EVENT-ORIENTED VISUALIZATION OF AUDIT LOGS

SRINIVASA RAMANUJAN SRIRAM

A PROJECT SUBMITTED
FOR THE DEGREE OF
MASTER OF COMPUTING – INFOCOMM SECURITY SPECIALISATION

SCHOOL OF COMPUTING

NATIONAL UNIVERITY OF SINGAPORE

2023

Project Advisor:
Associate Professor Liang Zhenkai

DECLARATION

I hereby declare that this project report is my original work, and it has been written by me in its entirety. I have duly acknowledged all the sources of information which has been used in the project report.

This project report has also not been submitted for any degree in any university previously.

Srinivasa Ramanujan Sriram

11/11/23

ACKNOWLEDGEMENTS

First and foremost, I am immensely grateful to Dr. Liang Zhenkai for having taken me under his wing and accepting to be my advisor for my master's project. I am indebted to Dr. Zhenkai for his mentorship and guidance not just for this project this semester, but for the first two semesters of my master's program too. It truly has been a wonderful learning experience allowing me to approach the field of security in different perspectives. I am deeply appreciative for him being very patient, compromising and supportive of me when I got stuck in between.

I would like to take this opportunity to thank all of Prof's PhD students who have helped me with the program and the project, starting with Liu Jiahao (who is my guide for the project), Zhang Chuqi, Anis Bin Yusof, Mingyuan Gao, Ahmad Soltani.

It would be criminal if I miss out on acknowledging the constant support from my dear friends at IIE (Indian Instrumental Ensemble), namely Ashwin, without whom I probably wouldn't have been able to complete the project and, Uma Gowri, Vaishnavi Hari, Sri Priyanka, Neha, Aparna and Xin Tong for all their love, affection, and motivation that they have showered which has kept me going even during the lowest of days.

This goes without saying, but I wouldn't be where I am now if not for my parents, their understanding and their support morally, financially and in every other aspect.

Finally, I thank The Almighty for sending so many opportunities my way and giving me the strength and guidance through others, to make wise use of them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SUMMARY

In our rapidly changing technological world, unpatched software vulnerabilities and a rising tide of cyber threats pose a serious threat to organizations. Log Analysis is a very daunting challenge for security professionals due to the nature of it. SIEM systems, EDR solutions, and SOAR platforms offer essential tools to streamline log analysis, visualization, and alerting, effectively empowering security engineers, analysts, threat hunters, and incident responders. This project explores open-source observability and security monitoring platforms, with a focus on Elastic Stack, building a simple entirely client-side plugin, to view audit logs in two levels to gain better understanding. Additionally, as extensions, ShadeWatcher's parsers and Neo4j's graph database are explored, seeking to improve log processing, behavioral analysis, and visualization to bolster cybersecurity efforts.

1. INTRODUCTION

With new technology comes new threats and with new changes comes new bugs. We have seen an increasing number of ransomware attacks, data breaches, phishing attacks etc. The average cost of a data breach in 2023 has been projected as $9.44 million [1]. Ransomware attacks have increased [2] by 20% and 90% of data breaches as attributed to phishing attacks [3]. With the popularity and legalization of cryptocurrencies in various countries, the most common types of malwares [4] are trojans, ransomware, cryptocurrency mining malware to name a few. Healthcare, financial & technology sectors are spending more than ever to protect their assets.

Log collection, storing, auditing and analysis are a few of the most important processes that help mitigate and battle cyber threats or even just to protect employees from themselves and their accidents. Be it for static or real-time analysis, logs provide us with a plethora of information on everything. High-volume log data, with its intricate details, often obfuscates critical events that indicate system compromises or malicious activities. Due to this very nature, it can be extremely onerous to find the information we need. It is literally looking for a needle in a haystack. The challenge lies not in the collection of data, but in its interpretation.

This is where Security information and event management (SIEMs), Endpoint Detection and Response (EDRs), Security Orchestration, Automation, and Response (SOARs) etc, come into play. They provide a nice, comfortable, customizable environment to parse, view, analyze, visualize logs, and create alerts based on signatures, behaviors etc. Typically, the job scope of a security engineer includes setting the environment, configuring the SIEMs, the endpoint agents, filters, etc and the job scope of a security analyst / threat hunter / incident responder to analyze logs and find useful information / threats.

So, what exactly are events in audit logs and what does visualizing audit logs based on events entail and how can they help the situation? For instance, consider a scenario, a regular user logs onto their laptop, checks their inbox and finds a mail with subject, *"An amount of $50,000 has been debited from the card ending with 6789",* with the body of the mail containing a link saying, *"Click on the link here if it was not done by you or if you think it's a mistake"*. The user panicking clicks the link immediately, triggering the download of an excel sheet named "Account Transactions.xlsx". Curious to see the transaction details, the user opens the excel sheet. And within a few seconds, the fans of the system seem to be running high and loud, and the screen is frozen and when the fans have cooled down, the users find all of their files having

an unfamiliar suffix to the extension and cannot open or read any of them. They also find a note in every folder / directory asking to pay a ransom to unlock the files.

This is a typical scenario of a ransomware attack through email phishing. Here the sequence of events would be -
logging in > opening mail > clicking on the link > file being downloaded > opening file > system getting infected.
Since ransomware is very pompous and leaves a huge, very discernible trail, making itself known to the users, the sequence of events can be identified, and the incident can be very clearly mapped as to how it happened for an isolated incident. Seeing the logs is not even required. But say if this happens in an organization with thousands of workers and thousands of systems connected to the network, it is going to be very hard to find out details such as

- what happened,
- what was the entry point of the malware,
- who was the user responsible (knowingly or unknowingly),
- what type of malware is it,
- what all systems, applications have been infected,
- Is there anything left to salvage, etc.

This is where audit logs or audit trails come into the picture. They would contain all the answers to the above question and more. Let's assume that based on the logging configuration setup, 100 logs are generated per system every 30 seconds. That would be 200 logs every minute from a system and if there were 1000 systems in the network being monitored, that would be 20000 logs every minute. Combing through each and every single is not possible even with numerous filters. Here, if the sequence of occurrence of events (file open, command execution, file read etc) are mapped and visualized, the entire incident can be understood on a high-level and an aerial view is obtained. Combining that along with statistics and metrics obtained from those systems and applying process of elimination and analysis, an incident reporter or an analyst would be able to not just identify cause of trouble, but can narrow it down to the individual, the system and even the time in seconds. Such is the benefit of an event-oriented visualization of audit logs.

The major challenge with audit logs or any logs for that matter is the volume and complexity of the data and handling them rightly to gain insight into incidents and events. Other challenges would be in collecting, storing, etc, but we will not be delving into those.

In this project, open-source observability, and security monitoring platforms such as Elastic Stack [7] and Grafana [8] are explored, but focused on Elastic Stack, to query and visualize audit logs and provide a low and high level view of the logs. Logs offer both low-level and high-level information. Often low-level information is easily seen due to them being fields (key: value pairs). The high-level information, such as events, behaviors etc. are hard to see. The low-level view is accomplished by use of statistical method with the help of kibana dashboards and the high-level view is accomplished through events in the logs with the help of a custom-made plugin. Additionally, as extensions, parsing and processing logs using ShadeWatcher's parsers [6] to extract behaviors and visualizing it using Neo4j's graph database [9] is done to obtain graphs and subgraphs of the behaviors. The aim of the project is to make analysis and the analysts' tasks easier.

2. BACKGROUND

2.1. Logs

Logs are records of events that have occurred in a system. Each event generates a log record which typically contains information ranging from time to the commands, arguments, files etc.

An example log record from a Linux system collected through Auditbeat is as follows:


```
{"@timestamp":"2020-10-31T14:14:47.785Z","@metadata":{"beat":"auditbeat",
"type":"doc","version":"6.8.12"},"process":{"ppid":"18113","name":"uname","
exe":"/bin/uname","cwd":"/","pid":"18115"},"file":{"path":"/lib/x86_64-
linux-gnu/libc.so.6","device":"00:00","inode":"18874451","mode":"0755",
"uid”:"0","gid":"0","owner":"root","group":"root"},"auditd":{"session":"705
","data":{"a2":"7fd1daaf6168","a1":"80000","arch":"x86_64","syscall":"open"
,"a3":"362e6f732e6362","a0":"7fd1daaf6d60","tty":"(none)","exit":"3"},"path
s":[{"ogid":"0","ouid":"0","rdev":"00:00","cap_fe":"0","cap_fi":"0000000000
000000","cap_fp":"0000000000000000","cap_fver":"0","nametype":"NORMAL","dev
":"08:02","inode":"18874451","item":"0","mode":"0100755","name":"/lib/x86_6
4-linux-gnu/libc.so.6"}],"sequence":166849,"result":"success"},"user":{
"egid":"0","name_map":{"fsgid":"root","fsuid":"root","egid":"root","euid":"
root","sgid":"root","suid":"root","uid":"root","auid":"anonymized","gid":"r
oot"},"auid":"1000","fsgid":"0","suid":"0","fsuid":"0","gid":"0","sgid":"0"
,"uid":"0","euid":"0"}}
```


From seeing this, we can see how comprehensive this is. Even though this is human-readable, having tens of thousands of such huge data is going to be hard to keep track of, view and analyze.

2.2. Log Formats and Types

On a layman level, logs are nothing but information and any information can be structured or unstructured or semi-structured. Examples for the same are as follows:

Structured:

```
{"timestamp":"2023-10-30T23:18:23+08:00","level":"INFO","message":"Web
server received a request for /index.html"}
```

Unstructured:

```
Web server received a request for /index.html
```

Semi-Structured:

```
[Mon Oct 30 23:18:23 SGT 2023] [INFO] Web server received a request for
/index.html
```

Apart from the classification based on how the log is formatted, logs can be collected / obtained from various sources for various purposes.

Log Type	**Description**
System Logs	Records **system-level events and errors**, including operating system events and hardware-related issues. Examples include syslog, Windows Event Logs.
Application Logs	Captures events and activities related to a **specific software application**. These logs help track application performance and errors.
Security Logs	Focuses on **security-related events and activities**, such as login attempts, access control, and suspicious activities. Examples include audit logs and authentication logs.
Web Server Logs	Tracks **requests and interactions with a web server**, including IP addresses, URLs, HTTP status codes, and user agents. Commonly used in web analytics.
Database Logs	Records **database activities**, such as queries, transactions, and errors. Used for performance monitoring and debugging.
Network Logs	Monitors **network traffic**, capturing information on connections, bandwidth usage, and potential security threats. Includes firewall and intrusion detection logs.
Error Logs	Specifically focuses on **error messages**, **stack traces, and debugging information**. Helps in identifying and resolving software issues.
Event Logs	Records various **events and occurrences** in a chronological order, providing a historical view of system or application behavior.
Access Logs	Monitors **access to resources**, such as files, directories, or services, and logs who accessed them, when, and from where.
Performance Logs	Collects **data related to system or application performance**, including CPU usage, memory usage, and response times.

Table 2.2: Types of Logs.

2.3. Log Management & Processes

The entire pipeline may be a long winding one involving multiple stages, but nonetheless, every stage is significant when it comes to analysis & threat hunting.

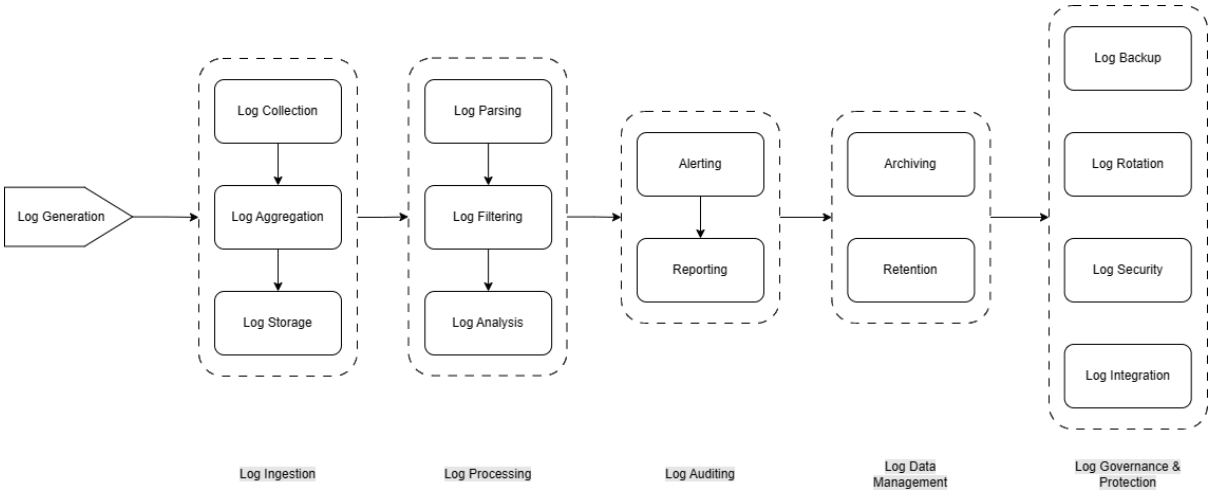

Figure 2.3. Log Processes.

Log Process	**Description**
Log Generation	The process of **creating log entries** to record events, activities, and errors. Logs are generated by various components, including applications, operating systems, and devices.
Log Collection	Involves gathering log data from different sources, such as servers, network devices, and applications. This data is typically collected centrally for **analysis and monitoring.**
Log Aggregation	Combines log data from multiple sources into a single repository or system for **centralized management**. Aggregation simplifies log **analysis and correlation**.
Log Storage	Involves storing log data efficiently and securely, either in flat files, databases, or log management systems. Proper storage ensures **data integrity and retention**.
Log Parsing	The process of **breaking down log entries** into structured or semi-structured formats, making it easier to extract meaningful information from logs.
Log Filtering	Involves the application of filters or rules to **select relevant log entries and exclude noise**. Filtering reduces the volume of data for analysis.

Log Analysis	Analyzing log data to identify **patterns, trends, anomalies, and critical events**. Log analysis helps in **troubleshooting, security monitoring and performance tuning**.
Alerting	Generating alerts or notifications when specific log events meet predefined criteria, such as **security breaches or system errors**.
Reporting	Creating reports and visualizations based on log data for **compliance, auditing, and performance tracking**.
Archiving	Moving log data to long-term storage for **historical reference and compliance** purposes while freeing up space in active storage.
Retention	Defining and enforcing rules for how long log data should be kept, considering **regulatory requirements and operational needs**.
Log Backup	Regularly creating backups of log data to prevent data loss in case of **system failures or data corruption**.
Log Rotation	Managing the size of log files by periodically **archiving or deleting older log data**, ensuring log files don't become too large.
Log Security	Implementing measures to protect log data from **unauthorized access, tampering, or deletion**. This includes **access control and encryption**.
Log Integration	Integrating log data with other tools and systems, such as **SIEM** (Security Information and Event Management) solutions or **incident response platforms**.

Table 2.3. Log Processes.

2.4. Log Analysis & Visualization

There are many ways logs can be analyzed based on the need & requirement. Some are:

1. Pattern Recognition: One of the most common ways to analyze logs, where it involves looking for recurring patterns of events / incidents in the log data.
2. Signature / Anomaly Detection: With the help of AI and Machine Learning, deviations from regular behavior are found which may or may not be malicious always.

3. Correlation / Behavior Analysis: There is usually a specific or a sequential behavior to events and this correlation can be identified to infer the bigger picture.
4. Search and Querying: One of the simplest and easiest form of analysis, but hardest and most time-consuming methods if done without additional tools and parsers.
5. Time-Series Analysis: This involves plotting and tracking the changes occurring with time and analyzing the trends.
6. Statistical Analysis: Usually combined along with Search & Querying and Time-Series, this is a method to understand trends, usage metrics etc.
7. Regression Analysis: A backtracking method which works causality relations.

That said, for the visualization part, it usually is the output of the analysis methods put in a diagrammatical / graphical way.

For instance,

- Time-series graphs can be used to display and track the time-series based analysis.
- Node graphs and provenance graphs can be particularly useful to display correlations and behaviors.
- Line graphs, bar graphs, pie charts, gauges, metric displays, etc, can be used to display results obtained from statistical analysis.

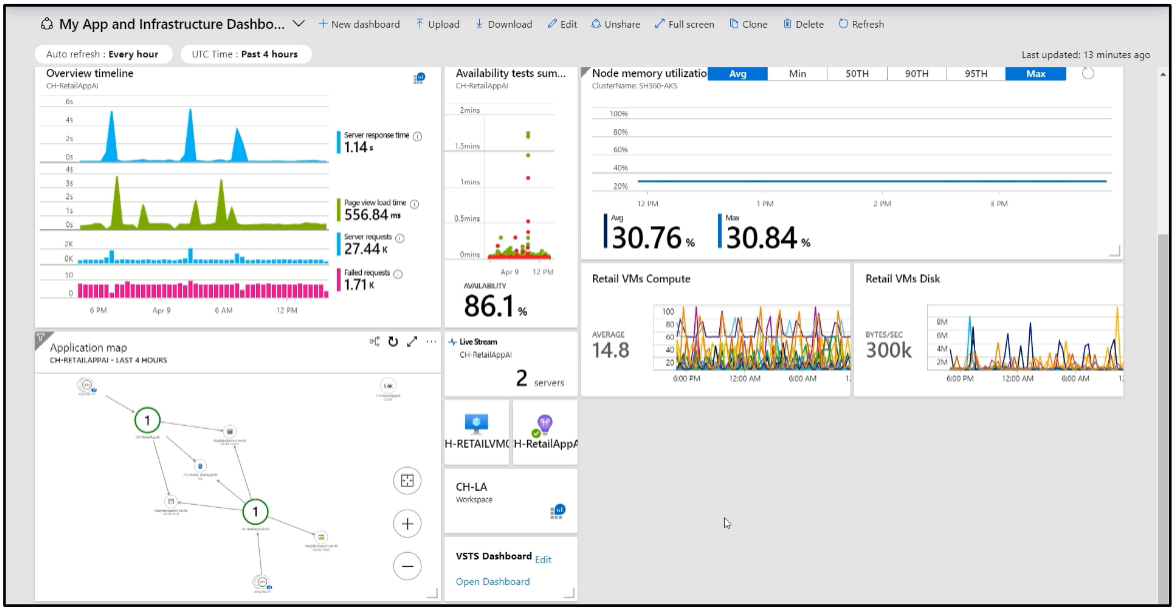

Figure 2.4. An example dashboard with results from Statistical Analysis

2.5. Audit Logs

Commonly also known as ‘audit trails’, are a separate category of logs which contain records of actions, event etc, inside a system / server or an application. These aren’t general purpose logs. They are collected, used, and maintained for security activities and compliance.

They are used predominantly for the following purposes:

1. Forensic Analysis: These are digital forensics, for legal and forensics investigations, to gather evidence by tracing events and actions.
2. Incident Investigation: Here it is used to understand how an incident occurred, affected systems and applications, the endpoints, data breaches etc. This later could be used to troubleshoot, or in recovery & response activities.
3. Security Monitoring: To track and monitor activities happening in an organization and filter out potentially suspicious activities. This could be ranging from simple logins to file uploads/ downloads, emails etc.
4. Compliance and Accountability: When it comes to an organization, some operations are a “must-be-done” type. This may be for internal audits or to comply with regulators or to withhold the principle of accountability in the organization.

Some common types of audit logs may be:

1. Security Audit logs: Records security related events such as logons/offs, access to files, change in permissions etc.
2. Application Audit logs: Records application errors, user interactions, application-to-application interactions etc.
3. Database Audit logs: Records queries executed, changes like insertion, deletion, updations to databases, access to databases etc.
4. Operating System Audit logs: Records logons/offs, applications used, new installations and uninstallations, general system monitoring etc.
5. Network Audit logs: Records network related activities such as traffic, connections, ports, protocols, firewall stats, etc.

.

3. TECHNIQUE AND IMPLEMENTATION

The focus and aim of this project is to produce a pipeline / methodology of visualizing log data as best as possible, on different levels, to gain enough insight and make the process of analyzing and threat hunting easier to deal with.

The project workflow tasks can be split into three paths. This can be seen below.

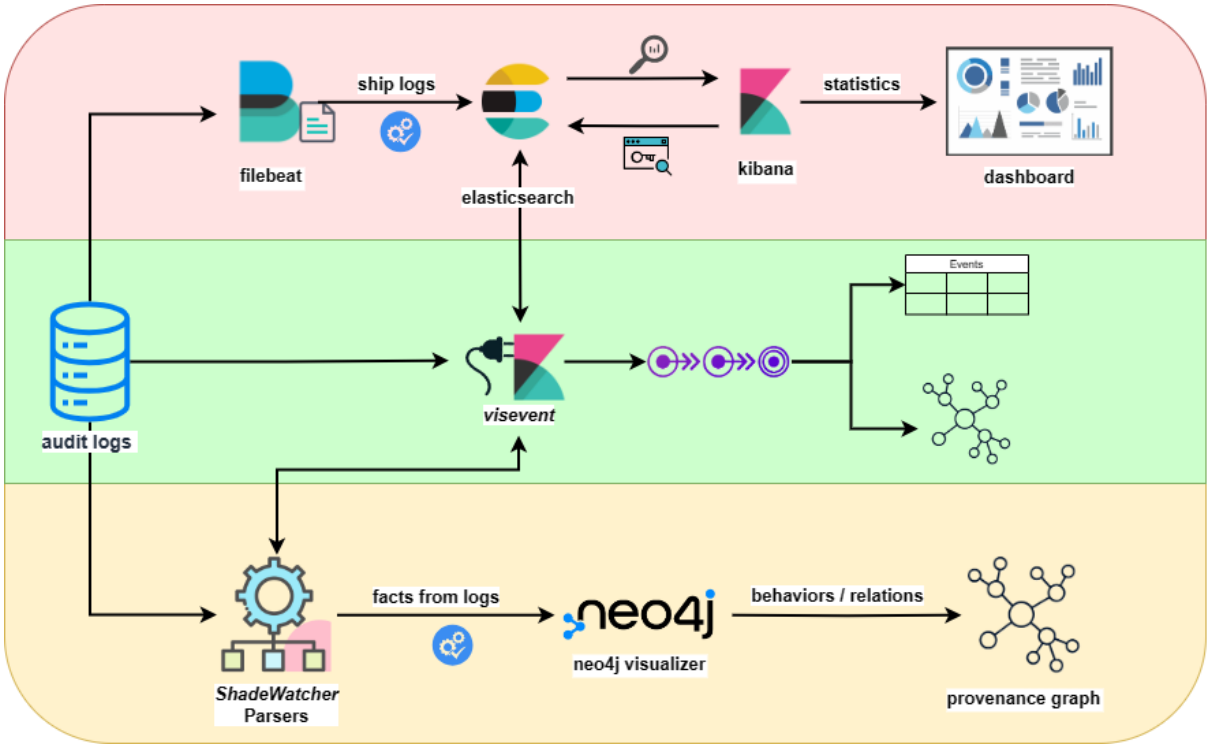

Figure 3.1. Project Pipeline.

The green path would be the primary work of this project with the red and yellow being extensions.

Audit logs are brought into Elastic stack with the help of the inbuilt file uploader and filebeat processing it through a pipeline(commands) to clean the logs of unwanted meta field. The way elastic deals with data is by creating what is called ‘index’ or ‘index pattern’ which can commonly be considered as templates containing field that the data can be mapped onto for query, search and visualization. Additionally, elastic also creates what is known as ‘data views’ which is used to determine what needs to be displayed with respect to the log data. Once the data is ingested into the service, we can then use the ‘visevent’ plugin that could be found in

the elastic menu to perform our visualization. The working of the plugin is elaborated in its own section below.

As for the other paths, the red path deals with the complete Elastic Stack, involving shipping log files through filebeat, querying, and searching using Elastic Search and displaying dashboards using Kibana. All of the above steps described for the plugin can be said for this path too with a slight change in the end stage. Instead of feeding into the plugin made and using it to visualize events, kibana dashboard is used to visualize static metric data. Kibana itself can be considered as a plugin, a core plugin.

Finally, the yellow path deals with using ShadeWatcher parsers to extract significant facts from the logs and using Neo4j Graph Database to visualize provenance graphs. Detailed process of this is explained in detail in the upcoming sections.

As mentioned in the introduction previously, the choice of the observability platform was between Elastic Stack and Grafana, both open-source and providing almost similar ways to handle, view and visualize logs. Ultimately Elastic Stack was chosen to continue with due to the prevalence and preference of it among security professionals and documentations available. That said, this shouldn't be considered as a demerit for Grafana, since it definitely is also an excellent option to use as observability and security platform providing numerous other facilities and plugin support.

One of the biggest, most time-consuming aspects of this project was the data itself. Even though the dataset is of structured data, 70% of the time and effort was spent on getting the data format right to use with the individual applications.

All the codes, scripts, and the entire plugin itself can be found in the github repository [23].

3.1. Elastic Stack

Elastic Stack, also known as ELK Stack (for **E**lasticSearch, **L**ogstash, **K**ibana) is a collection of open-source tool suite often used for querying, analysis, visualization of data. This suite has a wide range of use cases such as being a SIEM, application performance monitoring, log and event data monitoring and analysis, infrastructure monitoring, business analytics etc.

As already in the name ‘ELK’, the main components making up the stack / suite are:

- ElasticSearch [12], which is a powerful and efficient RESTful search and analytics engine capable of handling large volumes of data and providing real-time results for structured, semi-structured and unstructured data.
- Logstash [13], though not used for the project, is a powerful data ingestion agent capable of parsing, transforming and filtering data before it reaches elastic search engine and kibana. This can accept data from various, multiple sources with different formats making it very convenient to ingest data.
- Kibana [14], a very powerful, dynamic data visualizer capable of making the job of monitoring and visualizing easy through its intuitive web-based user-interface with graphs, charts, reports etc. This has support for multiple plugins from various different applications to integrate with.
- Beats [15], is a family of eight ‘lightweight’ data shippers to ship data such as windows event log, linux audit logs, system metric data, network packet data, log files, etc. For this project, since we already have the log data as files, Filebeat [16], which is used to ship log files, is predominantly used.

3.1.1. Setup

With the help of thorough documentations for all components and an active community, the setup of the ELK stack is very customizable and simple.

One thing to note here is that, for a production environment such as a company or organization, even though this is all open-source, high security, protection, and access control is present for the entire stack.

Each of the components comes with its own configuration file in yaml format, with all the configuration options available that can be included or excluded based on the requirements.

The ELK stack can be setup in a docker environment or a dedicated server environment, but for the project purpose and simplicity sake, the Elasticsearch and kibana (along with filebeat) has been setup in the local machine (running Windows 11), accessed thru localhost (ports 9200 and 5601).

As for the plugin itself, since it requires a development environment, Kibana v8.10.2 and v8.10.4 were forked and cloned from the official repository and setup in windows subsystem for linux (wsl) on win 11 system. All the development and building took place inside in the wsl.

Dependencies and packages used were:

1. Node.js - v18.17.2 and v18.18.9
2. Html2canvas – v1.4.1
3. Vis-network – v9.1.8
4. Vis-data – v7.1.7

3.1.2. Kibana Plugin – 'visevent'

The plugin is a very simple application built entirely client-side with the help of React and Elastic's own libraries. Additional external dependencies have already been mentioned above.

It had two components, 'app.tsx' and 'Graph.tsx', which are used for program logic and functionality.

Due to time, resource and knowledge constraints, many complex activities have been actively avoided but leaving room for ample updations and improvements. Currently, the max number of nodes and edges and number of log records to use have been preset due to the huge data set and limited resources. This limitation can be removed paving way to show the complete picture.

The way this plugin works is by fetching specific significant field values such as file, process_id, process_arguments, file path, user, process_result, unique id, syscall, etc, from the data and displayed as a database table. These individual pieces of information are then mapped to one another to create a sequence of events. This is then rendered as a graph using vis-network, vis-data and react. Additionally, to show the same event in the form verbal form, another events table is displayed showing the event sequence.

3.1.3. Challenges

Getting the dependencies right is the biggest challenge. Since newer Kibana versions don't offer backward compatibility due to the high rate of change, it is very imperative to get things right and working for each version.

One other major problem was with using the plugin itself. Once the production build was complete and ready and even installed, the entire Elastic system didn't start.

Figure 3.2. Challenges

Elastic and Kibana have a very strict policy with respect to Content-Security-Policy. So, finding out ways to bypass it took some time and research. Additionally, the handling the *ERR_CONTENT_DECODING_FAILED* was very random and tedious. Deleting the plugin's '.br' (Brotli Compressed files) helped solve the issue.

4. EXTENSIONS

To expand the project's vision and to offer better ways to perform analysis, ShadeWatcher's parsers were used along with Neo4j Graph Database to visualize the provenance graphs.

4.1. Neo4j

With any data, to store and retrieve it, typically an RDBMS (relational database management system), such as MySQL, PostgreSQL, etc, are used. These are then connected to dashboards, SIEMs and other platforms which are capable of handling and processing data.

An RDBMS uses a structured tabular model to store and access data. Relationships between different tables are made using the concept of foreign keys. This may just be sufficient in some cases but not the best option if you want to explore and drill down on the relationships more.

A graph database is a type of database which uses graph data models unlike RDBMS to store and access the data. The data themselves are represented as 'nodes', and the relationships between them are represented as 'edges'. This presents a well-connected network of nodes and edges which makes it easier to see the bigger picture.

Neo4j [9] is a prevalent, powerful, open-source graph database management system capable of storing, managing, visualizing, and querying the network of data through graphs. Just like any DBMS using a query language to interact with the data, Neo4j has its own query language called the Cypher Query Language [21], which is intuitive and quite similar to the well-known structured query language (SQL) additionally with its own grammar.

4.1.1. Setup

Neo4j can be set up as web+terminal-client interface or a full-fledged GUI – Neo4j Desktop.

For the ease of usage, Neo4j Desktop is used for the project and installed on the same local system as Elastic suite and accessed through Neo4j browser with the help of its native 'bolt' protocol running on localhost (port 7687).

Since the desktop application is a complete package by itself, it isn't a big hassle to set it up. Neo4j allows a connection to an RDBMS along with relations which can be graphically plotted. But for the project, the mode of data is NDJSON files. More about it is addressed in the next section.

Note: For informational purposes and to provide a full picture, working python script and SQL queries to create database, table and to import audit logs data into MySQL is provided. This can be connected to Neo4j and can be processed. In a real-time environment, uploading files may not be feasible or sufficient.

4.1.2. Data

For this part of the project, the structured but raw data cannot be given to the graph database. Hence the trace data is parsed with the help of ShadeWatcher's [6] *'driverbeat'* parser, which is a specially made parser competent in handling audit logs. One of the goals of the ShadeWatcher project, as per official documentation, is to extract behaviors and identify anomalies from audit logs.

The parsed output of the log data is aggregated in five different "fact files", namely:

- edgefact – containing edge details *(edge_id, node1_hash, node2_hash, relationship, sequence_number, session, and timestamp)* between two nodes / entities of the data.
- procfact – containing process details *(id, process_id, executable, parent_process_id, and arguments)* of a node.
- sockfact – containing socket details *(id, name/ip_address)* of the node.
- filefact – containing file details (id, name_of_file, version (if any)) of the node.
- nodefact – containing node details (id, type_of_node).

The parser is programmed to identify and map 28 relations between the nodes. A few from the list are *'delete', 'pipe', 'clone', 'read'* etc. Similarly, it is programmed to map the node to three types – *'file', 'process', and 'socket'.*

The fact file obtained from the parser were observed to semi-structured data in the form of text files with each line containing a record separated by spaces. This had to be converted into a structured format and the chosen format was NDJSON. Additionally, the relations between the nodes and socket types were as numbers and not descriptive. This was done to train the machine learning model for the ShadeWatcher recommendation system.

Hence, parsing through the fact files, the replacement of the appropriate value for the relation and socket types had to be done. It was also observed that the fact files contained duplicate entries which probably is from the log data itself. So, deduplication had to be done to further refine it.

Once the data (procfact, sockfact, filefact) is processed and prepared, it can be loaded into the graph database with the help of APOC library functions [22], which is yet another open-source

project under the Neo4j community. At this stage, the information from the fact files are represented only as distinct nodes. With the edge information from the edgefact file, the APOC library function, and cypher queries the relationships are established between the nodes.

4.2. ShadeWatcher

With research and development very much still actively with the ShadeWatcher project, keeping the setup environment up-to-date and subsequently updating the documentation is imperative.

The original version of the project has been run with the setup environment of Ubuntu 16.04.3 LTS and numerous libraries and dependencies previously used have been deprecated since and gone obsolete.

As a small contribution to the ShadeWatcher project, the entire parser setup is replicated on Ubuntu 18.04.3 LTS and Ubuntu 20.04.4 distros with updated libraries and dependencies.

As mentioned, the setup requires many installations and previously, they were separate chunks of instructions to be executed one after the other. To save time and effort and in a way automate the process by only requiring very minimal interaction, a one-shot shell script has been created to complete the installation of all libraries and dependencies and compile the parsers fully.

After this, the user can just run their audit log file through the parsers, with the required arguments to receive the fact files previously addressed. Additionally, the user can open neo4j browser and see the graph database, its nodes, relations etc.

5. RESULTS

As has been mentioned many times previously, the plugin has a simple user interface. It has 3 components in total, a table displaying fields and its values, a graph and an events table which is a verbal form the graph.

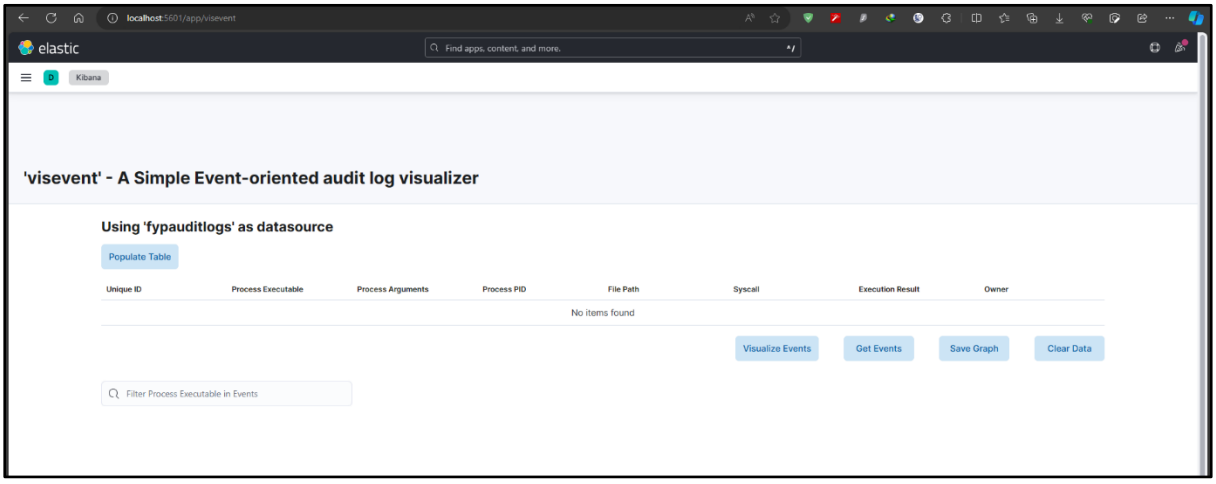

Figure 5.1. Plugin - General View

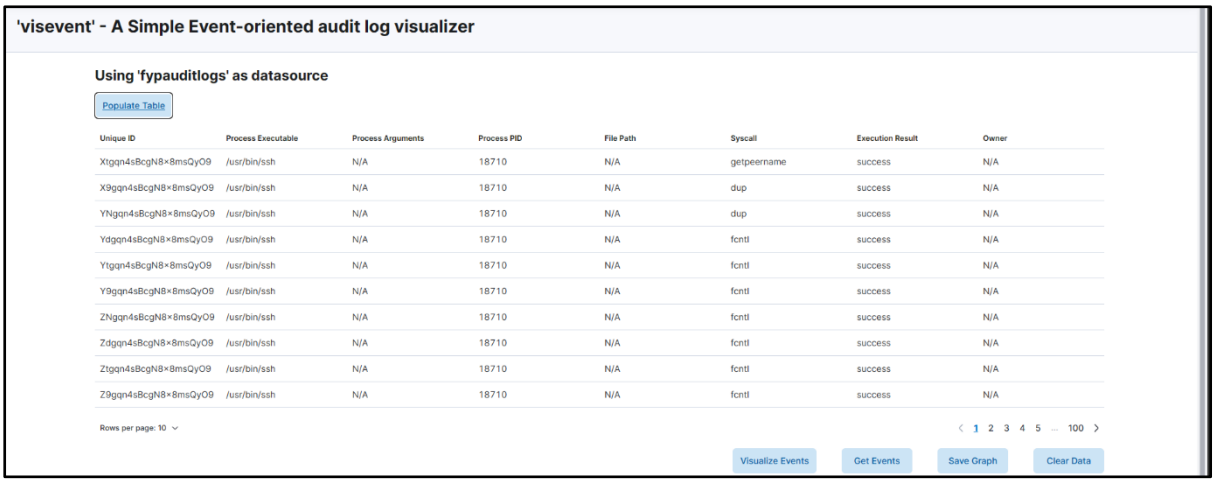

Figure 5.2. Plugin - Field Populated Table

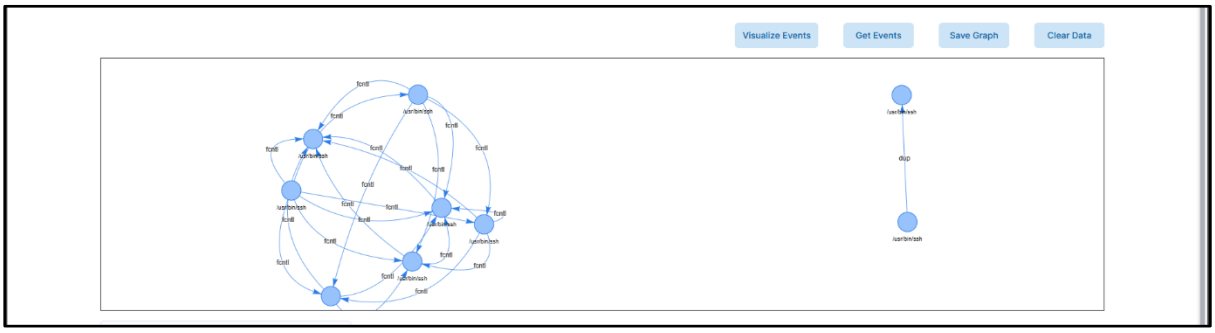

Figure 5.3. Plugin – Interactive Graph

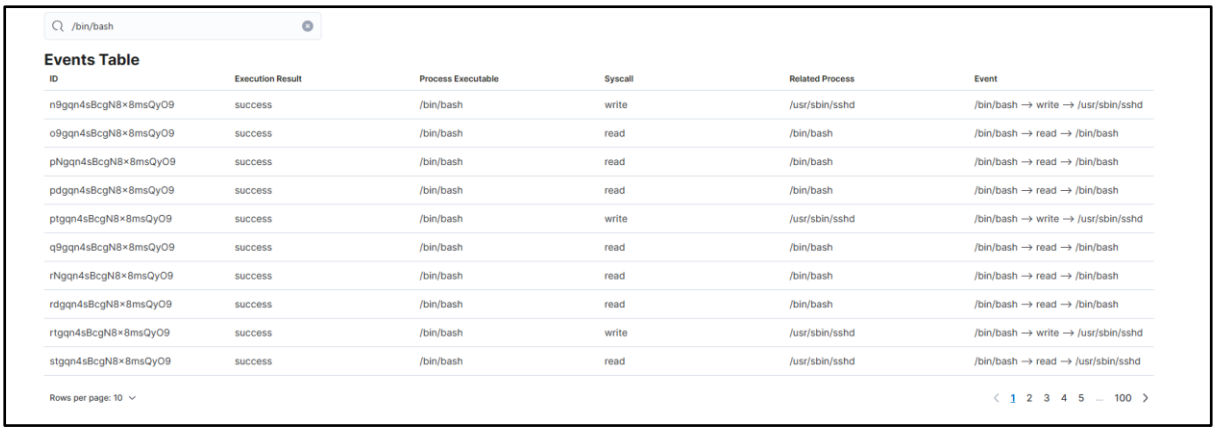

Figure 5.4. Plugin – Event's Table filtering '/bin/bash'

Using ELK suite was to gain a statistical insight into the audit logs with the help of the visualization techniques offered by the suite. In a typical live-production environment, the same would be constantly accepting data, refreshing, and updating the visuals in real-time based on the events occurring.

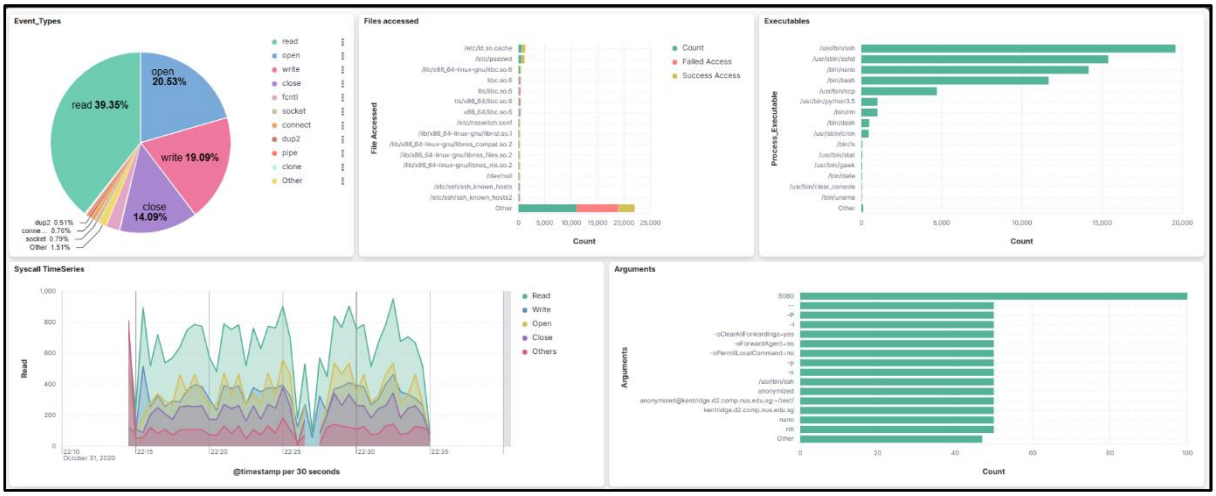

Figure 5.5. Kibana Dashboard

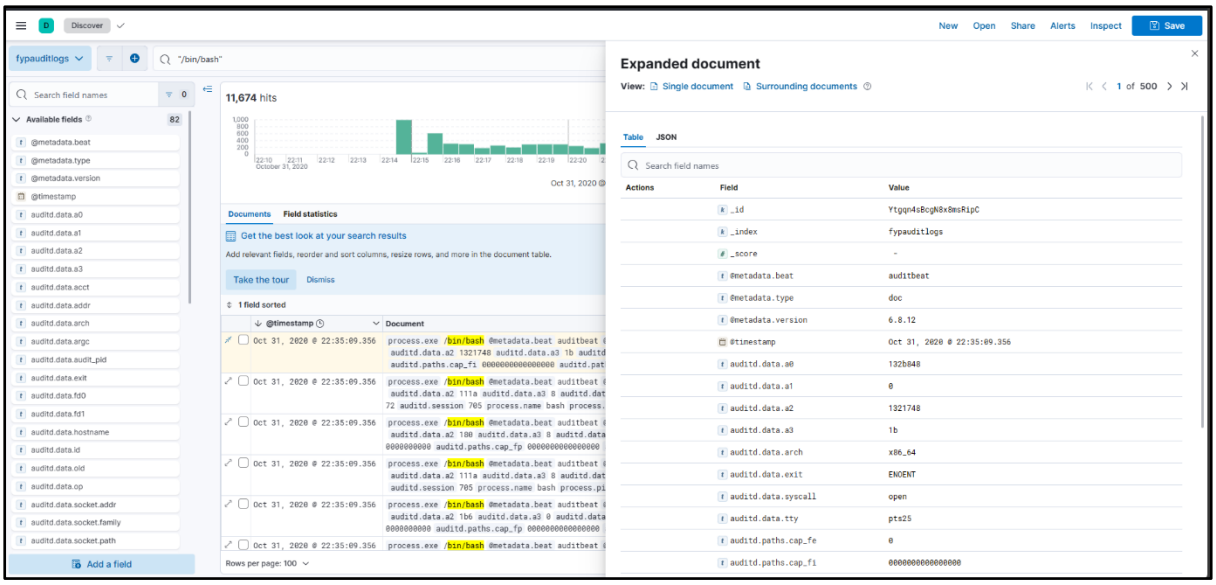

Figure 5.6. Elasticsearch Query Result

The results from Neo4j are extremely well-connected provenance graph. Each node can be then expanded into subgraphs if there exist deeper relationships which can be seen below.

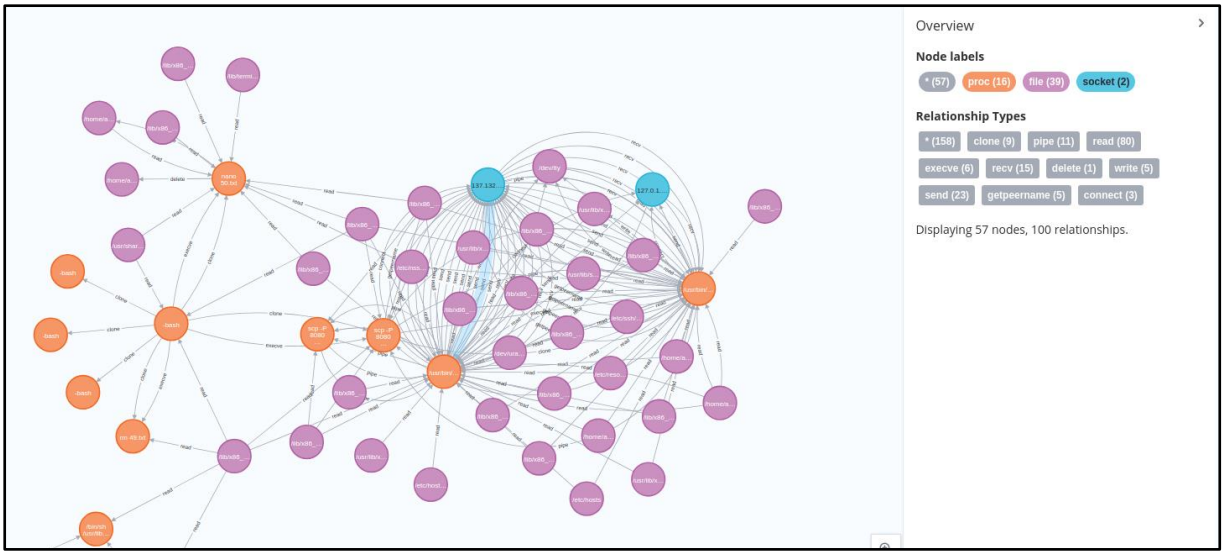

Figure 5.7. Provenance Graph

6. FUTURE WORK

The plugin has a lot of room for improvement like handling requests other plugins, spicing up the visuals, minimizing code, or even developing as a standalone plugin to be used with any application, and not just Elastic Suite and Kibana. A feature that could be added would be select a single node and forward-track that node alone till there is no other leaves. The current shows all connections.

Since all the platforms and applications used in this project are open-source, with an active development community, extensive documentation and guides, custom plugins can be made and used along with these platforms integrating other security applications per our use case and needs.

With an abundance of data, it is essential to have powerful parsers, filters and other tools which can process the data properly, segregate and extract the core information from the data dump.

With the help of machine learning, large-language-models and natural language processing, a more comprehensive, powerful and a unified platform can be built to handle querying, analysis, visualization etc. The use case of such a unified platform will not be limited only to the field of security and can be used in other sectors such as healthcare, finance, entertainment etc.

7. CONCLUSION

Though the project is meant to predominantly focus on visualization and analysis, much of the effort has been invested in understanding the audit logs, the structure, the data present etc – ie: in other terms, data processing and analytics, as is common with most projects. It can be safely said that the aim of providing multiple levels of visualization has been satisfied, but not exhausted.

Open-source applications and the aid of active development communities' pave way to more opportunities for newer, better innovations and improvements.

Though there are several vendors providing several products and services, open-source and paid, there seems to be a standard format for handling the immense data that is being observed and collected which can be followed independent of the platform, source, or vendor. This creates a major issue, often very tedious and time-consuming since the data must be processed for each application separately. Hopefully, with all the traction security has gained in recent years, a proper standard is set that can be adopted irrespective of everything.

The hope and goal of taking up this project is that this serves as a passage to deeper explorations, better understanding, and application of knowledge for the betterment of the self and future career prospects.

The field of security is a treasure trove of scope, both for research and career, waiting to be seized. With a lot more attention, importance and mainly funding given to security in the current times, the need and the opportunities that come along with it are endless.

8. BIBLIOGRAPHY

[1] "Cost of a data breach 2023," https://www.ibm.com/reports/data-breach, 2023, online; Accessed 18 October 2023.

[2] "2023 SonicWall Cyber Threat Report," https://www.sonicwall.com/2023-mid-year-cyber-threat-report/, 2023, online; Accessed 18 October 2023.

[3] "DBIR Report 2023 – Summary of Findings," https://www.verizon.com/business/resources/reports/dbir/2023/summary-of-findings/, 2023, online; Accessed 18 October 2023.

[4] "2023 State of Malware Report: What the channel needs to know to stay ahead of threats," https://www.malwarebytes.com/blog/business/2023/04/top-5-cyberthreats-facing-msps-and-vars-in-2023, 2023, online; Accessed 18 October 2023.

[5] "Trace," https://drive.google.com/drive/folders/1EvIyZI7eCiON8xT8D8G4CENI_vXVshb2, [n.d.], online; Accessed 10 September 2023.

[6] J. Zeng et al., "SHADEWATCHER: Recommendation-guided Cyber Threat Analysis using System Audit Records," 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2022.

[7] "Elastic Stack: Elasticsearch, Kibana, Beats & Logstash," https://www.elastic.co/elastic-stack, 2023, online; Accessed 31 August 2023.

[8] "Grafana: The open observability platform," https://grafana.com/, 2023, online; Accessed 31 August 2023.

[9] "Neo4j Graph Database & Analytics," https://neo4j.com/, 2023, online; Accessed 18 August 2023.

[10] "Transparent Computing Engagement 3 Data Release," https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md, [n.d.], online; Accessed 10 September 2023.

[11] "Ground Truth," https://drive.google.com/drive/folders/1ATro9_PaoNlg376yA_moI1MbJGF-_HaV, [n.d.], online; Accessed on 10 September 2023.

[12] "Elasticsearch: The Official Distributed Search & Analytics Engine," https://www.elastic.co/elasticsearch/, 2023, online; Accessed 31 August 2023.

[13] "Logstash: Collect, Parse, Transform Logs," https://www.elastic.co/logstash, 2023, online; Accessed 31 August 2023.

[14] "Kibana: Explore, Visualize, Discover Data," https://www.elastic.co/kibana, 2023, online; Accessed 31 August 2023.

[15] "Beats: Data Shippers for Elasticsearch," https://www.elastic.co/beats, 2023, online; Accessed 31 August 2023.

[16] "Filebeat: Lightweight Log Analysis & Elasticsearch", https://www.elastic.co/beats/filebeat, 2023, online; Accessed on 31 August 2023.

[17] "Winlogbeat; Analyze Windows Event Logs," https://www.elastic.co/beats/winlogbeat, 2023, online; Accessed on 31 August 2023.

[18] "Auditbeat: Lightweight Shipper for Audit Data," https://www.elastic.co/beats/auditbeat, 2023, online; Accessed on 31 August 2023.

[19] "Elastic Common Schema," https://github.com/elastic/ecs, [n.d.], online; Accessed on 14 September 2023.

[20] "TCCDMDatum.avsc," https://drive.google.com/file/d/1oGMbSIZ73lh0xs8PUg--3Bj_wtC1q85-/view, [n.d.], online; Accessed 14 September 2023.

[21] "Cypher Query Language – Developer Guides," https://neo4j.com/developer/cypher/, [n.d.], online; Accessed 29 October 2023.

[22] "Awesome Procedures on Cypher (APOC)," https://neo4j.com/labs/apoc/, 2023, online; Accessed 29 October 2023.

[23] "Project_CodeBase," https://github.com/Srini-99/Project_CodeBase, 2023, online; Accessed 8 November 2023.