Reverse Proxy: Web Services Exposure and Protection

Tommaso Sommaruga *SUPSI 3rd year DS&AI*

Abstract
The task of this project was to create a protection shield for an exposed web server over HTTPS, using a reverse proxy system with Nginx. The system collects web traffic data and logs it in a server file. A dashboard was created to visualize the traffic statistics, with additional details provided by external IP geolocation and proxy detection services. This report discusses the system architecture, logging mechanisms, and the final dashboard solution.

Contents

1 Introduction **2**
1.1 Project Objectives 2

2 System Architecture **2**
2.1 Reverse Proxy with Nginx . . . 3
2.2 HTTPS Encryption Using win-acme 3
2.3 Log Server 3
2.4 Log Data Analysis Dashboard . 3

3 Traffic Logging **3**
3.1 Log Format Configuration . . . 4
3.2 Log Data Processing 4

4 Dashboard Overview **4**
4.1 Features of the Dashboard . . . 4
4.2 Real-time Data Updates 5
4.3 Interactivity 6
4.4 Tabular Data Visualization . . . 6

5 Geolocation and Proxy Information **6**
5.1 IP Geolocation with IP2Location 6
5.2 Proxy Detection with IP2Proxy 6

6 Challenges Encountered **7**
6.1 Technical Issues 7
6.2 Accessibility Issues 7
6.3 Data Retrieval and Performance Issues 7
6.4 Slow Performance with Large Log Files 8

7 Future Improvements **8**

8 Conclusion **9**

1 Introduction

In today's digital age, exposing a web service to the internet inherently brings security challenges. Web servers, especially those running over HTTP/HTTPS, are continually under attack from various sources such as Distributed Denial of Service (DDoS)[2] attacks, bots, and automated web scraping tools. Protecting these exposed web services is crucial for ensuring their availability and maintaining data integrity. This project addresses these concerns by developing a comprehensive web traffic monitoring and protection system using Nginx[3] as a reverse proxy.

The system is designed to act as a "shield" for the web server, filtering malicious traffic while logging and analyzing legitimate user requests. By leveraging Nginx's reverse proxy[1] features, all incoming traffic is intercepted, logged, and routed securely. This report details the steps involved in implementing the system—from configuring the Nginx reverse proxy and setting up HTTPS encryption to parsing the traffic logs and building a real-time monitoring dashboard. Additionally, open-source geolocation and proxy databases are utilized to enrich the data collected from incoming traffic, providing valuable insights into the origins and nature of each request.

The primary goal of the project is not just to protect the web server, but also to give administrators an intuitive interface to monitor and analyze traffic. The dashboard provides an interactive view of traffic data, enabling quick identification of issues such as potential bots, unauthorized access attempts, or unusual spikes in traffic.

1.1 Project Objectives

The main objectives of this project include:

- **Create a reverse proxy using Nginx:** Set up Nginx to act as a reverse proxy that forwards requests to the backend web server, while logging all incoming traffic.
- **Implement HTTPS encryption:** Ensure secure communication by setting up SSL certificates using the win-acme tool to automatically generate and renew certificates from Let's Encrypt.
- **Log all incoming traffic to a centralized server:** Configure Nginx to capture detailed information about each request, including the IP address, request method, user agent, and other essential data, storing these logs in a centralized file for further analysis.
- **Analyze the collected log data to generate actionable insights:** Process the log data to extract relevant information, such as traffic volume, error rates, and geolocation data, to provide a deeper understanding of web traffic and user behavior.
- **Develop a real-time dashboard for monitoring web traffic and potential security threats:** Create an interactive dashboard that visualizes key metrics, such as the number of requests, top visited URLs, most active IP addresses, and potential security risks like bot activity and failed login attempts.

2 System Architecture

The architecture of the system is built around a reverse proxy[4] setup using Nginx, which handles all incoming requests and logs their details. The web application, hosted on a custom domain, is served through this reverse proxy, ensuring that all traffic is monitored and controlled. This section details the key components that make up the architecture of the system:

2.1 Reverse Proxy with Nginx

Nginx is configured as a reverse proxy, responsible for routing incoming HTTP/HTTPS traffic to the web application. Acting as a middle layer, it ensures that the client interacts with the proxy server instead of the web application directly. Nginx is also configured to log the request data, including details about the IP address, user agent, request method, response status, and more. This enables comprehensive logging for security and traffic analysis purposes.

The reverse proxy setup plays a vital role in protecting the web server by hiding it from direct exposure to the internet. It also helps to offload SSL/TLS encryption work and efficiently manage incoming requests.

2.2 HTTPS Encryption Using win-acme

To ensure the security of the traffic between the client and the server, HTTPS encryption is implemented using SSL certificates generated by the win-acme tool. Win-acme automates the process of obtaining and renewing SSL certificates from Let's Encrypt, which guarantees secure communication. By using these certificates, the system ensures that sensitive data is transmitted over a secure connection, preventing eavesdropping and man-in-the-middle attacks.

2.3 Log Server

All incoming web traffic is logged using a custom log format defined in the Nginx configuration. These logs are stored in a centralized log file for further analysis. The format captures essential information, including the real IP address, request details, status code, and user agent. By logging all incoming traffic, the system enables detailed monitoring and analysis of the website's usage.

2.4 Log Data Analysis Dashboard

The log data is processed and visualized through a dashboard. This dashboard provides real-time insights into the traffic patterns, such as the most visited IPs, URLs, status codes, and more. The dashboard integrates geolocation data and proxy information for detailed traffic analysis. It enables administrators to easily monitor web traffic, spot unusual patterns, and gain valuable insights into how the web application is being used.

3 Traffic Logging

One of the key components of this system is the comprehensive logging mechanism that captures detailed information about all incoming web traffic. The Nginx reverse proxy is configured to log all access requests, ensuring that every interaction with the web server is tracked. These logs provide critical insights for monitoring traffic, analyzing usage patterns, and detecting potential security threats. The following aspects are captured in the logs:

- **IP address**: The real IP address of the client making the request. This is crucial for identifying the origin of the traffic and for geographical or security analysis.

- **Timestamp**: The precise date and time when the request was made, enabling detailed time-based analysis of traffic patterns.

- **Request details**: Information about the HTTP method used (e.g., GET, POST), the requested URL, and the protocol (e.g., HTTP/1.1). This allows for deeper insights into what parts of

the website are being accessed most frequently.

- **Response status**: The HTTP status code returned by the server, such as 200 (OK), 404 (Not Found), or 500 (Internal Server Error). This helps identify any issues with the website or possible attack attempts.

- **User agent**: The browser or tool used by the client to make the request. This is valuable for identifying the type of users accessing the site (e.g., mobile users, desktop users, bots).

- **Referrer**: The URL from which the request originated, providing context on how users are arriving at the website (e.g., directly, via a search engine, or through an external site).

- **X-Forwarded-For**: A header used to capture the originating IP if the request has been forwarded through a proxy. This is especially useful for tracking requests that pass through intermediate proxies, such as when the site is behind a content delivery network (CDN) or other proxy services.

3.1 Log Format Configuration

The Nginx log format is customized to capture these details in a structured way, enabling efficient analysis and data extraction.

This log format ensures that all relevant data is captured consistently, allowing for effective parsing and analysis. The logs are stored in a centralized log file, which is then parsed and processed to extract useful information. This information is later fed into the dashboard for visualization.

3.2 Log Data Processing

To ensure that the log data is meaningful, it is processed and filtered to extract key insights. The log files are parsed in real-time to identify traffic patterns, errors, or suspicious activity. The processed data can include the following key metrics:

- **Traffic volume**: The total number of requests, segmented by time, IP address, or URL.
- **Error analysis**: The distribution of different HTTP status codes, helping to identify common errors such as 404s or 500s.
- **User behavior**: Analysis of user-agent data to understand the types of devices and browsers used to access the site.

4 Dashboard Overview

The dashboard is the central component of this system, allowing users to visualize and monitor web traffic data in real-time. Built with Dash, the dashboard fetches the log data, processes it, and displays various statistics about the traffic.

4.1 Features of the Dashboard

The dashboard provides the following key features:

- **World Map Visualization**: A geolocation heatmap of the IP addresses accessing the web server.

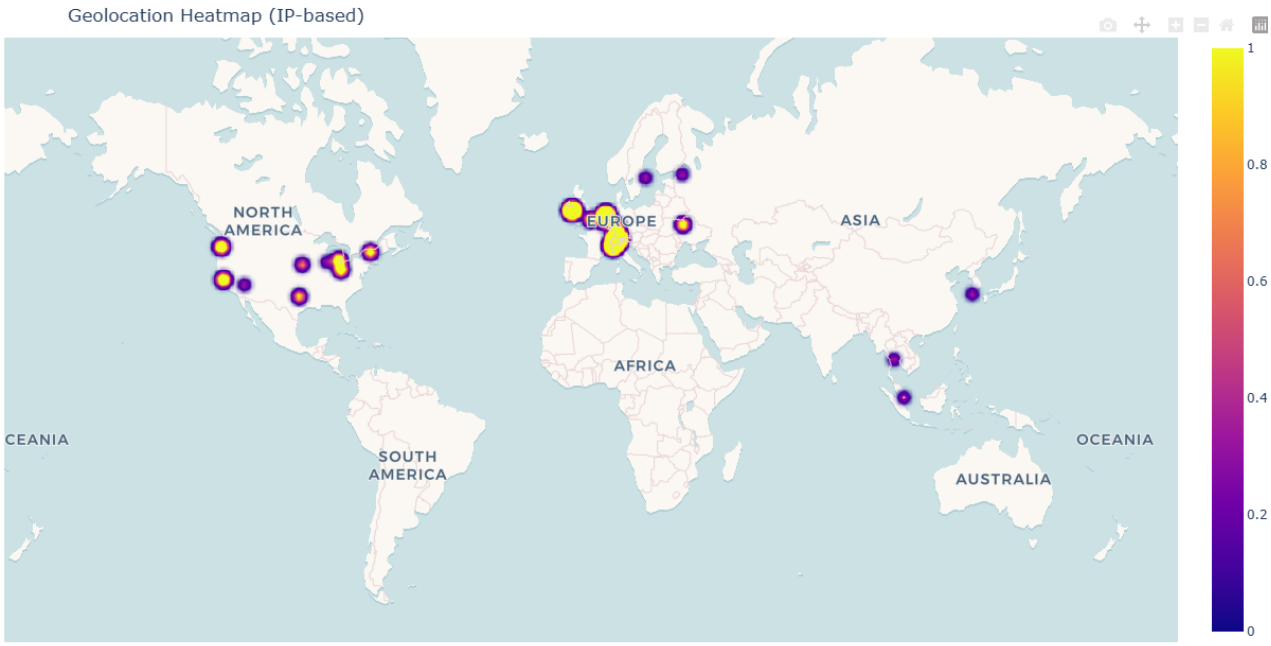

Figure 1: Geolocation heatmap

- **Traffic Over Time**: A bar chart showing the number of requests per time interval.
- **Status Code Distribution**: A pie chart visualizing the distribution of HTTP status codes.
- **Top IPs and Paths**: Bar charts showing the most frequent IP addresses and requested URLs.
- **IP Table**: A detailed table showing information about each IP, including country, user agent, and more.

4.2 Real-time Data Updates

The dashboard updates every 30 seconds using the Dash 'Interval' component, ensuring that the displayed data reflects the most recent web traffic activity.

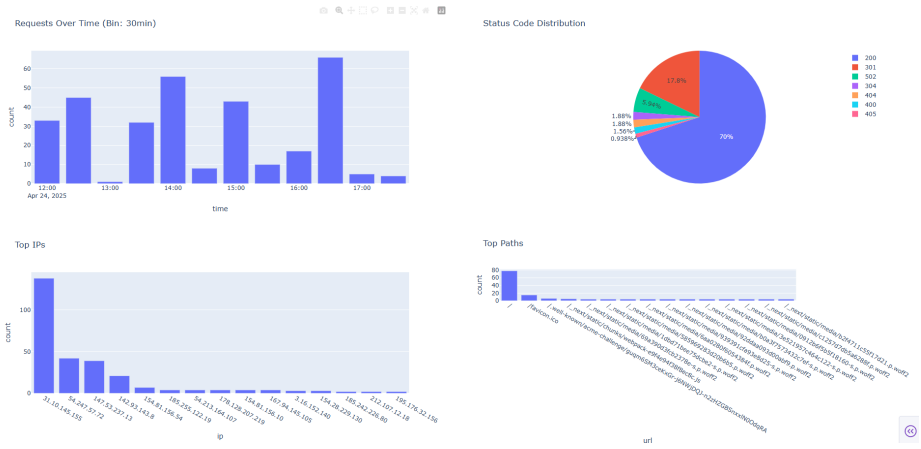

Figure 2: Insights

4.3 Interactivity

Users can filter the data based on various parameters such as IP address, URL, user agent, HTTP method, status code, and country. These filters allow for a detailed view of specific subsets of the traffic.

4.4 Tabular Data Visualization

The dashboard includes a tabular data visualization feature that presents the raw logs in a structured table format. This allows users to easily analyze and filter through the log entries, helping to identify patterns and anomalies in the traffic data.

ip	country	latitude	longitude	user_agent	is_bot	isp	usage_type	count
112.133.30.187	Korea (the Republic of)	33.513191	126.523346	curl/7.88.1	false	-	-	1
137.184.181.41	United States of America	37.354111	-121.955490	Mozilla/5.0 (compatible)	false	-	-	2
142.93.143.8	Netherlands (Kingdom of the)	52.378502	4.899980	-	false	-	-	1
142.93.143.8	Netherlands (Kingdom of the)	52.378502	4.899980	Go-http-client/1.1	false	-	-	18
142.93.143.8	Netherlands (Kingdom of the)	52.378502	4.899980	Mozilla/5.0 (Linux; Android 6.0; HTC One M9 Build/MRA788279) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2469.98 Mobile Safari/537.3	false	-	-	1
142.93.143.8	Netherlands (Kingdom of the)	52.378502	4.899980	Mozilla/5.0 (l9scan/2.0.3313e2733323e23353e2734313; +https://leakix.net)	false	-	-	1
147.53.237.13	Italy	45.737358	7.316600	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36	false	-	-	38
147.53.237.13	Italy	45.737358	7.316600	win-acme/2.2.9.1701 (+https://github.com/win-acme/win-acme)	false	-	-	1
154.28.229.130	United States of America	44.309170	-73.235001	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36	false	-	-	2
154.28.229.130	United States of America	44.309170	-73.235001	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36	false	-	-	1
154.28.229.3	United States of America	44.309170	-73.235001	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36	false	-	-	1
154.81.156.10	Netherlands (Kingdom of the)	52.378502	4.899980	-	false	-	-	4
154.81.156.54	Netherlands (Kingdom of the)	52.378502	4.899980	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36 Edg/90.0.818.46	false	-	-	7
16.16.66.75	Sweden	59.332748	18.064840	Mozilla/5.0 (compatible; Let's Encrypt validation server; +https://www.letsencrypt.org)	true	-	-	1
162.142.125.44	United States of America	42.259865	-83.719894	Mozilla/5.0 (compatible; CensysInspect/1.1; +https://about.censys.io/)	true	-	-	2

Figure 3: Tabular raw logs

5 Geolocation and Proxy Information

In addition to the raw web traffic data, the system enriches the information with geolocation and proxy data. This allows for a deeper analysis of the traffic and provides insights into the geographical locations of visitors and whether they are using proxies or VPNs.

5.1 IP Geolocation with IP2Location

The IP2Location database is used to obtain geolocation information for each IP address. This includes the country, latitude, and longitude of the client. The information is visualized on a world map to show where the requests are coming from.

5.2 Proxy Detection with IP2Proxy

The IP2Proxy database was integrated to detect whether an incoming request is originating from a proxy, VPN, or anonymizing service. However, in addition to proxy detection, the IP2Proxy database also provides valuable information about the **ISP** (Internet Service Provider) and **USAGETYPE** (type of internet usage, such as residential or business). These fields could provide valuable insights into the traffic source and help distinguish between legitimate traffic and suspicious behavior (e.g., bot traffic).

- **ISP**: This field provides information about the Internet Service Provider for each IP address. Analyzing the ISP can help determine the traffic's origin, distinguishing between residential ISPs

and data centers, which may indicate suspicious or automated traffic.

- **USAGETYPE**: This field provides details about the usage type, distinguishing between residential (typically legitimate users) and business (potentially associated with data centers or bot traffic).

However, during the development, I encountered issues with extracting these specific fields—ISP and USAGETYPE—using the IP2Proxy database. Despite correctly querying the database, the data for these fields returned as dashes ('-'), indicating that the values were not available or could not be resolved for the given IP addresses.

Moving forward, further investigation into the database's configuration or use of a more reliable geolocation service might resolve these data gaps.

6 Challenges Encountered

Throughout the development of this project, I faced several challenges that were critical to the project's progress. These challenges ranged from technical difficulties to accessibility issues, and some performance-related problems during data processing.

6.1 Technical Issues

One of the primary challenges I faced was my limited expertise in some of the areas necessary for this project, particularly in dealing with network protocols, server administration, and IP geolocation. As a result, I had to learn many of these concepts on the go, experimenting with different approaches to understand how various components of the project work together. This self-learning process often involved a significant amount of trial and error.

For instance, working with the IP2Proxy and IP2Location databases was more complex than initially anticipated. Understanding how to properly integrate these databases and retrieve the necessary fields (e.g., ISP and usage type) required additional research into how geolocation and proxy detection databases operate. The learning curve was steep, but it allowed me to acquire a deeper understanding of network traffic analysis and geolocation technologies.

6.2 Accessibility Issues

Another significant hurdle was related to the accessibility of resources needed to host and manage the web application. Initially, I was working with a Dockerized environment, but I quickly realized that using the IP addresses forwarded from the Docker container was problematic for the reverse proxy setup and SSL certificate generation.

In order to expose the web app publicly, I needed to temporarily open ports on my router to allow external access. This posed a security risk, as the application was exposed to potential attacks. Additionally, setting up SSL certificates for the domain was a challenge. I used the Win-ACME tool to generate SSL certificates, but the process of configuring the certificates and ensuring that everything worked correctly across multiple environments (local machine, server, etc.) took more time than expected.

6.3 Data Retrieval and Performance Issues

When attempting to retrieve the ISP and USAGETYPE fields from the IP2Proxy database, I encountered difficulties. Despite correctly querying the database, these fields often returned dashes ('-'), indicating that the data was unavailable or could not be resolved for the given IP addresses. This issue was

not anticipated and made it impossible to use the ISP and USAGETYPE fields as planned. Possible reasons for this could include incomplete coverage of the IP address ranges in the database or limitations in the database version used.

Furthermore, one of the performance-related challenges was the time-consuming process of evaluating and reading large files containing IP information.

Lazy evaluation was used to store results in memory (via caching), but this did not completely eliminate the performance issues when working with large datasets. The time-consuming nature of querying both databases and the log files made it difficult to scale the analysis efficiently without additional optimizations.

6.4 Slow Performance with Large Log Files

As the number of log entries increased, the process of parsing and analyzing the log files became noticeably slower. This became especially apparent as the size of the log file grew. Since each log entry required parsing, database queries, and IP enrichment, this process became increasingly resource-intensive over time.

For larger log files, this slow performance could become a significant bottleneck, limiting the ability to process data in real-time or on-demand. Efficient handling of larger log files, such as using batch processing or optimizing the reading and parsing process, became a necessity as the logs accumulated. This issue was exacerbated by the fact that the logs were continuously growing, requiring frequent processing of larger and larger data sets.

One potential improvement for addressing this performance issue would be to implement a more efficient logging strategy, such as storing logs in a database or using a more optimized format that can be more easily queried. Additionally, employing parallel processing or more efficient algorithms for reading and analyzing log entries would help to mitigate the slow performance observed with large log files.

7 Future Improvements

Given the difficulties faced, the next steps for the project would include:

- Optimizing data processing to handle large log files more efficiently, perhaps by implementing better batch processing or asynchronous querying.
- Investigating the IP2Proxy and IP2Location databases further to resolve the issue of missing data (e.g., ISP and usage type fields) and ensure comprehensive coverage for more accurate results.
- Improving the security of the application, especially in terms of port forwarding and public access during development, potentially by utilizing a secure cloud-based environment for hosting.
- Continuing to learn and experiment with advanced network protocols and web security techniques to better understand and mitigate security risks.
- Implementing a more scalable solution for log management, such as using log management systems (e.g., ELK stack or Splunk) to handle large volumes of logs efficiently.
- Log File Rotation To ensure that the log files do not grow excessively large, a log rotation mechanism is implemented. This helps manage log file size and ensures that older logs are archived and stored properly.

8 Conclusion

In conclusion, this project successfully implements a robust protection shield for an exposed web server using Nginx as a reverse proxy. By effectively capturing and analyzing incoming traffic through detailed logging, the system offers real-time insights into web activity. The integration of geolocation and proxy information enhances the traffic analysis, providing valuable data such as the origin of requests and the nature of the traffic.

The dashboard serves as an intuitive tool for visualizing the collected data, helping administrators track key metrics such as the most visited IPs, URLs, and HTTP status codes. This not only assists in identifying potential security threats but also provides a clear picture of user behavior, enabling data-driven decision-making.

However, the system's potential extends beyond its current capabilities. Future work could involve the addition of advanced security features, such as anomaly detection algorithms that automatically flag unusual traffic patterns or suspicious IP addresses. The ability to block or blacklist recurring malicious IPs and integrate additional external data sources would further enhance the security of the system. Additionally, integrating machine learning techniques to predict and mitigate threats could provide proactive protection.

Overall, this project demonstrates the power of reverse proxies in securing web applications, and it lays the groundwork for future enhancements aimed at improving the security, functionality, and scalability of web services.

References

[1] F5 Networks, Inc. Nginx reverse proxy, 2024. URL: https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/.

[2] Anshuman Singh and Brij Gupta. Distributed denial-of-service (ddos) attacks and defense mechanisms in various web-enabled computing platforms: Issues, challenges, and future research directions. *International Journal on Semantic Web and Information Systems*, 18:1–43, 04 2022. doi:10.4018/IJSWIS.297143.

[3] Wikipedia contributors. Nginx, 2024. URL: https://en.wikipedia.org/wiki/Nginx.

[4] Wikipedia contributors. Reverse proxy, 2024. URL: https://en.wikipedia.org/wiki/Reverse_proxy.