect that directly impacts the efficiency and scalability of our study. Figure 13 and Figure 14 show the network performance of

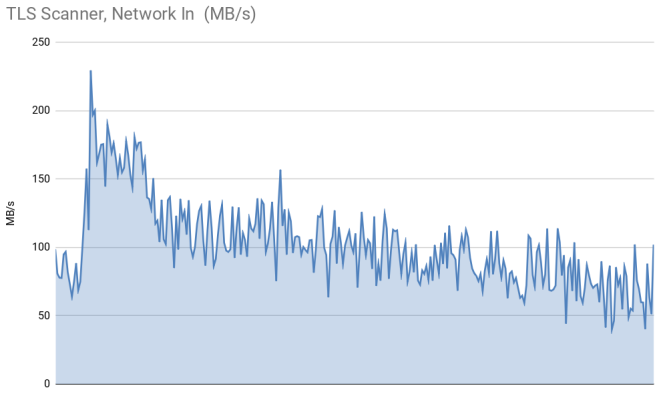

Fig. 13. TLS Scanner, Network In (MB/s).

the SSL/TLS scanning processes. These statistics indicate that the tools reached a maximum network throughput of 200 Megabyte (MB)/s when scanning 100 domains concurrently, translating to an average network load of 2 MB/s per domain. This network utilisation level demonstrates our approach's scalability, as we can scan many domains in parallel. This aligns with the core requirement of conducting the scans within the time frame allocated for this study. The ability to scan multiple domains concurrently while maintaining a reasonable network throughput ensures that no disruption is caused to the websites being scanned.

Ethical considerations are of great concern when conducting large-scale web security scans. The network performance statistics show that our scans operated within an acceptable level of network throughput, preventing any excessive network activity that could potentially disrupt the surveyed websites. 2 MB/s is well within normal bandwidth usage for a user requesting a medium to large sized web page from a web server. It is crucial to maintain a balance between data collection efficiency and ensuring sustainable, ethical data collection practices. These results demonstrate that this balance was achieved.

Our scanning methodology consistently adhered to ethical standards by only scanning publicly accessible information. Our scans did not involve intrusive or harmful actions, demonstrating our commitment to ethical security scanning. Vulnerability probing was conducted by probing web servers for externally facing information available to any client. Furthermore, the study's premise justifies the nature and scale of our scanning practices, as it aims to enhance web security by identifying and addressing potential vulnerabilities. In this way, our study actively contributes to improving online security practices, ultimately benefiting both website operators and their users.

B. Survey Findings

1) SSL/TLS Configurations:

1) **Strong Adoption of Modern Encryption**: Figure 15 shows that a significant number of websites, around 90,000 or just under 50%, have adopted a minimum TLS

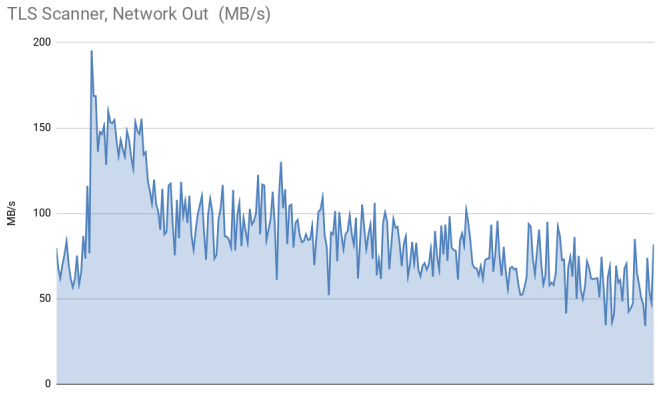

Fig. 14. TLS Scanner, Network Out (MB/s).

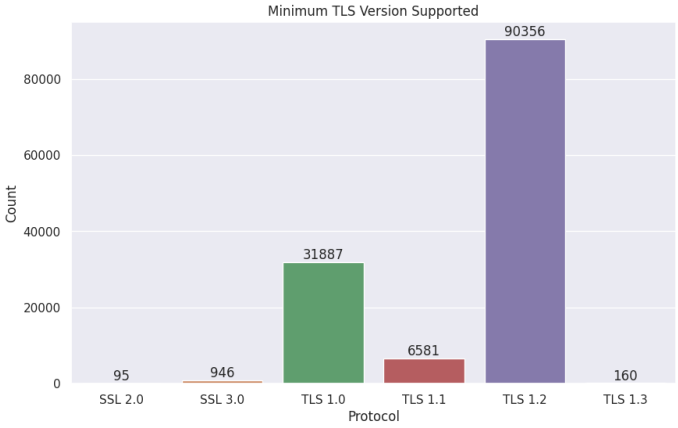

Fig. 15. Minimum SSL/TLS Version Supported.

version of 1.2. This means that older and less secure TLS and SSL protocols can not be used to encrypt connections to these sites. This indicates a commendable effort by .nz websites to uphold secure encryption standards. Users of these websites can reasonably trust the confidentially and integrity of their data during transmission when visiting any of these 90,000 websites.

2) **Insecure Protocols Remain**: Approximately 32,000 websites still continue to support the outdated TLS 1.0 protocol. Interestingly, a lower 6,500 websites still support TLS 1.1, a newer version of TLS over TLS 1.0. This indicates an interesting trend that seems to suggest that websites that use outdated SSL/TLS versions are intentionally continuing to support these protocols. It is important to note that TLS 1.1 is in the process of being actively deprecated on major operating systems such as Microsoft's Windows 11 as of September 2023, making its usage incredibly problematic from a security standpoint. Furthermore, just over 1,000 websites still support SSL protocols, which have been deprecated for several years. The continued use of these insecure protocols is concerning as it exposes these websites and their users to unnecessary risks and vulnerabilities.
3) **Coexistence of Secure and Insecure Protocols**: Figure 16 shows the preferred SSL/TLS versions supported for each website. What is interesting is that some websites which support modern TLS versions, such as TLS 1.2 and 1.3, continue to maintain support for outdated and insecure TLS versions (TLS 1.0 and 1.1). This is likely done to maintain compatibility with older devices and browsers, which may not support newer protocols. These websites allow these insecure protocols to make themselves accessible to as many people as possible. This begs whether this is an acceptable trade-off between compatibility and security.

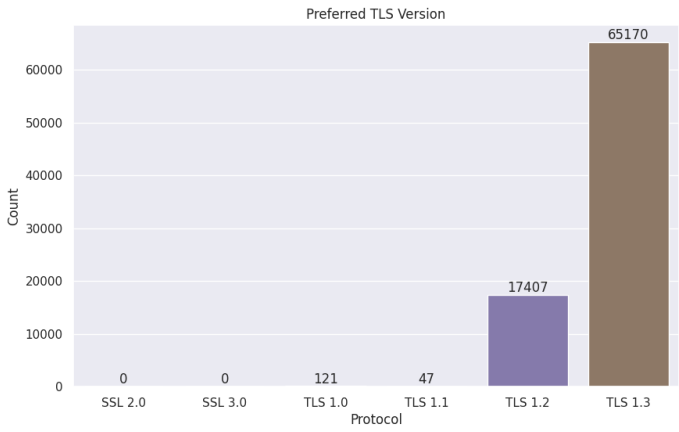

Fig. 16. Maximum SSL/TLS Version Supported.

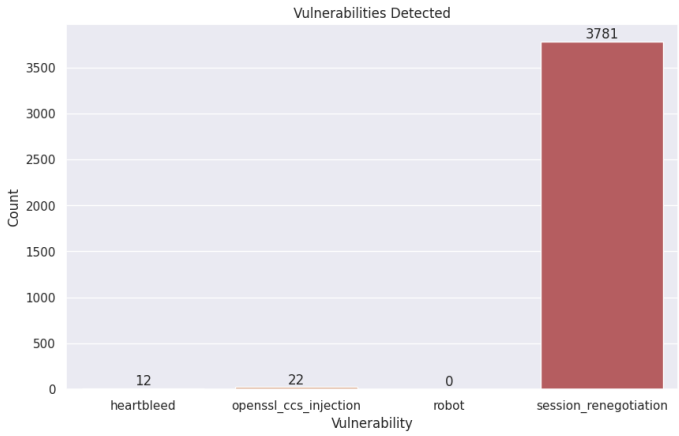

Fig. 17. Vulnerabilities Detected.

2) Vulnerabilities:

1) **Session Renegotiation DOS**: Figure 17 shows that our scans revealed approximately 4,000 websites which are vulnerable to Session Renegotiation DOS attacks. This attack can disrupt websites by overwhelming their capacity to handle multiple requests simultaneously. Session Renegotiation is renegotiating the current authentication data of a TLS connection instead of starting a new one. This could happen, for example, if a user was visiting an online shopping store without an account and then chose to log in to buy something. Session Renegotiation DOS is a known vulnerability and weakness of

TLS 1.2, which has been fixed in the newest TLS 1.3. The vulnerability abuses that Session Renegotiation is far more computationally expensive for the server than the client (15x more). To overwhelm a website, clients can send many Session Renegotiation requests, overwhelming the server and potentially causing disruption. To mitigate this risk, these vulnerable websites should upgrade to TLS 1.3.

2) **OpenSSL CCS Injections**: We discovered 22 instances of OpenSSL CCS Injection vulnerabilities across the dataset. OpenSSL CCS Injection is a vulnerability that affects specific versions of OpenSSL, a library used in TLS implementations, that can be exploited to perform a man-in-the-middle (MITM) attack. A MITM attack allows attackers to decrypt, view, and modify any transmissions sent between the client and the server. Any clients with a vulnerable version of OpenSSL would be susceptible to this attack when visiting any of these 22 websites. This vulnerability could be fixed if these websites were to update the version of OpenSSL used in their TLS implementations.
3) **Heartbleed**: Our survey also uncovered 12 instances of the Heartbleed vulnerability. Heartbleed is a famous security vulnerability stemming from a bug in the OpenSSL library that can be abused to allow attackers to read sensitive data from a server's memory. The bug is located in the TLS heartbeat extension and was present in many previous versions of OpenSSL until its discovery in 2012. As before, affected websites are urged to update the OpenSSL library versions they are using on their web servers.

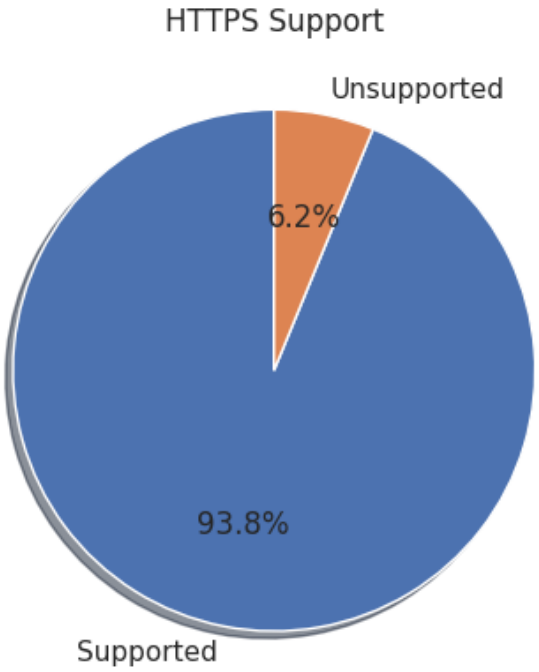

Fig. 18. HTTPS Support.

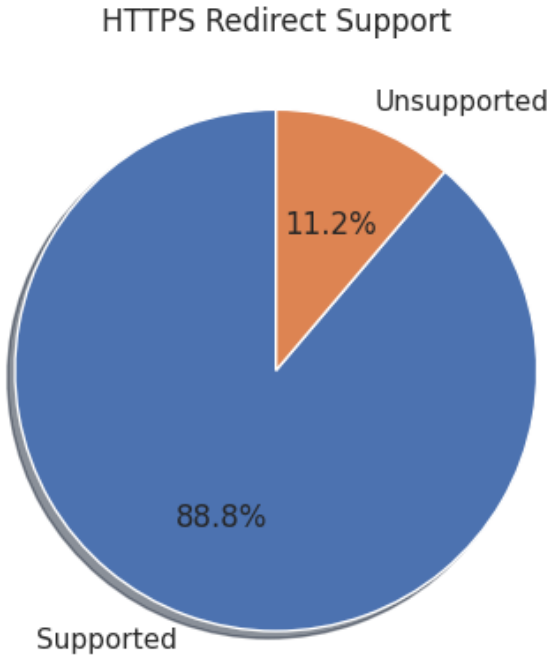

Fig. 19. HTTPS Redirect Support.

3) HTTP/HTTPS Usage:

1) **Lingering HTTPS Adoption**: As per Figure 18, alarmingly 6.6% of websites still do not support secure HTTPS. This means some websites still lack the basic encryption for securing data transmission. Communication conducted over HTTP is completely unencrypted and does not use]SSL or TLS, sending data entirely in plane text. This exposes users' data to interception by malicious actors, posing serious security concerns.
2) **Absence of Secure Redirection**: Our survey also found that around 11.2% of websites do not support automatic redirection from HTTP connections to HTTPS 19. This refers to upgrading insecure HTTP connections to secured HTTPS connections wherever possible. Secure redirection is fundamental to web security as it prevents users from using insecure HTTP connections when accessing websites. The absence of this feature essentially doubles the number of websites that could be accessed using completely unsecured connections. These findings highlight a lack of HTTPS enforcement and proper redirection practices within the .nz domain. This calls for immediate action to address these vulnerabilities, upgrade protocols, and ensure web security best practices to protect users' data.

4) Certificate Lifespans:

1) **Certificates Have Short Lifespans**: Figure 20 shows a prominent trend in the average lifespan of certificates. Our survey shows that approximately 64,000 certificates were identified as having a 90-day lifespan. This lifespan of a certificate indicates how long the certificate is valid before needing to be renewed. This indicates a trend

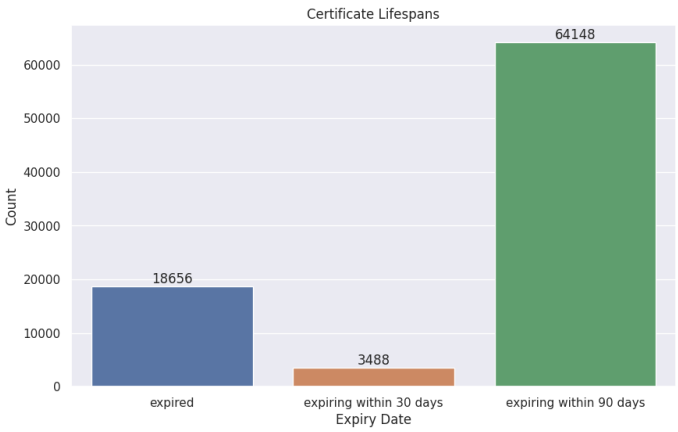

Fig. 20. Certificate Lifespans.

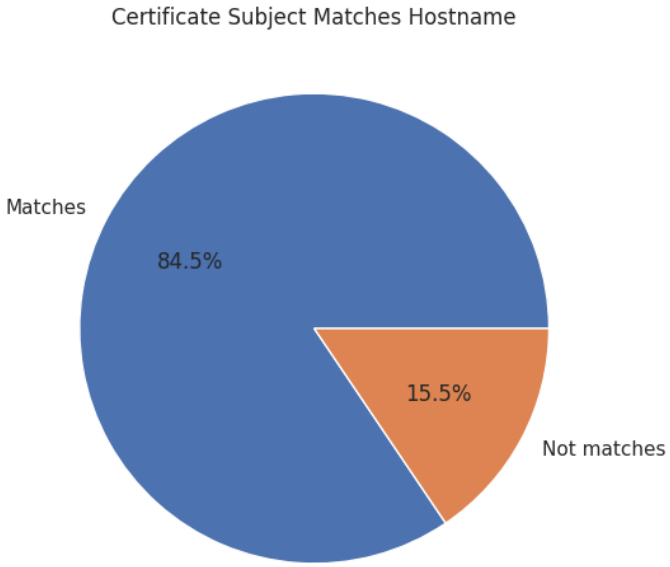

Fig. 21. Certificate Subject Matches Hostname.

in web certificate issuers opting for shorter certificate lifespans, which could be done to encourage the adoption of automated certificate renewal tools. Interestingly, the survey also found approximately 19,000 websites with expired certificates. Expired certificates can lead to significant security issues, as without a certificate, users can not validate the web server's identity.

2) **Mismatched Host Names**: Alarmingly, our investigation also found that approximately 15.45% of certificates had subject names which failed to match the name of the domain presenting the certificate. The subject name on a certificate refers to the object the certificate is assigned to, in this case, a website URL. A mismatch between the websites URL and its certificate subject indicates the website is using an incorrect or misconfigured certificate or could even be attempting to impersonate another website.

C. Comparison

When comparing these results to F5 Lab's findings, our study is more recent and tailored to the .nz domain. F5 Labs revealed that 63% of websites designated TLS 1.3 as the preferred protocol [8]. In contrast, our results show that only 42.48% of websites in the .nz top-level domain prefer TLS 1.3. This is a similar result to the SSL Pulse report, which found that 64.8% of the top 135,000 most popular websites preferred TLS 1.3 [11]. This strongly suggestions that TLS 1.3 adoption in the .nz domain lags behind the global web landscape. However, it is important to note that the F5 Labs study focused on the top one million websites, which may be biased towards better security due to their popularity.

In addition, F5 Labs found that 0.4% of websites still preferred TLS 1.0 and 0.002% still preferred SSL 3.0 [8]. In contrast, our results show that an incredibly low 0.07% of websites in the .nz domain prefer TLS 1.0 and no websites were found to prefer SSL 3.0. This demonstrates a positive trend in the .nz domain, with a significantly lower presence of severely deprecated and insecure SSL/TLS versions compared to the broader web landscape. These results indicate that .nz websites are more willing to discontinue using severely outdated SSL/TLS protocols but less ready to upgrade to the latest and most secure TLS 1.3.

Similar to the SSL Pulse report, which reported 99.9% of websites support at least TLS 1.2, our results show that 99.8% of .nz websites also support at least TLS 1.2 [11]. The SSL Pulse report also highlighted the presence of websites supporting outdated TLS versions, including 32.5% supporting TLS 1.1 and 30.1% supporting TLS 1.0. Our results show that 28.5% of .nz websites support TLS 1.1 and 23.62% still support TLS 1.0. This indicates that .nz websites, on average, support less deprecated SSL/TLS versions than the most popular websites. SSL Pulse is a monthly survey that accurately reflects the current state of the web with a comparable time frame to our investigation.

Regarding vulnerabilities, F5 Labs identified that 2.8% of websites were vulnerable to TLS 1,2 session renegotiation DOS. Our results show that a comparable 2.5% of .nz websites are vulnerable to the same session renegotiation vulnerability. On certificates, our results agree with those of F5 Labs, indicating that most certificates for .nz domains also have a 90-day expiry period. Certificates are generally assigned by a CA, so changes to CA policies would impact all regions equally.

VI. Conclusions and Future Work

The extensive survey of SSL/TLS and HTTPS configurations in the .nz domain has provided valuable insights into the state of web security and encryption practices within New Zealand. Our findings have important implications and demonstrate clear opportunities for improvement within the .nz domain. Based on our results, we recommend the following best practices for enhancing web security within the .nz domain.

Website administrators are encouraged to upgrade their web servers to modern and secure TLS protocols, with a minimum

of TLS 1.2 supported. Support for protocols older than TLS 1.2 should be phased out, and SSL should not be supported at any version. Discontinuing support for these older and insecure protocols is critical to ensure the safety of both the web servers and their users.

Adopting certificates with shorter 90-day lifespans is a positive trend, encouraging automatic certificate renewal, which prevents certificates from expiring unintentionally. Website administrators are encouraged to remain diligent and employ automated certificate renewal processes to minimise the impact of expiring certificates.

All websites should prioritise the adoption of HTTPS to ensure secure data transmission. HTTPS is a fundamental security feature for data security and integrity. Website owners are strongly encouraged to obtain and install valid SSL/TLS certificates to enable secure HTTPS communication. Furthermore, the absence of secure HTTPS redirection should be addressed, as it leaves users vulnerable to accessing websites via HTTP connections. Implementing secure HTTPS redirection ensures traffic is automatically directed to HTTPS connections, enhancing security with limited implementation complexity.

Website administrators should actively monitor for and address vulnerabilities such as Heartbleed, Session Renegotiation DOS, and OpenSSL CCS Injection. Upgrading to secure protocols and maintaining up-to-date security libraries and dependencies is the simplest way to mitigate these vulnerabilities. These vulnerabilities have been well-known to the public for years and should be addressed soon. The steps for prevention and mitigation of each vulnerability identified have been extensively studied and should be implemented. We recommend that all organisations or individuals with the resources implement continuous security monitoring for the web applications. Regular security scans and assessments can help maintain and improve security standards, finding vulnerabilities early. Organisations, individuals, and other entities should follow the best practices, standards, and guidelines for web security. Initiatives aimed at educating website owners and administrators about the importance of these best practices are essential.

The findings of our study serve as a reference point for web security within New Zealand, against which future research can be measured. Our survey tools and methodology can be extended to assess security within other domains, such as websites in the Australian top-level domain. Our study extends beyond the .nz domain, providing insights that can benefit web security standards on a global scale. Future work may involve extending our tools and methodology to monitor and assess web security on an ongoing basis, providing valuable data to the broader cybersecurity community.

Our study contributes valuable insights into the web security practices within the .nz top-level domain. It highlights the progress made in adopting secure protocols while identifying areas for improvement that require attention. The ongoing commitment to security, education, and increased awareness are essential for maintaining a safe and secure web landscape within New Zealand.

REFERENCES

[1] R. Oppliger, *SSL and TLS: Theory and practice, Second edition*. Artech House, 2016.

[2] K. Moriarty and S. Farrell, "RFC 8996: Deprecating TLS 1.0 and TLS 1.1." https://datatracker.ietf.org/doc/rfc8996/. accessed 11-10-2023.

[3] C. Adams and S. Lloyd, *Understanding PKI: Concepts, standards, and deployment considerations*. Addison-Wesley, 2010.

[4] R. E. Klima and N. Sigmon, *Cryptology: Classical and modern*. Chapman &; Hall/CRC, 2018.

[5] C. Wong, *HTTP pocket reference*. O'Reilly, 2000.

[6] D. Warburton, "The 2021 TLS telemetry report." https://www.f5.com/labs/articles/threat-intelligence/the-2021-tls-telemetry-report.

[7] V. L. Pochat, "A Research-Oriented Top Sites Ranking Hardened Against Manipulation." https://tranco-list.eu/. accessed 11-10-2023.

[8] D. Warburton, "Introducing the cryptonice HTTPS scanner." https://www.f5.com/labs/articles/threat-intelligence/cryptonice. accessed 11-10-2023.

[9] B. Lokhande, "SSL Labs Grading Update: Forward Secrecy, Authenticated Encryption and ROBOT." https://blog.qualys.com/product-tech/2018/02/02/forward-secrecy-authenticated-encryption-and-robot-grading-update. accessed 18-10-2023.

[10] D. Fisher, "CRIME attack uses compression ratio of TLS requests as side channel to Hijack Secure Sessions." https://threatpost.com/crime-attack-uses-compression-ratio-tls-requests-side-channel-hijack-secure-sessions-091312/77006/. accessed 11-10-2023.

[11] Qualys, "Qualys SSL Labs - SSL Pulse." https://www.ssllabs.com/ssl-pulse/. accessed 14-10-2023.

[12] web.archive.org, "The top 500 sites on the web the sites in the top sites lists are ordered by their 1 month Alexa Traffic rank." https://web.archive.org/web/20220101025437/alexa.com/topsites. accessed 18-10-2023.

[13] nabla c0d3, "Nabla-C0D3/sslyze: Fast and powerful SSL/TLS Scanning Library.." https://github.com/nabla-c0d3/sslyze. accessed 20-05-2023.

[14] D. A. Wheeler, "Preventing Heartbleed," *Computer*, vol. 47, no. 8, pp. 80–83, 2014.

[15] H. Böck, *Return Of Bleichenbacher's Oracle Threat (ROBOT)*. 27th USENIX Security Symposium, 2018.

[16] Free Software Foundation, "GNU Affero General Public License Version 3 (AGPL-3.0)." accessed 30-05-2023.

[17] Q. Nguyen, *Mastering concurrency in python: Create faster programs using concurrency, asynchronous, multithreading, and Parallel Programming*. Packt Publishing Ltd, 2018.

[18] Prbinu, "Prbinu/TLS-scan: An internet scale, Blazing Fast SSL/TLS scanner (non-blocking, event-driven)." https://github.com/prbinu/tls-scan. accessed 20-04-2023.

[19] Veracode, "Crashtest Security." https://crashtest-security.com/. accessed 14-10-2023.

[20] "Health Insurance Portability and Accountability Act of 1996 (HIPAA)." https://www.cdc.gov/phlp/publications/topic/hipaa.html. accessed 14-10-2023.

[21] "General Data Protection Regulation (GDPR)." https://gdpr-info.eu/. accessed 14-10-2023.

[22] I. Consulting, "International Organization for Standardization." https://www.iso.org/home.html.

[23] M. Kolybabi, "ssl-enum-ciphers NSE script - Nmap Scripting Engine documentation." https://nmap.org/nsedoc/scripts/ssl-enum-ciphers.html. accessed 16-10-2023.

[24] Qualys, "Qualys SSL Labs - Projects / SSL Server Rating Guide." https://www.ssllabs.com/projects/rating-guide/index.html. accessed 16-10-2023.

[25] Drwetter, "Drwetter/testssl.sh: Testing TLS/SSL encryption anywhere on Any Port." https://github.com/drwetter/testssl.sh. accessed 16-10-2023.

[26] F5-Labs, "F5-Labs/Cryptonice: Cryptonice is both a command line tool and library which provides the ability to scan and report on the configuration of SSL/TLS for your internet or internal facing web services.." https://github.com/F5-Labs/cryptonice. accessed 16-10-2023.

[27] H. C. Rudolph, "TLS ciphersuite search." https://ciphersuite.info/. accessed 16-10-2023.

[28] Microsoft, "Microsoft Azure Free Service." https://azure.microsoft.com/en-us/pricing/free-services. accessed 12-06-2023.