Longitudinal Analysis of SSH Honeypot Logs

Jasmine Dong

***Abstract*—Visualising attacks and attack patterns from Cowrie secure shell (SSH) honeypots can be challenging when working and handling vast amounts of data over a long period of time. Gathering and capturing information over an extended timeline can help identify changes in attackers' behaviour for specific periods, adding additional information to those already accessible aggregated data. The project aims to apply captured data logs from multiple instances of Cowrie honeypots deployed by the Victoria University of Wellington (VUW) cybersecurity team and use them to integrate a longitudinal analysis to visualise attack and attack patterns over a period of a couple of months.**

I. Introduction

A HONEYPOT serves as a security measure that entices potential cyber attackers by disguising itself as a potential target. By creating an open or otherwise deliberately vulnerable decoy, honeypots can lead attackers a stray and deter them away from critical IT systems [1]. Security analysts can acquire crucial data, including the identities of attackers and the specific techniques and tools utilised to attack and target vulnerable systems, by monitoring the influx of traffic into deployed honeypot systems. This information enables them to identify weaknesses and vulnerabilities in their systems and devise strategies to enhance their security measures.

The value of captured honeypot logs lies in their effective utilisation of extracting meaningful information and creating a visual representation that will facilitate learning and knowledge acquisition. Analysing and extracting large amounts of log data over an extended duration lacks a standardised method. One approach involves conducting a longitudinal analysis of honeypot log files between highly aggregated honeypot log data on one hand and detailed session playbacks on the other hand.

A significant challenge in analysing honeypot data is the examination and visualisation of log data collection from multiple honeypots over a prolonged period. This process is vital for identifying vulnerabilities in honeypot configurations and understanding the attackers' methodologies and attack campaigns. By employing a proficient longitudinal analysis and visualisation technique for honeypot log data, it becomes more feasible to detect trends and patterns in the attackers' behaviour that may extend over a specific timeframe.

The project aims to further extract valuable and meaningful information by employing and enhancing existing tools. Analysing the information gathered over time identifies changes in attacker behaviours for specific dates, providing a richer understanding beyond the insights from the current aggregated data. Furthermore, deploying honeypots in two different geographical locations, Los Angeles and London, by the VUW cybersecurity team will generate large text-based datasets, which may pose challenges for analysis and visualisation. Consequently, the objective is to extract and reduce the data into a more readable format. Additionally, detailed longitudinal visualisations are also intended to be generated, and these findings are planned to be integrated into the established threat map managed by the VUW cyber security team.

In order to handle large datasets effectively, with an emphasis on data reduction, it was crucial to seek out preexisting solutions designed for reducing longitudinal data and then extend them to include longitudinal visualisations. MapReduce programming offered a solution for further reducing the collected honeypot log data, and the discovered command tool encompassed all the essential features required for extraction. The primary challenge was identifying a method to enhance the visualisation of the extracted data using a longitudinal approach. Through the exploration of various graphs, inspiration was found, and subsequently, Python scripts were created to extract the data further, enabling the visualisation of data to align with specific graphs.

By utilising multiple programming languages, including Javascript, Cascading Style Sheets (CSS), HyperText Markup Language (HTML), and Python, along with various visualisation libraries for inspiration, interactive maps and arc diagrams were developed to illustrate honeypot data from Los Angeles and London over three months. Insights into attacker behaviours were uncovered, such as most-used commands, prevalent usernames and passwords, and the proportion of human to robot attackers. Displaying this data visually improves our understanding of attacker strategies and assists in strengthening our systems to prevent future attacks.

Honeypots often capture and store duplicate or redundant data, especially after an extended timeframe. Therefore, the cybersecurity team at VUW may be maintaining unnecessary data duplicates that can consume significant storage space and computing resources, providing no additional value to the study. Reducing data redundancy of the SSH Cowrie honeypot logs will result in lower energy consumption and a smaller environmental footprint.

Whether it is a dedicated server or a virtual machine in a data centre, the physical honeypot will eventually wear out, becoming electronic waste (e-waste). Handling old honeypot hardware responsibly is crucial, either by reusing or recycling it. Multiple virtual honeypots can be simulated on a single system using virtualisation or containerisation technologies, so this should be considered when efficiently configuring honeypots to reduce costs and potential e-waste [2].

This project was supervised by Masood Mansoori

II. Background

There are two types of server honeypots, physical and virtual. Physical honeypots are often complex and time-intensive to set up and deploy as specialised hardware and its own physical system is required to deploy. A virtual honeypot, on the other hand, runs on virtualised environments that simulate the behaviour of real systems and networks [2]. Honeypots are also categorised based on their interaction level. Interaction level refers to the level of capabilities offered to the attacker.

A low-interaction honeypot offers limited capabilities for an attacker to exploit and use, providing a limited virtualisation environment. A low-interaction honeypot, for instance, may limit the attacker to a predetermined set of supported software, packages and libraries. This could translate to restricting the installation of new packages, access to privileged accounts and support for a predetermined set of commands. The honeypot may also limit the amount of traffic and number of outgoing connections it generates to minimise the likelihood that the honeypot system may be used as a base to attack other hosts and networks. A downside of a low-interaction honeypot is its limited logging capability due to limited virtualisation and support for various features and commands.

On the other hand, a high-interaction honeypot provides a fully functional operating system for an attacker to use, with limited to no restrictions. This research uses a popular and highly capable medium interaction honeypot with extensive logging capabilities. This ensures a sufficient level of functionalities for attackers to use while ensuring detailed logging of the attackers' interaction with the system and ensuring the honeypot cannot be used to attack other systems.

Cowrie is a medium to high interaction SSH and Telnet honeypot to capture and log information about brute force attacks and shell interactions performed by attackers. Cowrie emulates a UNIX system using Python when operating in medium interaction mode (shell). On the other hand, when running in high interaction mode (proxy), it serves as an SSH and Telnet proxy that allows observation of attacker behaviour on a separate system [3].

III. Related Work

Due to its comprehensive logging capabilities, the Cowrie honeypot has been extensively used in literature to capture and analyse attacks. In this section, several publications will be discussed based on the analysis of Cowrie honeypot data.

Dominic Rudigier [4] completed and published his bachelor's thesis on the longitudinal analysis of SSH honeypot logs. An extensive analysis was done on honeypots, with Cowrie in particular. A command-line tool was developed, and several functionalities were created to assist in the longitudinal analysis of SSH honeypot logs and to reduce, analyse, and visualise changes over time in JavaScript object notation (JSON) log files generated by Cowrie instances. The command-line tool uses a MapReduce programming model to process large amounts of log data efficiently and in an acceptable timeframe. Utilising MapReduce enables the aggregation of results by dividing the processing into two steps: map and reduce [4]. Using a MapReduce programming model can assist in incrementally analysing all log files across multiple honeypots and allows parallel processing. The command-line tool contains two workflows; analyse-local and analyse-remote. analyse-local is a command to MapReduce all log files in a local folder by searching for the format cowrie.json.YYYY-MM-DD in it and creating a reduced.json file. Whereas analyse-remote is a command used to MapReduce all log files on a remote Cowrie node and generate a reduced.json file again.

Rudigier [4] utilised Sankey plot diagrams, a data visualisation depicting the flow of quantities between different entities. In the context of the thesis, they were used to visualise the flow of commands and connections between attackers and honeypots over time. The generated Sankey plots illustrated the relationships between internet protocol (IP) addresses, usernames, and commands performed by attackers over time. By these visualisations, we could determine the attackers' patterns and behaviours over time, enhancing honeypot security and the detection of novel exploits. Rudigier's system architecture and capabilities will be later discussed as this project aims to extend the capabilities of the currently developed tool by identifying and implementing additional analysis and visualisation components on several honeypots deployed globally by the cybersecurity team at VUW. The findings in his study will be applied to data collected over the past several months in London and Los Angeles, with the aim to identify potential features and patterns, which will be visualised accordingly in the research.

Rich [5] performed a six-year longitudinal analysis from October 2016 to September 2022 using data obtained from a deception network. The overall objective was to track the evolution of threat actors' tactics and strategies over this period. This study aimed to enhance the current literature by offering an in-depth understanding of cybersecurity challenges and closely examining the evolving cyberattack patterns and trends. In terms of methodology, the team incorporated several techniques, such as data preprocessing, exploratory analysis, clustering, and anomaly detection, all based on understanding patterns over time. Validating the derived results was necessary; hence, a thorough validation process was done, which ultimately testified to the reliability of their findings. During the initial phases of the study, the team focused on cleaning up the data from the honeypot. This involved the careful task of eliminating redundant information, patching any gaps in data, and systematically arranging it for subsequent evaluation. Central to their analysis were critical indicators like the time of the event, the source, and its intended target.

Additionally, they executed a thorough data normalisation process to ensure consistency among diverse data formats, establishing a solid foundation for a streamlined subsequent analysis. The study's geographical evaluation showed cyberattacks spanning six continents and 188 countries, with North America, Europe, and Asia leading in attack origins, particularly from the United States, Russia, and China.

Several attacks were identified from specific IP addresses from over 100 million collected IP address entries, hinting at potential botnet or centralised attack strategies. Despite the study's focus on numerical data, future work could benefit from incorporating qualitative insights, like expert feedback, to gain a broader understanding of network threats.

Ikuomenisan et al. [6] performed a comprehensive analysis of visualisation techniques used to interpret and convey attack patterns from honeypot network traffic data. Using the preferred reporting items for systematic reviews and meta-analyses (PRISMA) methodology, the aim was to carefully examine the range and intricacy of graphical methods featured in influential publications and, therefore, identify dominant trends in the representation of honeypot-related data.

The results indicated that from the initial 218 papers reviewed, only 37 were closely examined due to their importance. Predominantly, these studies relied heavily on traditional visualisation techniques, majorly utilising line, bar, and pie charts for statistical summaries. An issue was raised as these simplistic representations often masked deeper, more intricate insights into the data. In contrast, advanced visualisation methods such as box plots and histograms were rarely used, even though they offer a more detailed understanding. The consistent reliance on conventional visualisation methods, such as bar charts, line charts, and pie charts, indicated a ubiquitous trend in the literature, even though current advancements advocate for visuals with enhanced data density and precision. A key insight highlighted the unintended creation of confusion or misrepresentation stemming from poor visualisation decisions. This illustrates the need for clear visuals, minimal distractions, and easily understandable metrics in presenting data. The study emphasises the critical importance of effective data visualisation in cybersecurity research and points out the benefits of exploring visualisation methods beyond conventional approaches.

Valli [7] explored alternative methods for interpreting honeypot data, specifically using Graphviz and AfterGlow to visualise and better understand the data through visualisations. Graphviz is an open-source visualisation tool that offers varied graph layouts through other tools like Neato, Twopo, and Circo, interpreting files described with the DOT language. AfterGlow is a series of PERL scripts designed to generate link graphs from comma-separated values formatted files, transforming raw input files into normalised data sets for visualisations and analysis. The article highlighted that utilising DOT graphs from Graphviz offers a strategy to consistently highlight specific entities as the central focus of analysis visually intuitively for human analysts. However, it also discussed that generating bitmap Graphviz dot graphs can be processor-intensive, as evidenced by substantial central processing unit usage even on a high-specification machine during research trials. This processing demand notably hinders real-time graphical data creation from network security mechanisms like honeypots and firewalls, particularly as the data processed during initial tests was relatively minimal compared to potential larger-scale deployments. Implementing a structured query language (SQL) system to manage and process honeypot data instead of reprocessing log files marks a substantial advancement in analytical efficiency. Connecting with the Surfnet intrusion detection system, which integrates data into a preexisting SQL structure, effectively backs SQL functionalities with a Graphviz/afterglow engine to create visuals. Improving Surfnet allows the production of near-real-time graphical data and establishes a method for in-depth, long-term analysis of patterns and trends in honeypot data. This strategy mitigates specific temporal and spatial challenges identified in traditional textual analysis engines, enables the retrieval and review of specific data from the database, and provides temporary storage within a database framework, assisting in buffering and stabilising data transmission fluctuations.

Junaid et al. [8] analysed captured Internet of Things (IoT) attacks over four months using a medium-interaction server honeypot, Cowrie, on a public network. They chose Cowrie for this research paper because it simulates IoT services, including SSH and Telnet protocols, emulating a fake file system and allowing file manipulation. It can also effectively mimic low-powered IoT hardware without risking full system compromise. The authors engineered a new feature-based set on three feature groups: the depth of attacker interaction, attacker and behaviours, and utilised resources. This analysis led to a new set of 20 features that provided broader traffic coverage compared to previous models. The feature extraction process was performed in two stages, where they identified 52 distinct attack patterns representing 30,335 attack sessions. They employed automated machine learning analysis to tackle the complexity of manually comparing 52 attack patterns across 20 features. Specifically, unsupervised learning through clustering and a random tree classifier to group similar attacks and determine key characteristics. The authors identified that the work could be further extended by extracting new features to classify IoT attacks according to humans and botnet attackers, and they could focus on extracting more features such as mistyping, typing speed, and spelling mistakes. The study also advocated for developing preventative measures based on the early detection of attack patterns.

Junaid et al. [9] utilised Cowrie's data for feature identification and study of human attackers. They discussed the characteristics that differentiate human attackers from bots, such as adaptability, variability in attack processes, and different responses to failed attack actions. Human attackers can think outside the box and brainstorm ways to exploit system vulnerabilities. Over two months, the author deployed 15 Cowrie honeypots in five geographical locations, including North America, Australia, Singapore, Amsterdam, and New Zealand, to capture and analyse attacks. The study identified numerous features to help identify human attacks, such as instruction patterns, usage of modifier keys, cursor control keys, and other keys such as backspace, tab, enter, space bar, delete, and shortcut keys for copy, paste, exit and enter. Upon successful login, it was observed that human attackers made typographical errors and spent considerable time gathering information about the devices. Five case studies

of attackers were examined to identify the characteristics of human attackers and examine and analyse their skills, ability to exploit devices, and potential intentions. The attackers' behaviour and intentions varied broadly, from obtaining basic information by looking at the system information, central processing unit information and available free memory to the user's command history, downloading and installing malware files, and deleting cookies and history to remove any traces of malicious activity.

Junaid et al. [9] also found that deploying honeypots at various locations with improved deception resulted in longer engagements from attackers. In contrast, custom honeypots that were more IoT-oriented received fewer successful login attacks. Their findings suggest that strategically deploying honeypots in various locations and improving their level of deception can lead to increased engagement from attackers. They established that custom honeypots for IoT-specific attacks could be particularly effective in attracting targeted attacks. Furthermore, enhancing the deception within these custom honeypots can prolong the duration of their interactions.

Melike, Ebu Yusuf, and Muhammed [10] deployed an SSH Telnet honeypot using Cowrie software to understand the threats targeting these types of honeypots and record the activities of attackers who attempted to gain access to them. The paper further analysed the three classifications based on interaction: low interaction, medium interaction, and high interaction. Then, it identified the two classifications based on installation: production and research honeypots. The decision to use Cowrie was its ability to track effectively and analyse SSH and Telnet attacks and automatically record login credentials such as their username or password and instructions, outputs and timestamps of the whole shell session when the attacker successfully logs in. Additionally, Cowrie facilitates integration with various applications, such as the Elasticsearch, Logstash, and Kibana (ELK) stack tools, which can simplify the records analysis. Beats transfer logs to the server, Logstash organises them, Elasticsearch makes the data searchable, and Kibana provides visualisation. The project recorded and evaluated attacks over 47 days, and protocol, usernames, passwords, IP addresses, and countries were chosen as the headers for analysis from the log records. Results showed a high degree of commonality in the passwords used in attacks across honeypots. To conclude, the experiment lasted approximately 47 days, utilising an SSH and Telnet honeypot system, yielding a substantial amount of data exceeding 3.5 million.

Zachary [11] enhanced the analysis and visualisation capabilities of Cowrie, an open-source medium interaction server honeypot. It aimed to provide a modular threat map capable of visualising attacks on multiple Cowrie honeypots in near real-time. The solution was developed using the ELK stack framework, allowing for an extensible and scalable solution to support the growth and changes in requirements of the cybersecurity group at VUW. The delivered solution provides a Kibana canvas interface with various visualisations, including a visually engaging threat map and various metrics that form an attack map. The client can customise the existing canvas, create new visualisations, and generate multiple dashboards and canvases. This enables the client to tailor the information they want to share with different target audiences, offering infinite granularity in presenting the data. Despite its flexibility and comprehensiveness, the current system is very complicated for clients' needs and faces reliability issues due to the many components, packages and libraries it uses, such as Log stash, Elastic Search and Kibana. According to the client, the system components fail regularly. Throughout the research conducted for this project, Zachary observed that most established threat maps were proprietary and closed-source. These solutions are often developed by commercial organisations that consider threat intelligence as a valuable asset for their business.

Consequently, the inner workings and details of these solutions remain private. As a result, there is limited visibility into the specific sensors used for attack data collection, including the level of interaction they offer.

Additionally, direct access to their live data feeds is not available. The existing system lacks a MapReduce-like mechanism, accumulating unnecessary data and the inability to effectively handle large volumes of data from multiple instances of Cowrie honeypots over time. Therefore, it becomes necessary for the system to periodically flush all records to ensure that only meaningful information is extracted.

Table I summarises the literature mentioned above on Cowrie honeypot's log analysis, visualisation and attack patterns.

IV. Design

A. MapReduce programming model

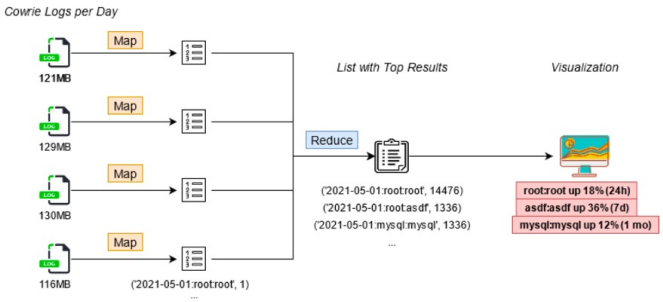

Fig. 1: MapReduce Diagram [4]

Rudigier [4] utilised the MapReduce programming model to process and analyse information from honeypots in parallel using multiprocessing. In his thesis, he illustrated his intended workflow, shown in Figure 1, which included multiple local Cowrie log files and the events contained within these were mapped based on their occurrence count.

TABLE I: Summary of Literature on Cowrie Honeypot's Log Analysis, Visualisation, and Attack Patterns

Author	Date	Area of Research	Findings
Dominic Rudigier [4]	2021	SSH Honeypot Logs	• Longitudinal analysis of SSH honeypots • Command line tool • Utilises MapReduce • Visualised using Sankey plot diagrams
Marshall S. Rich [5]	2023	Cyber Tactics and Techniques	• Longitudinal Analysis of six years using a deception network • An exhaustive exploration of the tactics and strategies utilised by cybercriminals to understand their modus operandi • Understanding how these tactics and techniques have matured in sophistication and target specificity over time
Gbenga Ikuomenisan et al. [6]	2022	Visual Methods in Honeypot Attack data	• Analysed honeypot data visualisation in key papers using PRISMA • Of 218 papers, 37 were crucial, predominantly using traditional charts such as line, bar, and pie charts • Highlighted the need for better visualisation in cybersecurity research
Craig Valli [7]	2009	Visualisation of Honeypot data	• Utilising Graphviz and AfterGlow for enhanced visualisation and interpretation of honeypot data • Generating Graphviz dot graphs is processor-intensive, challenging real-time data visualisation • Implementing SQL systems improves analytical efficiency and facilitates in-depth analysis of honeypot data trends
Junaid Haseeb et al. [8]	2021	IoT Attacks	• Used Cowrie to simulate IoT services • Created a new set of 20 features that provided broader traffic coverage • Identified 52 distinct attack patterns representing 30,335 attack sessions
Junaid Haseeb et al. [9]	2023	Deception-Based Security	• Discussed the characteristics that differentiate human attackers from bots • Deployed 15 Cowrie honeypots in five geographical locations • The attacker's behaviour and intentions varied broadly
Melike Başe et al. [10]	2021	SSH and Telnet Protocols	• Analysed the three interaction classifications • Discussed the ELK stack • There was a high degree of commonality in the passwords
Zachary Scott [11]	2022	Super V. Cowrie	• Provided a modular threat map capable of visualising attacks on multiple Cowrie honeypots • The solution provided a Kibana canvas interface with a variety of visualisations

Initial steps involved implementing existing solutions to understand the underlying structure of their system. The project began by cloning Rudigier's repository [4] and configuring and installing the necessary tools to ensure it ran successfully. Before executing any commands, a virtual Python environment created an isolated environment with its own Python libraries, dependencies, and configurations, preventing conflicts with other projects. Various commands were then explored [4] to MapReduce ("Python3 cowralyze.py map ..." and "Python3 cowralyze.py reduce ..."), analyse ("Python3 cowralyze.py statistics ...") and visualise ("Python3 cowralyze.py visualize...") changes over time in JSON log files generated by Cowrie instances, acquiring a more comprehensive understanding of their functionality. After successfully configuring the commands as intended, Cowrie honeypot data logs captured by the cybersecurity team at VUW were obtained. These logs originated from the honeypots deployed in Singapore and London. Subsequently, the analysis of the logs proceeded.

1) Map: The initiation of the MapReduce programming paradigm involves the utilisation of a map function. The map function processes individual log files and associates each event with its corresponding count within that file for a particular day, illustrated in Figure 2 [4]. This helps extract meaningful information from the log files and create an intermediary map result.

Fig. 2: Mapping individual London honeypot logs

2) Reduce: Subsequently, the reduce function comes into play, illustrated in Figure 3, which aggregates the count for the specific day and provides a reduced file with the aggregated counts per event per day [4]. This aggregation is performed based on the honeypot or host instance, known as a sensor. Furthermore, the reduction process assists a longitudinal analysis approach by combining one or more mapped files, each representing different days, into a reduced.json file. This step is crucial as it takes the intermediate results from the map phase and performs the necessary aggregation to produce the final output. Additionally, the reduced.json file serves as the foundation for visualisation and additional analysis.

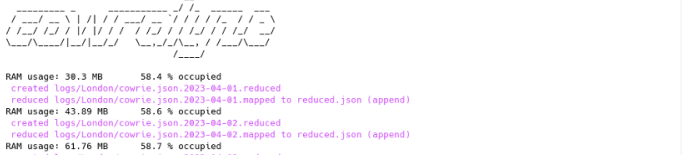

Fig. 3: Reducing mapped files into reduced.json

3) Visualisations - Graphs and Tables: After consolidating multiple Cowrie log files in a singular file called reduced.json, the next step was to produce statistics and graphs to understand the growth rate of event occurrences within a specific timeframe. Executing a straightforward command on reduced.json with parameters '-t' set at 20.0 and '-n' at 7 illustrates percentage shifts over the past seven days exceeding a 20 percent change. An HTML file, shown in Figure 4, is subsequently created, including a table detailing the attack's date, event type, command executed by the attacker, and the percentual change over the previously defined days. Through a deeper examination of attacker tactics, it is possible to discover new malware or vulnerabilities being exploited. For example, a recurrent unauthorised access attempt targets the "root" username and default password, especially during worm campaigns. Additionally, attackers often use the "chattr" command, which stands for "change attributes", to alter file properties, potentially to avoid detection, ensure sustained presence, or disrupt normal system functions.

% increase over time across all honeypots (n=7, Thresh=20.0%)

Date	Event	Command	Counts	% Overall	% change n days
2023-04-02	Login	root:3245gs5662d34	402	51.13	82.18
2023-04-02	Login	45gs5662d34:345gs5662d:	400	49.39	78.84
2023-04-02	Cmd	cd ~ && rm -rf .ssh && mkdir .ssh && echo "ssh-rsa..	404	50.4	62.06
2023-04-02	Cmd	cd ~; chattr -ia .ssh; lockr -ia .ssh	404	48.94	77.97
2023-04-02	Cmd	curl: option -L not recognized curl: try 'curl --help' or 'curl --manual' for more information	90	32.35	48.35
2023-04-02	Cmd	-aw.githubusercontent.com/	90	32.35	95.65
2023-04-02	Pre-disc-cmd	cd ~ && rm -rf .ssh && mkdir .ssh && echo "ssh-rsa..	402	49.65	61.26
2023-04-02	Pre-disc-cmd	curl: option -L not recognized curl: try 'curl --help' or 'curl --manual' for more information	90	32.35	48.35
2023-04-02	Pre-disc-cmd	uname -s -v -n -r -m	5	-73.68	-80.77
2023-04-02	Pre-disc-cmd	df -h | head -n 2 | awk 'FNR == 2 {print $2;}'	2	-69.23	-75

Fig. 4: Generated stats.html file

Fig. 5: Generated results.html file

The existing solution also created visualisations representing the changes over time on different honeypots for different events, such as password combinations, number of connections, connection frequency, number of downloads, and number of uploads. Extracting and analysing this information becomes meaningful as it enables the identification of trends and patterns performed by attackers over a specified period. A simple command generates an HTML file containing several plot graphs illustrated in Figure 5. The Sankey plot graphs are created using a Python library called Plotly. The tool's advantages are that it can analyse Cowrie events and create 2D plots to visualise data aggregated over all honeypot instances and 3D plots to split the specific event by sensor. However, the disadvantage is that it only illustrates the data in a plot-like format, making it very complicated to read when there are large amounts of data.

B. Alternative designs and systems - Elastic Stack

In the literature review, the ELK stack, a comprehensive software stack designed for log management and analytics, was utilised. As illustrated in Figure 6, the stack employs a three-tiered approach for proficient log analysis. Initially, Logstash is used to identify and collect log sources, structuring them methodically. These refined logs are subsequently forwarded to Elasticsearch, the second stage, where they are assembled, facilitating users to identify and understand

data trends and patterns. To conclude the process, Kibana offers tools to produce visual representations and interactive dashboards, optimising data interpretation and review [12].

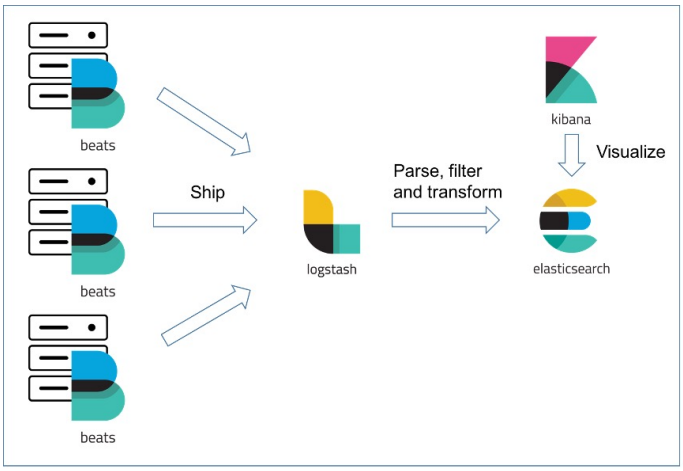

Fig. 6: ELK stack diagram [12]

C. Comparison

MapReduce and ELK stack are great systems that can analyse many Cowrie log files. Each has its strengths and limitations, including the complexity of the data, the type of analysis that wants to be performed, and the available resources. Comparing the two, MapReduce is more suitable for processing and generating large datasets in a distributed and parallel manner. It can handle large files efficiently and process various types of data. However, it is more complex and time-consuming than setting up an ELK stack, and it does not have the capabilities to produce visualisations. Whereas ELK stack is specifically designed for log and event data analysis and is well-suited for situations where you need to search, analyse, and visualise log data in near real-time. It is easy to set up and has built-in visualisation tools like Kibana. However, even though Elasticsearch can handle large volumes of data, it might not scale as efficiently as MapReduce for very large datasets. It is designed more for search and simple aggregations than complex ones.

As the primary goal for my project is a longitudinal analysis of SSH honeypot logs where we will be using a large amount of data to identify attackers' behaviours, using the MapReduce approach is more appropriate. After comparing the capabilities of the ELK stack and MapReduce programming, the decision was made to utilise MapReduce for several reasons.

• Powerful Data Processing: MapReduce processes extensive datasets, making it ideal for handling new Cowrie log files that require transformation and aggregation to extract meaningful information, such as trends in attacker behaviour.

• Simplified Implementation: The design is straightforward, and with existing open-source script files provided by Rodigier, the implementation process becomes simplified through copying, pasting, and subsequent modifications for integration into the new system.

• Efficiency and System Enhancement: MapReduce exhibits higher efficiency in processing large datasets than the ELK stack, potentially reducing processing power requirements and promoting better energy consumption. Despite the ELK stack's graphing capabilities via Kibana, enhancements within Rudigier's system will focus on long-term, valuable data representation.

D. Sustainability and Environmental Considerations

Both MapReduce and ELK stack can handle vast volumes of data, but this efficiency comes at the cost of substantial energy consumption. Therefore, it becomes crucial to employ effective energy management techniques and optimise the MapReduce algorithms to minimise energy usage and consumption time [13]. One possible solution to reduce energy is reducing the size of the Cowrie log files before employing MapReduce or ELK stack. This can be accomplished by eliminating redundant data or compressing the log files before processing, reducing the overall energy consumption when deploying such techniques.

When examining the integration of efficient data processing techniques, such as MapReduce and the ELK stack, with methods to minimise Cowrie log files, there are clear implications aligned with the United Nations (UN) Sustainable Development Goals [14]. Emphasising energy efficiency by reducing data size and optimising algorithms supports Goal 7, which focuses on "Affordable and Clean Energy" [15]. This approach reduces the energy requirements of data processing and advocates for a more energy-efficient digital infrastructure. Furthermore, such advancements contribute to Goal 9, "Industry, Innovation, and Infrastructure" [16] by establishing the foundation for sustainable industrialisation and resilient infrastructure development. Equally relevant is Goal 12, "Responsible Consumption and Production," where reducing computational resources through optimising data processing exemplifies resource efficiency within the tech sector. Finally, by reducing the energy requirements of large-scale data centres, we contribute to achieving Goal 13, "Climate Action," [17] as these centres are substantial sources of global carbon emissions. By adjusting our tech approaches to prioritise energy efficiency, we naturally align our strategies with the fundamental principles and aims of multiple UN Sustainable Development Goals.

V. Implementation

A. Reducing and Extracting Honeypot log files

After the design phase, the decision was made to utilise MapReduce to process Cowrie honeypot logs obtained from the cybersecurity team at VUW. The reason for choosing MapReduce was the idea of its simplicity to deploy and adaptability for extracting meaningful information. One limitation, however, was its capability for visualisation, but this was addressed and extended as part of the project. The collected logs originated from honeypots deployed in Los Angeles and London and were presented in JSON format, as shown in Figure 7. This project utilised a dataset covering

three months, specifically from June to August, with logs recorded daily. As a result, 92 logs from Los Angeles and an equal number from London, totalling 184 log files, needed to be condensed, extracted, and subsequently visualised.

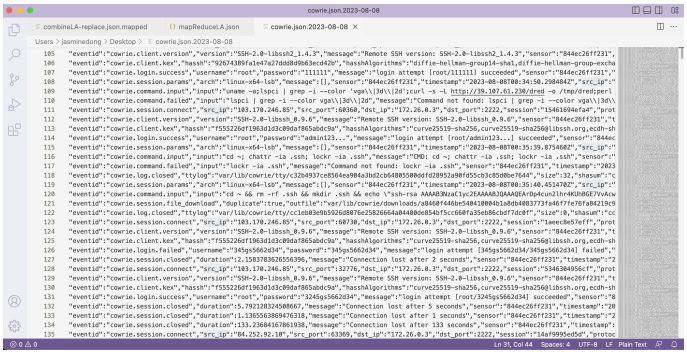

Fig. 7: 08-08-23 Los Angeles honeypot log file

The first step was to analyse the log files to identify what meaningful information could be extracted and what would be valuable visualised in a longitudinal approach to identify the attackers' trends and behaviours. The current Sankey plots illustrated the relationships between IP addresses, usernames, and commands performed by attackers over time. Rich's article on a longitudinal analysis of cyber adversarial tactics and techniques [5] discussed how such indicators served as effective in revealing potential threat actors. Additionally, assessing the attackers' geographical origin, the incident's timing and the nature of commands executed within specific periods can also help identify potential botnet activities. The log files, as shown in Figure 8, were reviewed using this data. Specific elements, such as successful and unsuccessful username and password attempts, the originating IP address of the attacker, and commands executed on the honeypots, were highlighted.

Fig. 8: Highlighting meaningful information

From examining the log files and reviewing related research, insights were obtained regarding the types of data that are valuable when visualised longitudinally to identify attacker patterns and techniques. The key features that can be extracted and their significance will be discussed next.

1) Commands: Examining the top commands and their patterns over time offers insights into attacker behaviours. Patterns that repeat may suggest an organised attack by a particular group, potentially allowing for attribution. Observing how these patterns change over time can help provide insights into attack evolution and the adaptability of attackers. This knowledge will enable organisations to prioritise defences based on common sequences, ensuring timely preventive actions. Additionally, by monitoring commands, security researchers can create alerts using command signatures and adjust security systems like host-based intrusion detection systems to notify administrators when a recognised command is issued on a host in a production environment.

2) Username and Passwords: Examining top usernames and passwords over an extended period offers valuable insights into attacker preferences and behaviours. This longitudinal analysis can highlight commonly targeted credentials, revealing evolving attacker tactics and possibly attributing attacks to specific threat actors. Any sudden surge in particular username or password attempts might indicate coordinated campaigns or newly released exploit kits. Additionally, consistently targeted weak passwords suggest the need for better password policies and user education. This analysis gives organisations the insights needed to enhance their security measures and adjust to the evolving threat environment.

3) Robots vs Human: Understanding whether a threat actor is human or robotic over an extended period is essential for deciphering attack patterns. A consistent robotic signature indicates automated campaigns, possibly revealing broader cyber threats and campaigns or botnets, whereas fluctuating human patterns can signal targeted attacks, advanced persistent threats or evolving tactics. Distinguishing between these two sources enables organisations to tailor their defence strategies and anticipate future threats more effectively. An example is a honeypot system that redirects bots and automated bots to other security systems such as Tarpits [18]. In contrast, human attackers are redirected to high-interaction honeypots that provide a fully functional operating system to capture detailed behavioural data.

4) Top Countries: Identifying the primary countries from which threat actors originate over a prolonged period is instrumental in mapping attack patterns. Recognising consistent origins can shed light on potential geopolitical motivations, state-sponsored activities, or region-specific cybercrime trends. This knowledge enables organisations to enhance their defences against particular regions and better anticipate and respond to emerging threats.

After deciding on the features to extract, the next step involved employing the MapReduce function to emphasise these features and identify patterns for visualisation. Figure 9 displays a mapping from a Los Angeles honeypot log file, highlighting timestamps and events like session connections or logins. Further examination of the mapping script was done to establish if more features could be extracted. However, information like the attacker's country of origin wasn't directly available in the JSON log files. Therefore, a more in-depth extraction was required by extracting source IP addresses to pinpoint the corresponding countries.

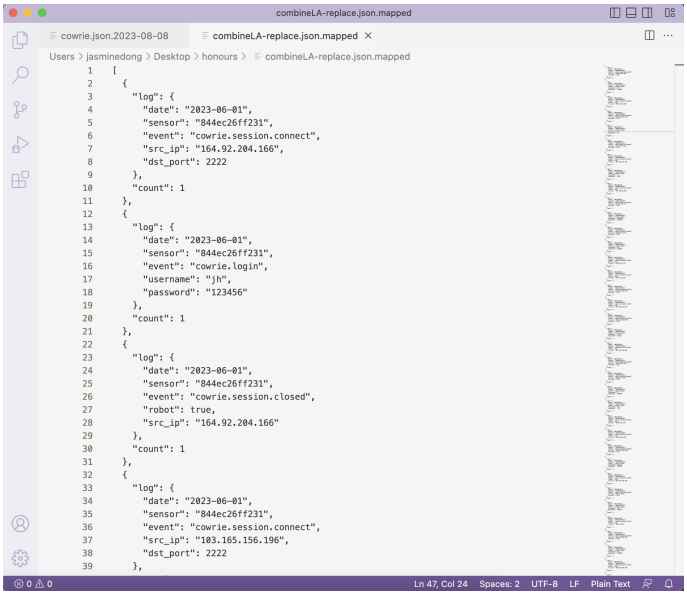

Fig. 9: Los Angeles log file mapped

The subsequent step involved consolidating the mapped files by aggregating all 92 files from Los Angeles into one file and doing the same for London. Through this process, we could identify the frequency of specific commands and the most commonly used usernames and passwords, helping us identify the attackers' patterns.

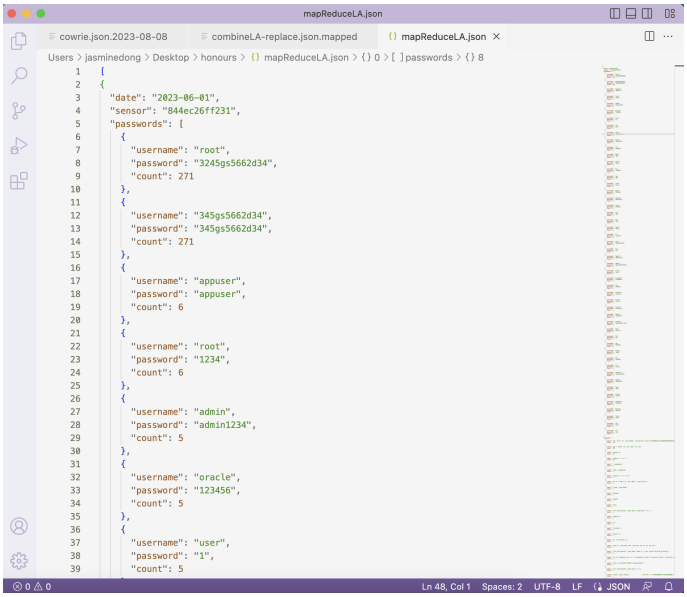

Fig. 10: Reducing the individual mapped log files

Illustrated in Figure 10 is the reduced.json file, which is performed by combining all the individual mapped files and then aggregating the count for the specific day and providing a reduced file with the aggregated counts per event per day. The current solution offers the option to specify a count using the parameter "-n *". For my project, a value of 2000 was selected, enabling the acquisition of the top 2,000 events. This allowed for the identification of tactics spanning a period of 2,000 days.

B. Visualisations

After extracting the data, the focus shifted to exploring visualisation libraries to determine the most effective way to display the data, ensuring clarity for security researchers longitudinally. As identified by Ikuomenisan et al. [6], bar, line, and pie charts were commonly used in studies, while plots and histograms, which often provide deeper insights, were less frequently utilised. Due to poor visualisation, some graphs led to data misrepresentation, emphasising the importance of clear, minimal distraction visuals. Therefore, these findings influenced the development of the visualisations made for this project.

1) Plotly and Matplotlib: At first, the Python visualisation library, Plotly [19], used by Rudigier [4], was considered. While leveraging existing scripts from this library would have been convenient, its graphing capabilities did not seem well-suited for handling extensive data sets. Although it offered visuals like scatter plots, which resemble Sankey plots, bar charts, pie charts, and histograms, these were already commonly used. This led to an exploration of other Python visualisation libraries, including Matplotlib [20]. Yet, after thorough evaluation, none of the available visualisations satisfied the vision of longitudinally visualising the data extracted, such as top command sequences, top usernames and passwords used and top countries.

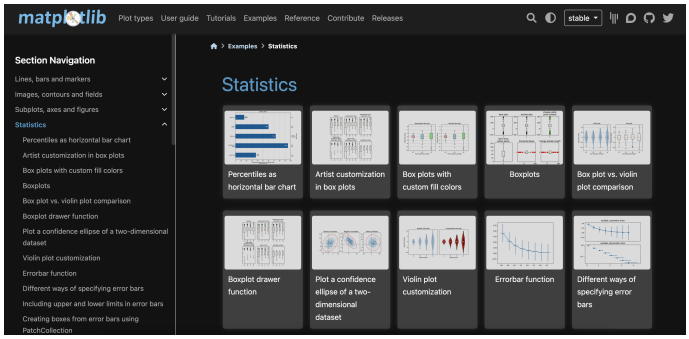

Fig. 11: Matplotlib Visualisation Library [20]

2) Plotset: The movement to consider other options and shift away from familiar Python libraries added a layer of complexity, but something that needed to be explored. After further discussion, Plotset and Vega were proposed and were, therefore, something that was looked into. Initially, Plotset [21] offered a vast array of visualisation options, including maps, distribution charts, bar charts, line and area charts, as well as others. This seemed ideal for longitudinal data display, and the tool's flexibility to change the colours and layout was an added advantage. However, a significant limitation was its capacity to handle large datasets. While Plotset could link to Google Sheets, transferring hundreds of thousands of lines in an appropriate format would cause issues such as exceeding memory. A potential solution could involve reducing the

dataset size, but this approach might introduce complexities, especially when moving data across platforms and potentially working with more data than initially planned.

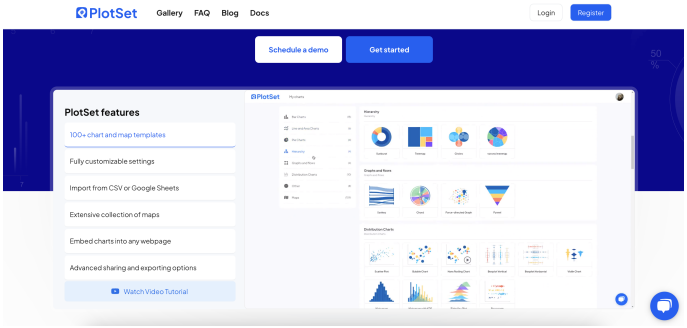

Fig. 12: Plotset Visualisation Library [21]

3) Vega: Furthermore, Vega was explored [22], revealing several graphs suitable for longitudinal data visualisation. The arc diagram, in particular, stood out for its potential in visualising attack command sequences and top commands commonly used. The layout provided was in JSON data format, which was more than ideal with the log files provided, offering a way to extract further and format the data for visualisation. The next step involved exploring languages that might support this visualisation. While these libraries explored provided valuable insights into different graphs and data formatting, implementing and visualising the graphs with the data provided using a language rather than Python presented challenges.

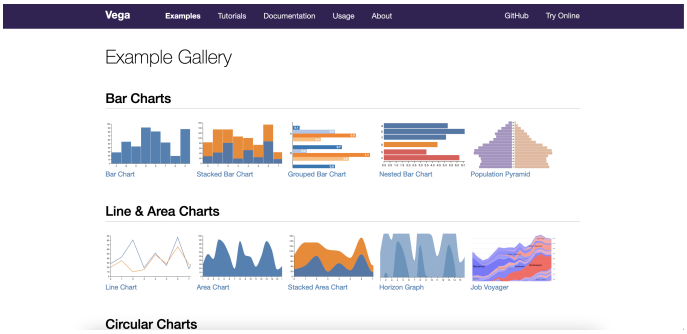

Fig. 13: Vega Visualisation Library [22]

4) D3 Graph Gallery: During the weekly Owhiti meetings, the challenges of converting particular graphs into a Python format were discussed. The potential solution of using Javascript was suggested, leading to further exploration. The discovery of a D3 graph gallery [23] was found, which showed a collection of simple charts designed with D3, a Javascript library. Among these was an arc diagram including template Javascript code and data input format [24]. Being unfamiliar with JavaScript, this presented a learning curve. Another challenge to overcome was to understand the required data format and the relationships between nodes, their associated nodes, and links. The next crucial task was transforming the reduced file from MapReduce to fit the arc diagram's format. This was achieved by writing Python scripts to extract the top commands from the reduced file, ensuring they were formatted to align with the appropriate nodes, associated nodes and links.

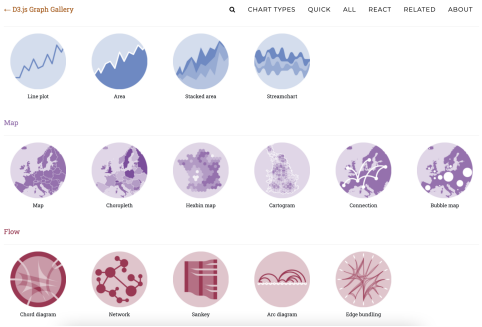

Fig. 14: D3 Visualisation Library [23]

5) Attack Map: Another visualisation option was a map highlighting the top countries of origin. An "Attack Map" was discovered on a website that utilised JavaScript, HTML, and CSS for an interactive experience [25]. This site offered a GitHub link with open-source code, enabling data integration for an interactive map [26]. The primary challenge was adapting the honeypot log data to fit this map's requirements. The process required extracting source IP addresses, and therefore, a Python script was created which involved using an API [27] to retrieve IP-related information for every source IP address that was extracted from the honeypot log files. The information included latitude, longitude, proxy, and hosting, to name a few.

Table II provides a summary of the various visualisation libraries that could have been used to display the extracted Cowrie honeypot data.

VI. Evaluation

The project aimed to further extract valuable and meaningful information in a more readable format by employing and enhancing existing tools and then generating longitudinal visualisations to illustrate the data to discover threat actors tactics and techniques. MapReduce was used to consolidate three months of data into a single file, and the time it took to reduce was extremely efficient. Therefore, making further modifications was unnecessary. If larger datasets were to be processed, the efficiency of MapReduce might be challenged, and its limits may be tested. In such scenarios, preliminary data reduction, particularly of redundant and repetitive data, might be necessary before deploying MapReduce. While the existing tool adequately managed data reduction, it fell short in the visualisation department. It was essential to leverage the reduced data to present meaningful information over an extended period. Data like source IP addresses and commands were further analysed, utilising IP APIs to determine the location associated with the IP addresses and to decipher the sequence of commands to understand possible tactics that the threat actors were performing.

TABLE II: Summary of Visualisation Libraries on Cowrie Honeypot's Log Data

Visualisation Library	Pros	Cons
Plotly [19]	• Utilised by Rudigier [4], potentially making it convenient to leverage existing scripts • Offers a range of visualisations, such as scatter plots, bar charts, pie charts, and histograms	• Not well-suited for handling extensive datasets • Insufficient for longitudinally visualising key extracted data like top command sequences, usernames and passwords, and countries
Matplotlib [20]	• Provides a wide array of visualisation options • Widely used in the Python community, which might provide a larger base of examples	• Did not satisfy the requirement for longitudinally visualising specified extracted data • May not offer as interactive or dynamic visualisations as some other libraries
Plotset [21]	• Provides the flexibility to alter colours and layout according to user preferences • Appears suitable for longitudinal data display	• Encounters limitations in managing large datasets, compromising its efficiency and reliability for extensive data • Could have data transfer issues as it uses Google Sheets
Vega [22]	• Utilises JSON data format for layouts, which is compatible and ideal with the provided log files • The arc diagram presents potential for visualising attack top commands	• Utilising a language other than Python to visualise the graphs with provided data posed difficulties • The need to learn new languages for visualisation might require additional time and resources
D3 Graph Gallery [23]	• Offers a collection of simple charts designed with D3, a JavaScript library • Provides an arc diagram, including template JavaScript code and data input format, which could streamline the graph creation process	• Will need to learn JavaScript and D3.js, which may slow down the initial development process • Necessitates extra steps and potentially complex scripting to transform data to fit the diagram's format
Attack Map [25]	• Utilises JavaScript, HTML, and CSS to create an engaging and interactive mapping experience • Offers access to open-source code through GitHub [26], enhancing accessibility and allowing potential customisation	• Requires significant effort and technical knowledge to adapt honeypot log data to meet the map's input requirements • Necessitates the creation and execution of scripts such as extracting and processing IP addresses

While Rudigier [4] developed both visualisations and statistical tables, it was observed that visualisations more effectively conveyed trends. Relying heavily on textual data can introduce confusion. Therefore, using tables to illustrate trends was deemed unnecessary for this project.

A. Arc Diagram

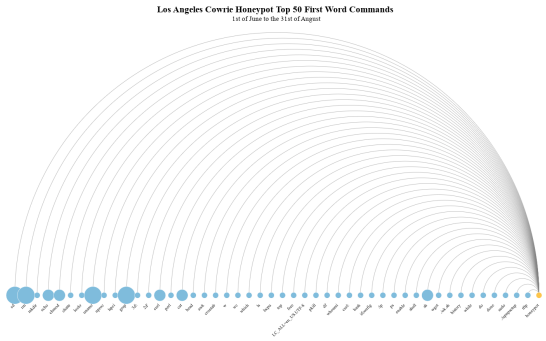

Fig. 15: Los Angeles Top 50 Commands

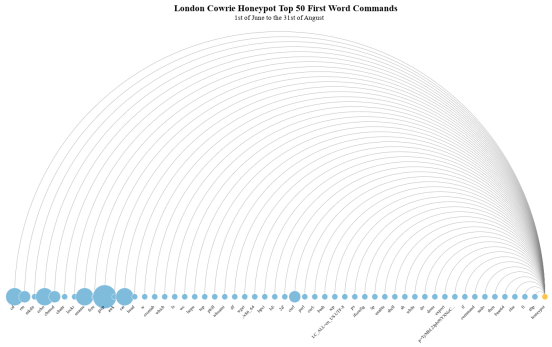

Fig. 16: London Top 50 Commands

Figure 15 displays the top 50 commands executed on the Los Angeles honeypot from June 1st to August 31st, excluding the subsequent parameters. Similarly, the top 50 commands for London during the same period were visualised shown in Figure 16. The initial idea behind this visualisation was to visualise the most frequently executed commands on the honeypot, thereby identifying potential malicious intents. Yet, feedback from user testing suggested that while the graph highlighted the most commonly used commands, a more insightful approach would be to show the relationships between various commands. Enabling a deeper comprehension

of frequently used sequences by attackers facilitates a clearer understanding of their tactics and techniques.

Therefore, an updated arc diagram was created in Figure 17 for Los Angeles and Figure 18 for London, showing the relationships between commands to the honeypot. The only disadvantage observed over the initial arc diagram is that, due to the presence of additional links to display, the data had to be reduced to the top 35 sequences to maintain clarity and visual ease of follow-through. Results show that the most commonly used commands were 'grep', 'echo', 'cd', and 'uname', which could suggest initial system reconnaissance by attackers. Specifically, 'uname' probes system details; 'cd' involves directory navigation; 'grep' looks through files for data or configurations; and 'echo' may indicate configuration changes or script implantation. The consistent use of these basic commands could reflect automated scans or foundational steps for more targeted attacks.

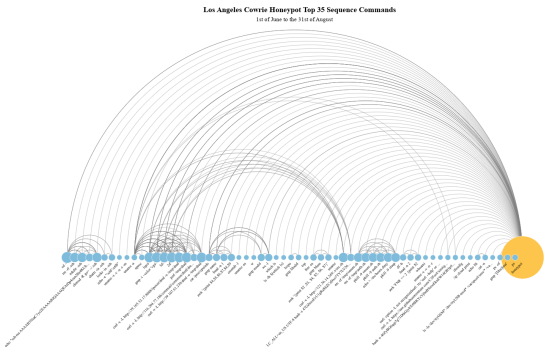

Fig. 17: Los Angeles Sequence Commands

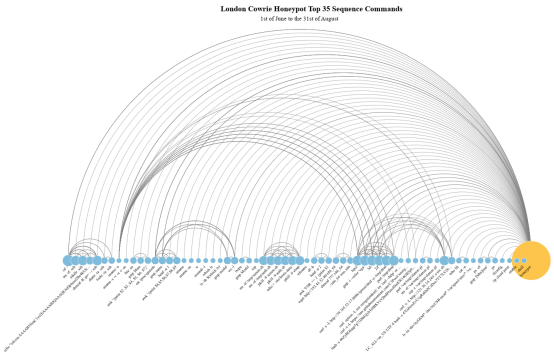

Fig. 18: London Sequence Commands

B. Interactive Map

Figure 19 displays the top 30 countries targeting the Los Angeles honeypot, and Figure 20 shows the top 30 countries targeting the London honeypot between June 1st and August 31st in an interactive map. In a cyber-attack context, three months can effectively highlight the most active countries, offering insights into the primary global threat behaviours and trends. Presenting only the top 30 countries ensures clarity and avoids overcrowding the visual representation. This focused approach likely encompasses the majority of critical attack vectors and enables a detailed examination of the strategies and methods adopted by each of these primary countries.

Fig. 19: Los Angeles Top 30 Countries Map

Fig. 20: London Top 30 Countries Map

Both honeypots discovered that the highest number of occurrences came from IP addresses in the United States and Singapore. This is crucial information as security resources can be prioritised and focused on threats from these regions. Additionally, it can offer insights into potential threat actors or groups, their motivations, capabilities, and objectives. Furthermore, ways to mitigate could involve blocking traffic or increasing traffic monitoring to understand the patterns more easily. There is, however, the idea that IP addresses can be spoofed, and virtual private networks can mask the true origin of an attack, so while geo-location data is valuable, it is essential not to rely solely on it.

C. Word Cloud

Some simple but effective pie charts were also created to illustrate the top 50 passwords and usernames for the Los Angeles and London honeypots using a Python script [28]. Knowing the top attempted usernames and passwords, such as "123456", "admin", "password", "root", and "ubuntu", offers valuable insight into attackers common methods. These predictable patterns suggest that many adversaries prioritise easily exploitable vulnerabilities. This emphasises the importance of implementing strong password policies and avoiding default or easily guessable credentials. Such information can guide the enhancement of intrusion detection systems to flag these generic attempts more efficiently and focus on securing frequently targeted accounts, enhancing overall system security.

1) Top Usernames: ”root” and ”admin” were the top two usernames in Los Angeles and London, as shown in Figures 21 and 22. The prevalence of these usernames likely stems from their common use as default administrator-level access credentials on various systems. Threat actors often target these usernames by leveraging brute force or dictionary attacks, aiming to secure the highest level of system access to manipulate and control it. Given that many systems employ default credentials, it is logical for attackers to prioritise these usernames when attempting unauthorised access.

Fig. 21: Los Angeles Top 50 Usernames

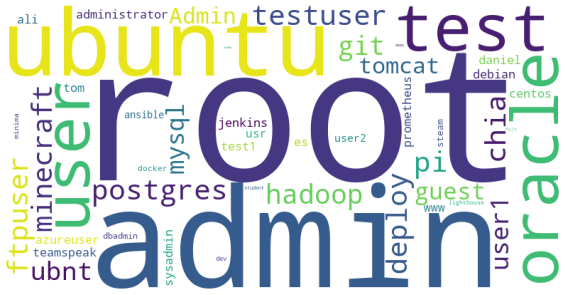

Fig. 22: London Top 50 Usernames

2) Top Passwords: Both Los Angeles and London had ”123455”, ”123”, and ”password” as some of their top passwords, illustrated in Figures 23 and 24. This may have been the case because many users opt for easily memorable and typeable passwords despite their weak security. Additionally, brute force attacks might start with commonly recognised weak passwords and bots, and automated scripts may be programmed to try simple passwords initially to maximise the chances of quickly breaking into accounts.

Fig. 23: Los Angeles Top 50 Passwords

Fig. 24: London Top 50 Passwords

D. Pie Charts

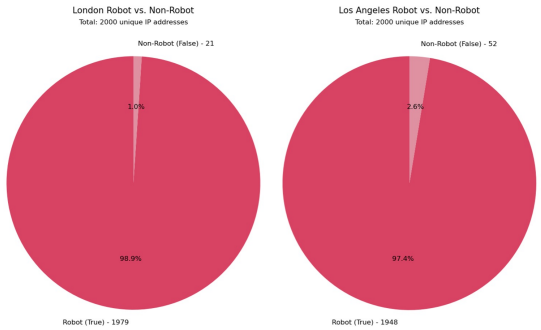

Fig. 25: London and Los Angeles - Robots vs Humans

Pie charts illustrating the relationship between robots and humans were also created. The robot parameter created by the visualisation suggests that if the connection time has been less than 10 seconds (configurable), it is most likely that this attack is scripted as any human being would need more time to execute the specified commands [4]. However, this can inadvertently produce false positives. This is because skilled human attackers might execute rapid operations, while bots could mimic slower, human-like interactions, both scenarios deviating from the parameter’s foundational beliefs.

Understanding whether a honeypot attacker is a robot or a human is crucial for longitudinal analysis, primarily for refining security strategies and comprehending threat evolution over time. Differentiating between automated and human-initiated attacks allows researchers to trace patterns, revealing how threat vectors evolve and if attackers (particularly humans) adapt strategies based on past interactions or security upgrades. This analysis helps adjust defences and predict future attack methods, aligning cybersecurity with evolving threats while correlating attack data with global trends to enhance predictive threat intelligence and preparedness.

VII. Conclusions and Future Work

Honeypots serve as a strategic cybersecurity tool by intentionally presenting themselves as vulnerable targets to entice cyber attackers, thereby safeguarding genuine information technology systems. This method facilitates the diversion of attackers from critical networks while security analysts collect key data concerning the attackers’ identity

and their chosen tactics and tools. Utilising honeypots allows analysts to decipher system weaknesses, which informs the enhancement of security protocols. A pressing issue in this field is managing and visualising data collected from various honeypots over long durations, which is crucial for understanding attackers' configurations, methodologies, and strategies.

The goal was to dive deeper into extracting meaningful information, using and improving existing tools, and analysing the gathered data over time. Implementing honeypots in two locations, Los Angeles and London, by the VUW cybersecurity team generated substantial text-based datasets, presenting challenges in analysis and visualisation. Therefore, there was an opportunity to minimise and refine the data into a more readable format. It was also important to create longitudinal visualisations, which would be incorporated into an established threat map monitored by the cybersecurity team at VUW.

While managing sizeable datasets effectively necessitates focusing on data minimisation, the MapReduce programming method was adopted as a potential solution to reduce honeypot log data further. The main challenge was deciding on a method to illustrate the visualisation of the extracted data longitudinally. While exploring various graphs, Python scripts were developed to facilitate further data extraction, enabling data visualisation in alignment with specific graphs. Honeypots frequently capture and store redundant data, potentially containing unneeded duplicates that utilise valuable storage space and computing resources without enhancing research outcomes. Mitigating this data redundancy, especially within the SSH Cowrie honeypot logs, resulted in reduced energy consumption and a diminished ecological footprint.

After concluding the project, many new opportunities have opened up for further research and growth in honeypots and cybersecurity. Future work could include further research to reduce honeypots' energy and resource consumption. Additionally, since the project only used three months of data, one idea for the future could be to use a more extensive six-month or yearly dataset. However, using MapReduce might be slower with more data. Furthermore, developing more algorithms that more efficiently manage data, potentially reducing the storage, computing power, and energy utilised is another idea. One more project could be to make dynamic, real-time visual maps of threats using the honeypot data to show global cybersecurity threats and predict new threats by looking at the data's location and timing and developing visualisation tools that represent attack patterns and highlight the techniques most often used by attackers.

ACKNOWLEDGMENTS

I wish to convey my sincere gratitude to Masood Mansoori for his consistent support and guidance. Furthermore, my appreciation is extended to my family and the 2023 engineering cohort for their continuous encouragement and support throughout my academic journey.

REFERENCES

[1] S. Lakhwani, "What is a honeypot? types, benefits, risks and best practices." https://www.knowledgehut.com/blog/security/honeypot, Jan 2023. (Accessed on 06/06/2023).
[2] N. Provos, "A virtual honeypot framework." https://www.usenix.org/legacy/publications/library/proceedings/sec04/tech/full_papers/provos/provos_html/honeyd.html, 1997. (Accessed on 06/06/2023).
[3] M. Oosterhof, "Github - cowrie/cowrie: Cowrie ssh/telnet honeypot https://cowrie.readthedocs.io." https://github.com/cowrie/cowrie, 2018. (Accessed on 06/06/2023).
[4] D. Rudigier, "Github - deroux/longitudinal-analysis-cowrie: Longitudinal analysis of ssh cowrie honeypot logs." https://github.com/deroux/longitudinal-analysis-cowrie, Nov 2022. (Accessed on 10/10/2023).
[5] M. S. Rich, "Cyberpsychology: A longitudinal analysis of cyber adversarial tactics and techniques," *Analytics*, vol. 2, no. 3, pp. 618–655, 2023.
[6] G. Ikuomenisan and M. Yasser, "Systematic review of graphical visual methods in honeypot attack data analysis," *Information Security*, vol. 13, no. 4, pp. 210–243, 2022.
[7] C. Valli, "Scholarly commons - annual adfsl conference on digital forensics, security and law: Visualization of honeypot data using graphviz and afterglow." https://commons.erau.edu/adfsl/2009/wednesday/4/, May 2009. (Accessed on 10/13/2023).
[8] J. Haseeb, M. Mansoori, H. Al-Sahaf, and I. Welch, "Iot attacks: Features identification and clustering," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 353–360, 2020.
[9] J. Haseeb, "Deception-based security framework for iot: An empirical study." https://openaccess.wgtn.ac.nz/articles/thesis/Deception-Based_Security_Framework_for_IoT_An_Empirical_Study/21965195, 2023. (Accessed on 06/06/2023).
[10] M. Başer, E. Y. Güven, and M. A. Aydın, "Ssh and telnet protocols attack analysis using honeypot technique: Analysis of ssh and telnet honeypot," in *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pp. 806–811, 2021.
[11] Z. Scott, "Super v. cowrie," 2022. (Accessed on 06/06/2023).
[12] "Elastic stack: Elasticsearch, kibana, beats & logstash — elastic." https://www.elastic.co/elastic-stack/. (Accessed on 06/06/2023).
[13] H. Chih-Chieh and H. Chu-Cheng, "Mapreduce - an overview — sciencedirect topics." https://www.sciencedirect.com/topics/engineering/mapreduce, 2017. (Accessed on 06/06/2023).
[14] U. Nations, "The 17 goals — department of economic and social." https://sdgs.un.org/goals, Jan 2016. (Accessed on 10/10/2023).
[15] U. Nations, "Goal 7 - department of economic and social affairs." https://sdgs.un.org/goals/goal7, Jan 2016. (Accessed on 10/10/2023).
[16] U. Nations, "Goal 9 - department of economic and social affairs." https://sdgs.un.org/goals/goal9, Jan 2016. (Accessed on 10/10/2023).
[17] U. Nations, "Goal 13 - department of economic and social affairs." https://sdgs.un.org/goals/goal13, Jan 2016. (Accessed on 10/10/2023).
[18] M. Rouse, "What is tarpitting?." https://www.techopedia.com/definition/1722/tarpitting, Jan 2014. (Accessed on 10/15/2023).
[19] "Plotly python graphing library." https://plotly.com/python/, 2023. (Accessed on 10/11/2023).
[20] J. Hunter, D. Dale, E. Firing, and M. Droettboom, "Statistics — matplotlib 3.8.0 documentation." https://matplotlib.org/stable/gallery/statistics/index.html, 2023. (Accessed on 10/10/2023).
[21] P. Inc, "Plotset - data visualization." https://plotset.com/, 2023. (Accessed on 10/10/2023).
[22] "Example gallery — vega." https://vega.github.io/vega/examples/, 2023. (Accessed on 10/10/2023).
[23] Y. Holtz, "The d3 graph gallery – simple charts made with d3.js." https://d3-graph-gallery.com/index.html, 2018. (Accessed on 10/11/2023).
[24] Y. Holtz, "Arc diagram template for d3.js." https://d3-graph-gallery.com/graph/arc_template.html, 2018. (Accessed on 10/11/2023).
[25] H. Target, "Cowrie honeypot analysis." https://hackertarget.com/cowrie-honeypot-analysis-24hrs/, 2023. (Accessed on 10/11/2023).
[26] "Github - lit-forest/leaflet.migrationlayer: Migration data visualization on map." https://github.com/lit-forest/leaflet.migrationLayer, Dec 2022. (Accessed on 10/11/2023).
[27] "Ip-api.com - geolocation api." https://ip-api.com/, 2022. (Accessed on 10/11/2023).
[28] D. Vu, "Python word clouds tutorial: How to create a word cloud — datacamp." https://www.datacamp.com/tutorial/wordcloud-python, Feb 2023. (Accessed on 10/13/2023).