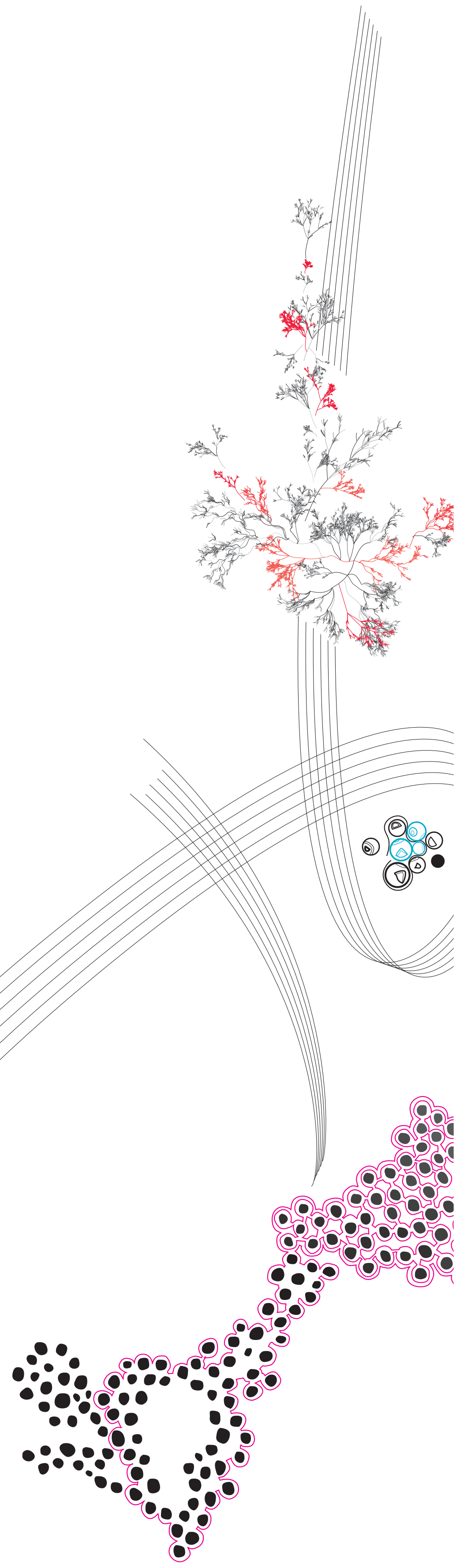


Design project TCS
Coursecode: 202001048

DeepCASE Dashboard

A detailed report regarding the DeepCASE Dashboard

Authors:
Group 1:
Beau Jonkhout,
Jan van Zwol,
Marijke van Iperen,
Katy Radzkova,
Kristupas Cepelis,
Lucian Trusca

Supervisors:
Dr. T. van Ede
Dr. A. Continella

April, 2024

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Preface

This section is dedicated to expressing special thanks of gratitude, for the ones that contributed to the project.

We would like to thank Dr. Thijs van Ede for his effort regarding the project. During the weekly meetings, we have received valuable feedback, positive reactions and generally an interesting time when talking about the project. Each week on the Tuesday at 16:00, we met to discuss general, specific and organisational issues regarding the development process. Each week, it felt rewarding to join the meeting, because Thijs always responded in a positive way, while still providing great feedback.

So again, thanks for all the effort.

Group 1

Dashboard Deepcase

Beau Jonkhout,
Jan van Zwol,
Marijke van Iperen,
Katy Radzkova,
Kristupas Cepelis,
Lucian Trusca [*]

April, 2024

Abstract

DeepCASE is a security tool that is used by security operators to monitor network systems for suspicious activity. It provides an automated way of analysing these security logs for similar events and can be used by security operators to filter for potentially harmful activity. This is done by using an artificial intelligence that determines which security events are correlated to each other. A dashboard is created that allows the users to interact with the system by scoring sequences in the manual phase and viewing the result of the semi-automatic phase. The main goal of this project is to create a dashboard showing the results of the DeepCASE security tool and providing security operators with a better understanding of potential attacker activities on monitored system.

Keywords: deepCASE, AI, security, dashboard

[*]Email:b.f.c.jonkhout@student.utwente.nl, j.vanzwol@student.utwente.nl, m.m.a.p.vaniperen@student.utwente.nl, k.cepelis@student.utwente.nl, k.radzkova@student.utwente.nl, l.trusca@student.utwente.nl

Contents

1 Introduction

The design report for the DeepCASE Dashboard project presents a comprehensive approach for the creation of a user-friendly dashboard that enhances the interaction between security operators and the DeepCASE module. By focusing on visualizing the decision-making processes of DeepCASE, the dashboard aims to provide security operators with a clearer understanding of attacker activity and the reasoning behind the alerts generated. Here, we will explore the journey to developing this solution, starting from the challenges faced by security operators and culminating in the innovative solution offered by DeepCASE and the added value of the dashboard.

1.1 Struggle of Security Operators

Security operators are tasked with the continuous monitoring of IT infrastructures to identify potential threats. Employing systems such as Network Security Monitors (NSM) and Intrusion Detection Systems (IDS), these operators sift through countless alerts to detect suspicious activities. However, the sheer volume of alerts generated can overwhelm even the most diligent teams, leading to a phenomenon known as alert fatigue. This occurs when the frequency and quantity of alerts cause important warnings to be overlooked or ignored, potentially allowing malicious activities to slip through undetected.

1.2 Core Problem and Solution with DeepCASE

At the heart of the issue is the fact that many of the detected events are not actually malicious, leading to a flood of unnecessary alerts that contribute to operator overload. DeepCASE addresses this problem by automating the correlation of security events through sophisticated contextual analysis. By understanding the context in which events occur, DeepCASE effectively filters out benign activities and isolates those events that truly indicate potential threats. This not only reduces the number of alerts that security operators must handle but also enhances the accuracy of threat detection, ensuring that significant alerts receive the attention they deserve.

1.3 The Proposed Dashboard

By streamlining the process of event analysis, DeepCASE significantly alleviates the workload on security operators, enabling them to focus on genuine threats without the distraction of irrelevant alerts. The proposed dashboard is designed to visually represent this contextual decision-making process, offer operators a way to interact with the module and easily see which events need special attention. Creating a GUI will increase the usability of DeepCASE and makes retrieving this information and completing these tasks easier.

1.4 Conclusion

In conclusion, the DeepCASE Dashboard project is not just about creating a tool for visualization; it is about transforming the way security operators interact with detection systems. Through this dashboard, operators can gain a deeper understanding of the system's analytics, leading to improved operational efficiency and enhanced security measures. The design report details the journey towards achieving this goal, outlining the challenges, the innovative solution provided by DeepCASE, and the impact of this solution on the day-to-day operations of security teams.

2 DeepCASE

In this section we will give a summary of the way DeepCASE functions.
DeepCASE [2] is a AI tool that is designed to filter security events within the network systems of a company that might need more attention from security operators. In short it does this by analyzing the security events and clustering them. Using these clusters security operators can see which events are similar to each other. If a new sequence is clustered with sequences that were already deemed suspicious or even malicious, that sequence might need more attention. This clustering functionality is what will save security operators a lot of time.

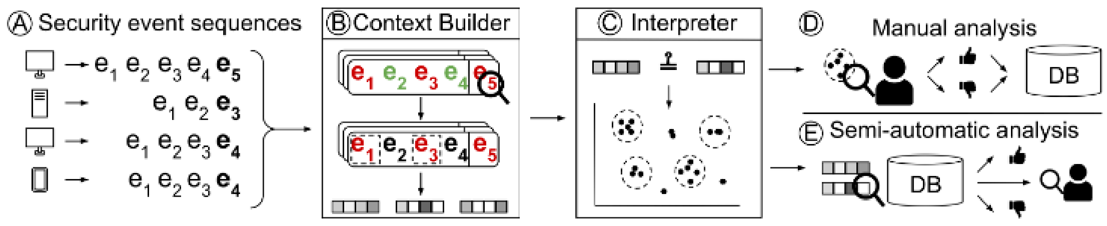


Figure 1: DeepCASE analysis flow

Figure 1 is a diagram that represents the flow of DeepCASE's analysis.

2.1 Sequences and the Context Builder

Every event happens within a context. Entering an incorrect password once is innocent enough. However, if it's the 100th time in ten seconds it becomes suspicious. That is why DeepCASE takes the context of an event into account. DeepCASE takes the nine previous events created by the same device and together with the current event this makes up a sequence. So when we refer to a sequence we mean a security event and the nine previous events created by the same device. This is part A in figure 1.
Not all events in a sequence are as closely correlated. To determine which events are most important in the sequence, the context builder is used. The context builder is an AI tool that analyzes sequence and tries to predict which event is going to be next in the sequence. If it predicts this correctly it will see which events were most important to make that prediction and then assumes these are also the most relevant to the sequence from a cybersecurity stand point. The importance of the events within the prediction are called attention vectors. If we refer to attention vectors we mean the degree to which an event is relevant to the sequence. This is part B in figure 1.

2.2 Clustering and the Interpreter

Once the sequences have been analysed by the context builder they are passed through to the interpreter. What happens here is that the sequences are put into clusters. It clusters the sequences based on the events contained in the sequence and their attention vectors. With this method the interpreter groups similar sequences together. These are called clusters. When we refer to clusters we mean a group of sequences that have been clustered together by the interpreter. This is part C in figure 1.

2.3 Risk labels and Manual Analysis

Once the sequences are clustered we still do not know which sequences require our attention. We only know which sequences are related. To accomplish this the security operator has to do some manual analysis. This means that the security operator has to look through a series of sequences and manually assign a risk label to each sequence. This risk label is a metric for how likely the sequences is to be malicious. This is part D in figure 1.

2.4 Semi-automatic Analysis

Once enough sequences are labelled DeepCASE can label sequences on it's own. It does this by making the assumption that if sequences are in the same cluster and thus similar, that should mean that their risk labels are also correlated. If a sequences is clustered together with sequences that were deemed as attacks it is probable that the sequence is also an attack. If there are sequences in a cluster with different risk labels there are multiple ways of determining what the default label for that cluster should be like: the minimum, maximum or median risk label value. This is part E in figure 1.

3 Requirements

In order to get a better sense of the requirements, we will start by providing the functional and non-functional requirements of the project. The requirements for the project are presented in the form of user stories and refer to Security Operator or Security Operations Center (SOC) as “user”, Group 1 of Design Project as “developer”, and the dashboard product as “ system”.

3.1 Functional Requirements

Must have

- As a user, I want to upload data in a format accepted by the DeepCASE module.
- As a user, I want to be able to train the DeepCASE module on the uploaded data.
- As a user, I want to be able to score individual events. (Manual analysis)
- As a user, I want the DeepCASE module to give a score to uploaded data (Semi-automatic analysis)
- As a user, I want the system to store the scores of the clusters.
- As a user I want to be able to inspect security event sequences. (Security event sequences)
- As a user I want to be able to see which events in the sequence are correlated. (Context builder)
- As a user, I want to be able to inspect individual events. (Interpreter)
- As a user, I want to be able to approve or disapprove the score given to a sequence. (Semi-automatic analysis)

Should have

- As a user, I want the results of the uploaded security files to be stored on the server and available for viewing by any user.
- As a user, I want the system to randomly suggest which sequences to rate during the manual phase.
- As a user, I want to have a basic manual integrated into the website.
- As a user, I want the system to visualise the clusters graphically.

Could have

- As a user, I want to be able to filter sequences with specific parameters.
  - The parameters could be time, IP address, type of events and risk values
- As a user, I want to have a basic guide on the website so that working with the new system is easy.
- As a user, I want the system to have regular backups.

- As a user, I want the system to suggest the most relevant sequences by an algorithm
  - The algorithm should be based on specific metrics, (e.g. greatest difference in vectors)

Will not have

- As a user, I want to connect it to real-time security loggers, so that the data can be analysed real-time.
- As a user, I want to receive notifications about newly detected malicious events.

3.2 Non-Functional Requirements

Underneath, you will find the requirements that are crucial for the general system's operability and have not been formulated yet in the functional requirements. Underneath you will find our defined non functional requirements:

Performance

- As a user, I want the dashboard and analysis to show results within a reasonable time frame (1 or 2 seconds).

Usability

- As a user, I want the dashboard to be understandable with minimal training.
- As a user, I want the dashboard to be intuitive.
- As a user, I want the dashboard to give me accurate and timely feedback when an error occurs.

Reliability

- As a user, I want the system to not have any substantial freezes.
- As a user, I want the system to be robust when against unexpected inputs.
- As a user, I want the data shown in the dashboard to be correct.

Scalability

- As a user, I want the dashboard's load time to remain static with the addition of data to the database.

Portability

- As a user, I want the system to be compatible with all major browsers.

Availability - How long is the average system downtime?

- As a user, I want the dashboard to be available 99

Maintainability

- As a developer, I want the code base to be maintainable.
- As a developer, I want the code to be commented on and documented.

4 Overall Process

In this section, we will reflect on the process and working dynamic in and outside the team.

4.1 Design Process

The design process of our project was iterative and dynamic, using Figma as our main design tool. Initially, we faced challenges with an early version of the design file that led to misinterpretations. However, these were small setbacks, but they were learning points that helped us in our approach.
Having weekly meetings, we established a consistent workflow that allowed a steady stream of design increments. This approach allowed us to introduce both minor adjustments and major revisions systematically. We updated the Figma file to suit the stakeholder's needs.

4.2 Coding Process

When implementing the figma design, we saw that the initial phase of our coding journey was marked by communication hurdles, leading to instances where coding best practices were overlooked. This occasionally resulted in frustration among team members, particularly when unexpected commits were made. Over the next few weeks, by embracing better coding practices such as strict adherence to abstraction, modularity, feature-branching, and establishing clear coding conventions, we saw a significant improvement. These enhancements fostered a more harmonious development environment, culminating in consistent commits and a smoother collaborative experience.

4.3 Cooperation and Collaboration

Our six-person team naturally divided into roles that played to each member's strengths — some gravitated towards organization, some focused intensively on coding, while others dedicated themselves to the nuances of styling. This organic structuring led to an efficient mode of operation.
Collaboration proved essential, especially when integrating distinct modules; real-time information exchange, facilitated by text communication on platforms like Discord or in-person meetings, helped our workflow. Tasks such as CSS implementation, user manual page additions are examples of collaboration. But tasks like presentation preparations, design report authoring, and testing, although too succinct for their own paragraphs, were integral to our project's success and are noteworthy contributions to our collaborative effort.

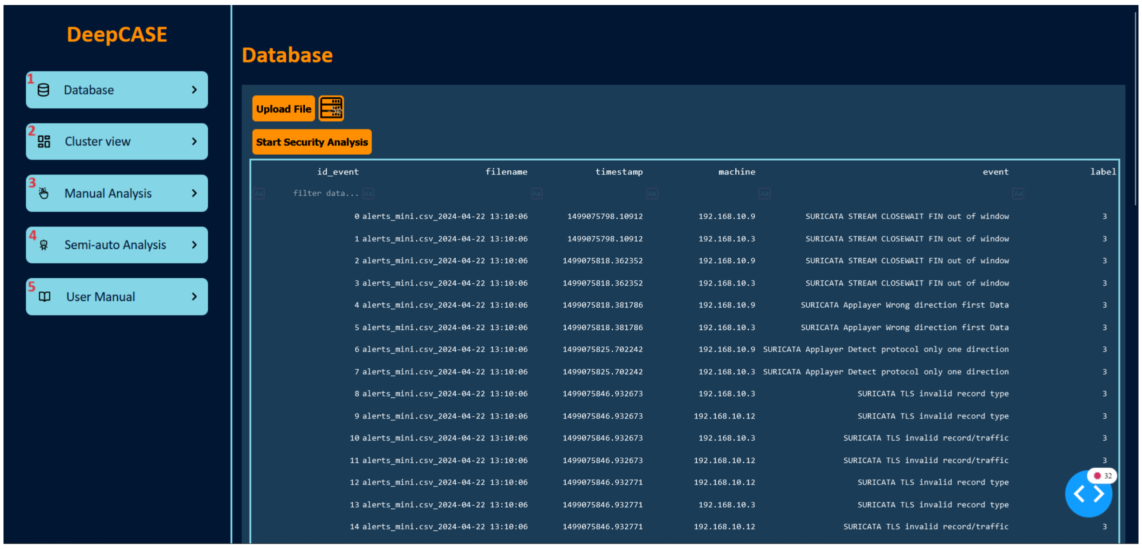
4.4 Conclusion

Reflecting on the entirety of our process, we went through various phases of growth and adaptation, both in design and development. Our journey from initial misinterpretations to a refined and efficient workflow highlights the value of iterative improvement and the power of collaboration. Through effective and consistent communication, having best practices, and leveraging individual strengths, we created a cohesive team dynamic that was essential in creating the dashboard.

5 UI design

In this section we will be discussing the design of the UI of the dashboard. In every subsection a screenshot of a different page will be shown. Every part part of the UI will have a description of it's function and a explanation on the decisions that let up to this design. The screenshots will include red numbers to easily refer to sections of the dashboard. These are of course not part of the actually dashboard.

5.1 Sidebar



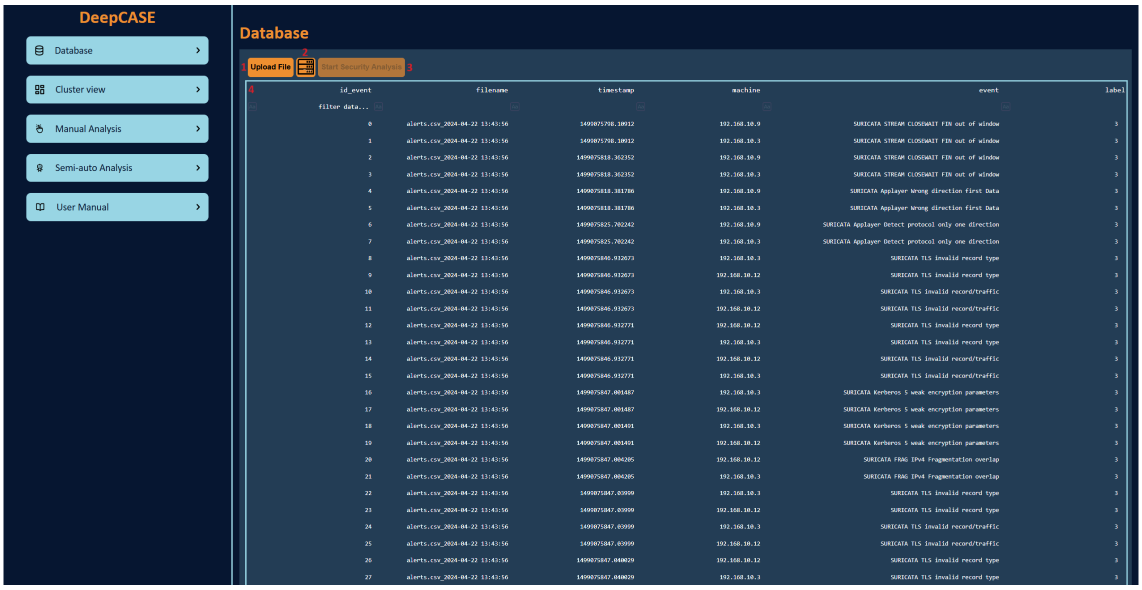
id_event	filename	timestamp	machine	event	label
0	alerts_mini.csv_2024-04-22 13:10:06	1499075798.10912	192.168.10.9	SURICATA STREAM CLOSEWAIT FIN out of window	3
1	alerts_mini.csv_2024-04-22 13:10:06	1499075798.10912	192.168.10.3	SURICATA STREAM CLOSEWAIT FIN out of window	3
2	alerts_mini.csv_2024-04-22 13:10:06	1499075818.362352	192.168.10.9	SURICATA STREAM CLOSEWAIT FIN out of window	3
3	alerts_mini.csv_2024-04-22 13:10:06	1499075818.362352	192.168.10.3	SURICATA STREAM CLOSEWAIT FIN out of window	3
4	alerts_mini.csv_2024-04-22 13:10:06	1499075818.381786	192.168.10.9	SURICATA Applayer Wrong direction first Data	3
5	alerts_mini.csv_2024-04-22 13:10:06	1499075818.381786	192.168.10.3	SURICATA Applayer Wrong direction first Data	3
6	alerts_mini.csv_2024-04-22 13:10:06	1499075825.702242	192.168.10.9	SURICATA Applayer Detect protocol only one direction	3
7	alerts_mini.csv_2024-04-22 13:10:06	1499075825.702242	192.168.10.3	SURICATA Applayer Detect protocol only one direction	3
8	alerts_mini.csv_2024-04-22 13:10:06	1499075846.932673	192.168.10.3	SURICATA TLS invalid record type	3
9	alerts_mini.csv_2024-04-22 13:10:06	1499075846.932673	192.168.10.12	SURICATA TLS invalid record type	3
10	alerts_mini.csv_2024-04-22 13:10:06	1499075846.932673	192.168.10.3	SURICATA TLS invalid record/traffic	3
11	alerts_mini.csv_2024-04-22 13:10:06	1499075846.932673	192.168.10.12	SURICATA TLS invalid record/traffic	3
12	alerts_mini.csv_2024-04-22 13:10:06	1499075846.932771	192.168.10.12	SURICATA TLS invalid record type	
13	alerts_mini.csv_2024-04-22 13:10:06	1499075846.932771	192.168.10.3	SURICATA TLS invalid record type	
14	alerts_mini.csv_2024-04-22 13:10:06	1499075846.932771	192.168.10.12	SURICATA TLS invalid record/traffic	3

FIGURE 2: Sidebar

The sidebar or navigation bar is the part on the left of the screen. This is used to open different pages of the dashboard. Clicking a button on this sidebar will open the corresponding page on the right of the screen. The sidebar itself is a static component both visually and functionally. So the sidebar will remain the same no matter on which page you are.
This design of a sidebar is very common in apps and websites. We tried to duplicate that design because almost every one will have encountered such a design and thus it will be familiar and intuitive to almost all users. We also included a visual representation of the page in all the buttons so that users can quickly find the correct button without having to read them all. Looking at symbols is generally quicker than reading words.

Elements:

1. Button that will open the database page
2. Button that will open the cluster view page
3. Button that will open the manual analysis page
4. Button that will open the semi-automatic analysis page
5. Button that will open the user manual page

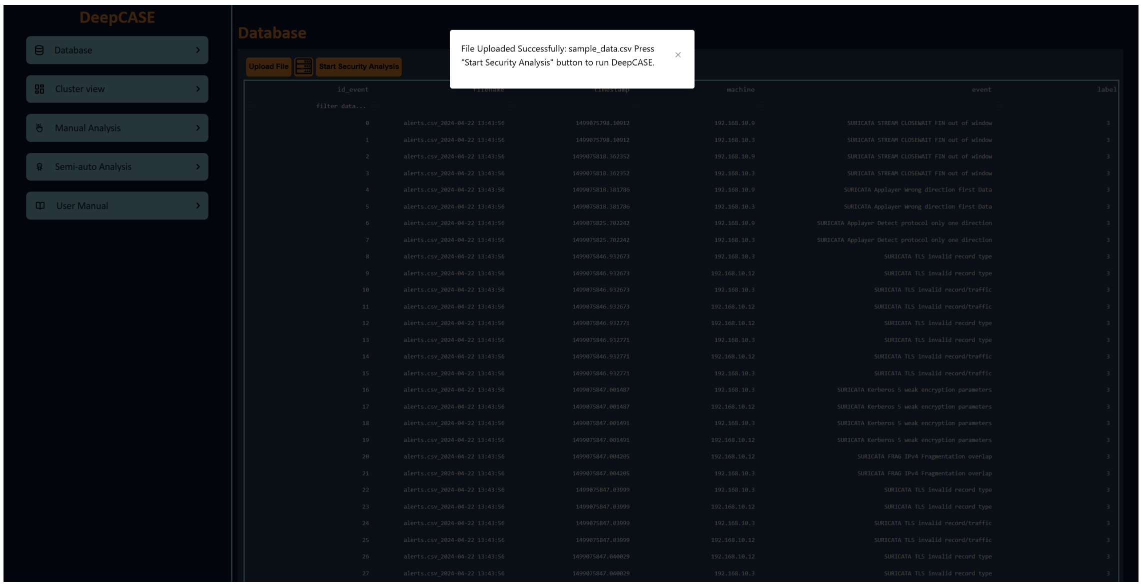


id_event	filename	timestamp	machine	event	label
0	alerts.csv_2024-04-22 13:43:56	1499075798.10912	192.168.10.9	SURICATA STREAM CLOSEWAIT FIN out of window	3
1	alerts.csv_2024-04-22 13:43:56	1499075798.10912	192.168.10.3	SURICATA STREAM CLOSEWAIT FIN out of window	3
2	alerts.csv_2024-04-22 13:43:56	1499075818.362352	192.168.10.9	SURICATA STREAM CLOSEWAIT FIN out of window	3
3	alerts.csv_2024-04-22 13:43:56	1499075818.362352	192.168.10.3	SURICATA STREAM CLOSEWAIT FIN out of window	3
4	alerts.csv_2024-04-22 13:43:56	1499075818.381786	192.168.10.9	SURICATA Applayer Wrong direction first Data	3
5	alerts.csv_2024-04-22 13:43:56	1499075818.381786	192.168.10.3	SURICATA Applayer Wrong direction first Data	3
6	alerts.csv_2024-04-22 13:43:56	1499075825.702242	192.168.10.9	SURICATA Applayer Detect protocol only one direction	3
7	alerts.csv_2024-04-22 13:43:56	1499075825.702242	192.168.10.3	SURICATA Applayer Detect protocol only one direction	3
8	alerts.csv_2024-04-22 13:43:56	1499075846.932673	192.168.10.3	SURICATA TLS invalid record type	3
9	alerts.csv_2024-04-22 13:43:56	1499075846.932673	192.168.10.12	SURICATA TLS invalid record type	3
10	alerts.csv_2024-04-22 13:43:56	1499075846.932673	192.168.10.3	SURICATA TLS invalid record/traffic	3
11	alerts.csv_2024-04-22 13:43:56	1499075846.932673	192.168.10.12	SURICATA TLS invalid record/traffic	3
12	alerts.csv_2024-04-22 13:43:56	1499075846.932771	192.168.10.12	SURICATA TLS invalid record type	3
13	alerts.csv_2024-04-22 13:43:56	1499075846.932771	192.168.10.3	SURICATA TLS invalid record type	3
14	alerts.csv_2024-04-22 13:43:56	1499075846.932771	192.168.10.12	SURICATA TLS invalid record/traffic	3
15	alerts.csv_2024-04-22 13:43:56	1499075846.932771	192.168.10.3	SURICATA TLS invalid record/traffic	3
16	alerts.csv_2024-04-22 13:43:56	1499075847.001487	192.168.10.3	SURICATA Kerberos 5 weak encryption parameters	3
17	alerts.csv_2024-04-22 13:43:56	1499075847.001487	192.168.10.12	SURICATA Kerberos 5 weak encryption parameters	3
18	alerts.csv_2024-04-22 13:43:56	1499075847.001491	192.168.10.3	SURICATA Kerberos 5 weak encryption parameters	3
19	alerts.csv_2024-04-22 13:43:56	1499075847.001491	192.168.10.12	SURICATA Kerberos 5 weak encryption parameters	3
20	alerts.csv_2024-04-22 13:43:56	1499075847.004205	192.168.10.12	SURICATA FRAG IPv4 Fragmentation overlap	3
21	alerts.csv_2024-04-22 13:43:56	1499075847.004205	192.168.10.3	SURICATA FRAG IPv4 Fragmentation overlap	3
22	alerts.csv_2024-04-22 13:43:56	1499075847.03999	192.168.10.3	SURICATA TLS invalid record type	3
23	alerts.csv_2024-04-22 13:43:56	1499075847.03999	192.168.10.12	SURICATA TLS invalid record type	3
24	alerts.csv_2024-04-22 13:43:56	1499075847.03999	192.168.10.3	SURICATA TLS invalid record/traffic	3
25	alerts.csv_2024-04-22 13:43:56	1499075847.03999	192.168.10.12	SURICATA TLS invalid record/traffic	3
26	alerts.csv_2024-04-22 13:43:56	1499075847.040029	192.168.10.12	SURICATA TLS invalid record type	3
27	alerts.csv_2024-04-22 13:43:56	1499075847.040029	192.168.10.3	SURICATA TLS invalid record type	3

FIGURE 3: Database page

5.2 Database

The database page is where the user can upload new files and manage the already uploaded files. When someone wants to analyse a csv file containing security events this is the page they will start on. To upload the file they can simply press upload file (**element 1**) and once it is uploaded a pop-up will appear as shown in figure 4.

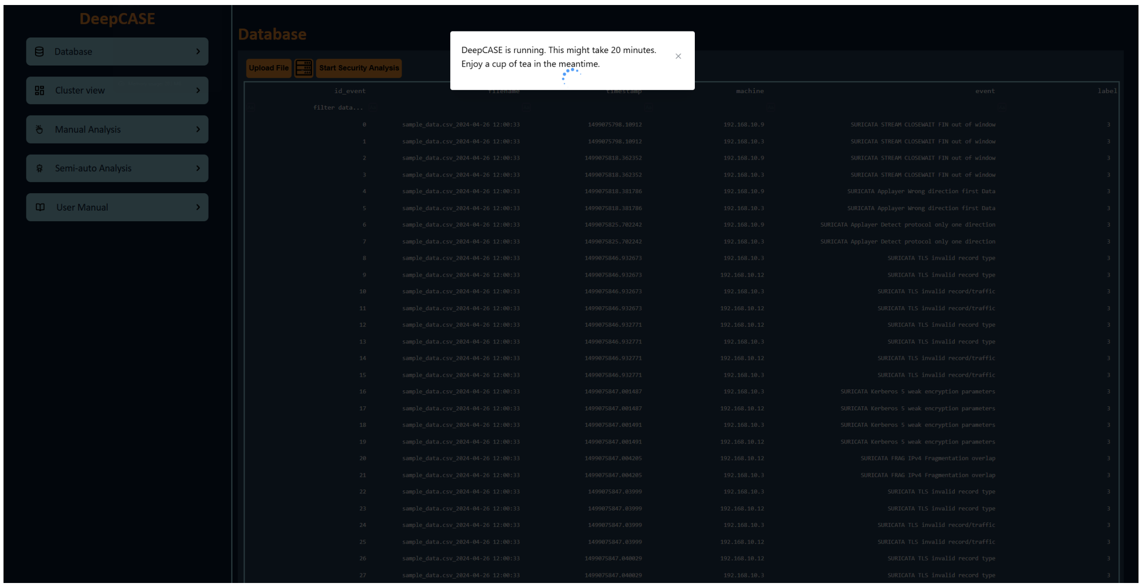


File Uploaded Successfully: sample_data.csv Press "Start Security Analysis" button to run DeepCASE.

FIGURE 4: Upload successful pop-up

After the file has been uploaded the user can let the uploaded data be analysed by DeepCASE by pressing the start security analysis button (**element 3**). Pressing the button will make the web server start the DeepCASE module and after it is done it will have clustered the security events.
The security analysis does take awhile. To alert the user of this fact the pop-up shown in

figure 5 will appear.



DeepCASE is running. This might take 20 minutes.
Enjoy a cup of tea in the meantime.

id_event	filename	timestamp	machine	event	label
filter data...					
0	sample_data.csv_2024-04-26 12:00:33	1499075798.10912	192.168.10.9	SURICATA STREAM CLOSEWAIT FIN out of window	3
1	sample_data.csv_2024-04-26 12:00:33	1499075798.10912	192.168.10.3	SURICATA STREAM CLOSEWAIT FIN out of window	3
2	sample_data.csv_2024-04-26 12:00:33	1499075818.362352	192.168.10.9	SURICATA STREAM CLOSEWAIT FIN out of window	3
3	sample_data.csv_2024-04-26 12:00:33	1499075818.362352	192.168.10.3	SURICATA STREAM CLOSEWAIT FIN out of window	3
4	sample_data.csv_2024-04-26 12:00:33	1499075818.381786	192.168.10.9	SURICATA Applayer Wrong direction first Data	3
5	sample_data.csv_2024-04-26 12:00:33	1499075818.381786	192.168.10.3	SURICATA Applayer Wrong direction first Data	3
6	sample_data.csv_2024-04-26 12:00:33	1499075825.702242	192.168.10.9	SURICATA Applayer Detect protocol only one direction	3
7	sample_data.csv_2024-04-26 12:00:33	1499075825.702242	192.168.10.3	SURICATA Applayer Detect protocol only one direction	3
8	sample_data.csv_2024-04-26 12:00:33	1499075846.932673	192.168.10.3	SURICATA TLS invalid record type	3
9	sample_data.csv_2024-04-26 12:00:33	1499075846.932673	192.168.10.12	SURICATA TLS invalid record type	3
10	sample_data.csv_2024-04-26 12:00:33	1499075846.932673	192.168.10.3	SURICATA TLS invalid record/traffic	3
11	sample_data.csv_2024-04-26 12:00:33	1499075846.932673	192.168.10.12	SURICATA TLS invalid record/traffic	3
12	sample_data.csv_2024-04-26 12:00:33	1499075846.932771	192.168.10.12	SURICATA TLS invalid record type	3
13	sample_data.csv_2024-04-26 12:00:33	1499075846.932771	192.168.10.3	SURICATA TLS invalid record type	3
14	sample_data.csv_2024-04-26 12:00:33	1499075846.932771	192.168.10.12	SURICATA TLS invalid record/traffic	3
15	sample_data.csv_2024-04-26 12:00:33	1499075846.932771	192.168.10.3	SURICATA TLS invalid record/traffic	3
16	sample_data.csv_2024-04-26 12:00:33	1499075847.001487	192.168.10.3	SURICATA Kerberos 5 weak encryption parameters	3
17	sample_data.csv_2024-04-26 12:00:33	1499075847.001487	192.168.10.12	SURICATA Kerberos 5 weak encryption parameters	3
18	sample_data.csv_2024-04-26 12:00:33	1499075847.001491	192.168.10.3	SURICATA Kerberos 5 weak encryption parameters	3
19	sample_data.csv_2024-04-26 12:00:33	1499075847.001491	192.168.10.12	SURICATA Kerberos 5 weak encryption parameters	3
20	sample_data.csv_2024-04-26 12:00:33	1499075847.004205	192.168.10.12	SURICATA FRAG IPv4 Fragmentation overlap	3
21	sample_data.csv_2024-04-26 12:00:33	1499075847.004205	192.168.10.3	SURICATA FRAG IPv4 Fragmentation overlap	3
22	sample_data.csv_2024-04-26 12:00:33	1499075847.03999	192.168.10.3	SURICATA TLS invalid record type	3
23	sample_data.csv_2024-04-26 12:00:33	1499075847.03999	192.168.10.12	SURICATA TLS invalid record type	3
24	sample_data.csv_2024-04-26 12:00:33	1499075847.03999	192.168.10.3	SURICATA TLS invalid record/traffic	3
25	sample_data.csv_2024-04-26 12:00:33	1499075847.03999	192.168.10.12	SURICATA TLS invalid record/traffic	3
26	sample_data.csv_2024-04-26 12:00:33	1499075847.040029	192.168.10.12	SURICATA TLS invalid record type	3
27	sample_data.csv_2024-04-26 12:00:33	1499075847.040029	192.168.10.3	SURICATA TLS invalid record type	3

FIGURE 5: Running DeepCASE pop-up

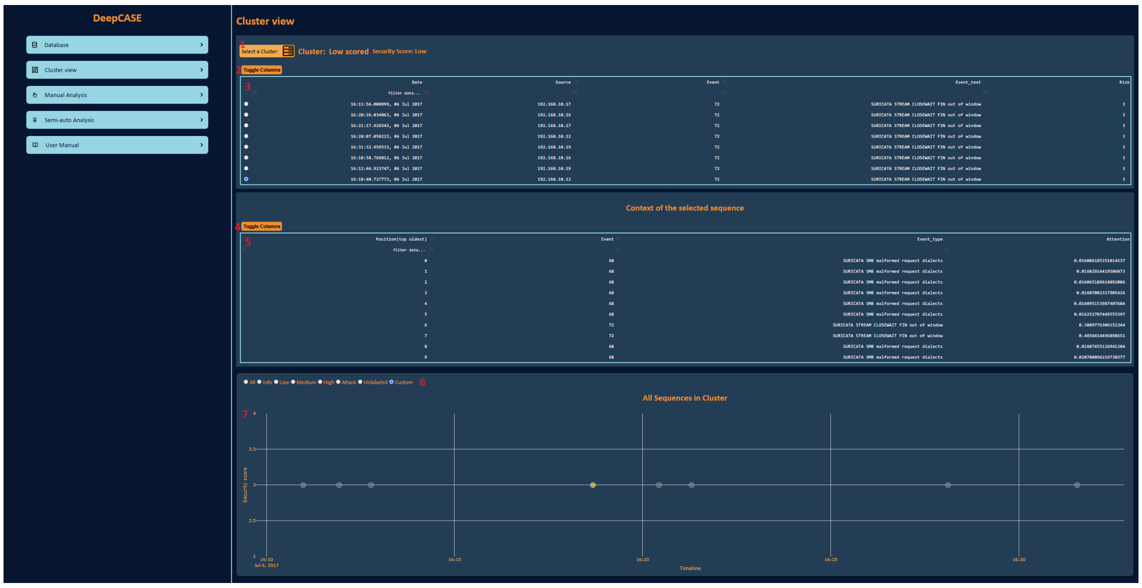
After the security analysis is done different sequences will be clustered and the user can start analyzing the data using the dashboard.
The user can also easily switch between files they have already uploaded by clicking the button with the server icon (**element 2**). This will reveal a list of uploaded files. The user can select a file and the events contained in that file will than be shown on the different pages to be analysed by the user.
The currently loaded file can also be inspected in table form in the table on the screen (**element 4**). Here the user can look through the unordered data and use the second row to filter the data. This can be used for example to check whether certain security events are present in the file or to check whether the correct file has been upload.

Elements:

1. Button to upload a file from the user's computer
2. Button to select one of the uploaded files
3. Button to start the security analysis of the previously uploaded file
4. Table that shows the currently selected file

5.3 Cluster View

This is the main page for analyzing security events, sequences and clusters. After a csv file has been uploaded and processed by the DeepCASE module, a security operator can select a cluster they want to analyze by pressing the select a cluster button (**element 1**). After a cluster has been selected the different sequences belonging to that cluster will be displayed in the table below (**element 3**). This table displays the time the sequence occurred, the source of the attack/request, a numerical representation of the event type, a description of the event and a risk label.



	Date	Source	Event	Event_text	Risk
	filter data...				
○	16:11:56.008999, 06 Jul 2017	192.168.10.17	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
○	16:20:26.034063, 06 Jul 2017	192.168.10.16	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
○	16:21:17.926543, 06 Jul 2017	192.168.10.17	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
○	16:28:07.058223, 06 Jul 2017	192.168.10.12	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
○	16:31:32.959533, 06 Jul 2017	192.168.10.19	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
○	16:10:58.769812, 06 Jul 2017	192.168.10.16	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
○	16:12:46.923747, 06 Jul 2017	192.168.10.19	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
●	16:18:40.727772, 06 Jul 2017	192.168.10.12	72	SURICATA STREAM CLOSEWAIT FIN out of window	3

Position(top oldest)	Event	Event_type	Attention
filter data...			
0	68	SURICATA SMB malformed request dialects	0.016086185351014137
1	68	SURICATA SMB malformed request dialects	0.01602916419506073
2	68	SURICATA SMB malformed request dialects	0.016065189614892006
3	68	SURICATA SMB malformed request dialects	0.01607002317905426
4	68	SURICATA SMB malformed request dialects	0.016095153987407684
5	68	SURICATA SMB malformed request dialects	0.016251707449555397
6	72	SURICATA STREAM CLOSEWAIT FIN out of window	0.3809776306152344
7	72	SURICATA STREAM CLOSEWAIT FIN out of window	0.4856414496898651
8	68	SURICATA SMB malformed request dialects	0.01607455126941204
9	68	SURICATA SMB malformed request dialects	0.020708896219730377

FIGURE 6: Cluster view page

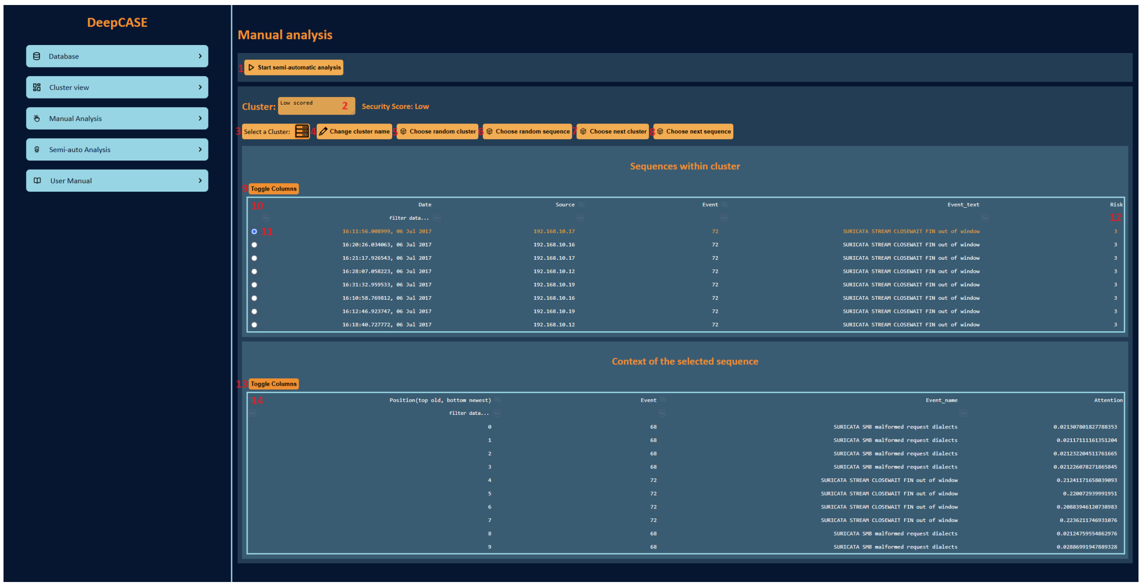
The user can adjust the table by toggling certain columns on or off using the the toggle columns button (**element 2**) to tailor the view to their desires and make the view more suitable for their needs.
The user can inspect a sequence by selecting a sequences using the radio buttons on the leftmost column of the top table (**element 3**). The selected sequence will be displayed in the second table (**element 5**). Here the user can see the 9 preceding events to get a better understanding of the sequence. This table displays the position of every event in the sequence (the order in which they occurred), again a numerical and text based description of the event and a attention score. The attention score shows which events are most relevant to the sequence. A higher attention score means more relevance and the scores of all events in a sequence should add up to 1.
Again the columns of this table can be toggled on or off just like the top table using the toggle columns button (**element 4**).
The graph on the bottom of the page (**element 7**) shows the different sequences in the cluster in visual form. Every sequence is displayed using a colored dot. The risk labels are plotted on the vertical axis and the timestamp on the horizontal axis. In this view the user can a get better insight in the timeline of the events. The user can also use the buttons on the top of the graph (**element 6**) to highlight sequences of a certain risk level. Sequences of that risk level will appear colored in the graph while the other dots will become gray.

Elements:

1. Button to select a cluster
2. Button to toggle columns in element 3 on or off
3. Table showing the sequences contained in the selected cluster
4. Button to toggle columns in element 5 on or off
5. Table that shows information about the selected sequence
6. Buttons to highlight a class of sequences in element 7

7. Graph that plots the sequences' risk level against time of occurrence

5.4 Manual Analysis



Date	Source	Event	Event_text	Risk
filter data...				
16:11:56.008999, 06 Jul 2017	192.168.10.17	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
16:20:26.034063, 06 Jul 2017	192.168.10.16	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
16:21:17.926543, 06 Jul 2017	192.168.10.17	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
16:28:07.058223, 06 Jul 2017	192.168.10.12	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
16:31:32.959533, 06 Jul 2017	192.168.10.19	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
16:10:58.769812, 06 Jul 2017	192.168.10.16	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
16:12:46.923747, 06 Jul 2017	192.168.10.19	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
16:18:40.727772, 06 Jul 2017	192.168.10.12	72	SURICATA STREAM CLOSEWAIT FIN out of window	3

Position(top old, bottom newest)	Event	Event_name	Attention
filter data...			
0	68	SURICATA SMB malformed request dialects	0.021307801827788353
1	68	SURICATA SMB malformed request dialects	0.02117111161351204
2	68	SURICATA SMB malformed request dialects	0.021232204511761665
3	68	SURICATA SMB malformed request dialects	0.021226078271865845
4	72	SURICATA STREAM CLOSEWAIT FIN out of window	0.21241171658039093
5	72	SURICATA STREAM CLOSEWAIT FIN out of window	0.220072939991951
6	72	SURICATA STREAM CLOSEWAIT FIN out of window	0.20883946120738983
7	72	SURICATA STREAM CLOSEWAIT FIN out of window	0.2236211746931076
8	68	SURICATA SMB malformed request dialects	0.02124759554862976
9	68	SURICATA SMB malformed request dialects	0.02886991947889328

FIGURE 7: Manual analysis page

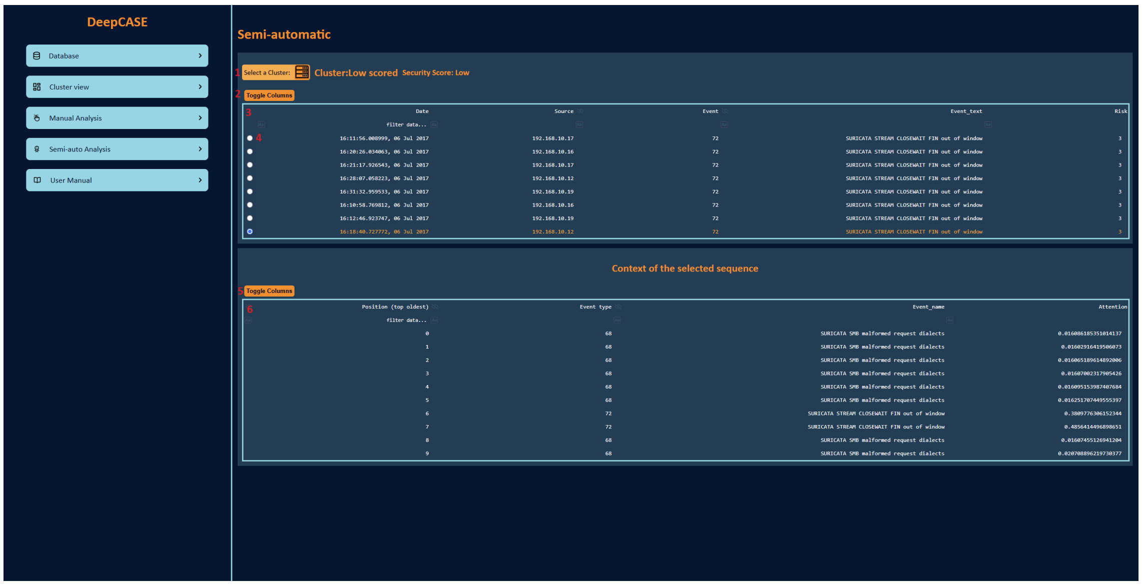
The manual analysis page allows the user to alter certain attributes of the clusters and events and help train the AI system. This is why this page is the most interactive one on the dashboard.
A user can select a particular cluster with the select a cluster button (**element 3**) Which will then appear in the table below (**element 10**). The user can also choose to let the dashboard pick a cluster for them by either letting the dashboard select a random cluster (**element 5**) or pick the cluster the dashboard feels is most relevant (**element 7**). The same option are available for letting the dashboard pick a sequence with respectively **element 6** and **element 8**.
When a cluster is selected the dashboard shows the name of the cluster in a text box (**element 2**). The user can decide to change the name of a cluster to a name they may find more fitting. To do this the user has to write the new name in the text box (**element 2**) and press the change cluster name button (**element 4**).
The sequences contained in the selected clusters will appear in the top table (**element 10**). Just like the other tables, the columns in this table can be modified by using the toggle columns button (**element 9**).
The user can select a sequence in the table using the radio buttons in the left most column (**element 11**). Once selected the text in the row will turn orange to show the user it is selected. Then the context of the sequence will be displayed in the bottom table (**element 14**) which works the same as on the cluster view page and again the columns are toggleable with the toggle columns button (**element 13**).
In the top table (**element 10**) the user can change the values of the risk label of the selected sequence by clicking on the current risk label (**element 12**), typing in an integer value and pressing enter. This new risk label will then be saved to the database.
Once the user has changed all the risk labels they feel needed to be changed, they can start the semi-automatic analysis. By pressing the start semi-automatic analysis button

(**element 1**) they will startup DeepCASE again which will then take the newly inputted values of the risk labels into account.

Elements:

1. Button to run the semi-automatic analysis
2. Text box to display and change a cluster name
3. Button to select a cluster
4. Button to confirm the change of a cluster name
5. Button that selects a random cluster
6. Button that selects a random sequence
7. Button that select a relevant cluster
8. Button that select a relevant sequence
9. Button to toggle columns in element 10
10. Table that displays sequences in the selected clusters
11. Radio buttons to select a sequence
12. Adjustable risk label of the sequences
13. Button to toggle columns in element 13
14. Table that shows information about the selected cluster

5.5 Semi-auto Analysis



	Date	Source	Event	Event_text	Risk
	filter data...				
○	16:11:56.008999, 06 Jul 2017	192.168.10.17	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
○	16:20:26.034063, 06 Jul 2017	192.168.10.16	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
○	16:21:17.926543, 06 Jul 2017	192.168.10.17	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
○	16:28:07.058223, 06 Jul 2017	192.168.10.12	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
○	16:31:32.959533, 06 Jul 2017	192.168.10.19	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
○	16:10:58.769812, 06 Jul 2017	192.168.10.16	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
○	16:12:46.923747, 06 Jul 2017	192.168.10.19	72	SURICATA STREAM CLOSEWAIT FIN out of window	3
◉	16:18:40.727772, 06 Jul 2017	192.168.10.12	72	SURICATA STREAM CLOSEWAIT FIN out of window	3

Position (top oldest)	Event type	Event_name	Attention
filter data...			
0	68	SURICATA SMB malformed request dialects	0.016086185351014137
1	68	SURICATA SMB malformed request dialects	0.01602916419506073
2	68	SURICATA SMB malformed request dialects	0.016065189614892006
3	68	SURICATA SMB malformed request dialects	0.01607002317905426
4	68	SURICATA SMB malformed request dialects	0.016095153987407684
5	68	SURICATA SMB malformed request dialects	0.016251707449555397
6	72	SURICATA STREAM CLOSEWAIT FIN out of window	0.3809776306152344
7	72	SURICATA STREAM CLOSEWAIT FIN out of window	0.4856414496898651
8	68	SURICATA SMB malformed request dialects	0.01607455126941204
9	68	SURICATA SMB malformed request dialects	0.020708896219730377

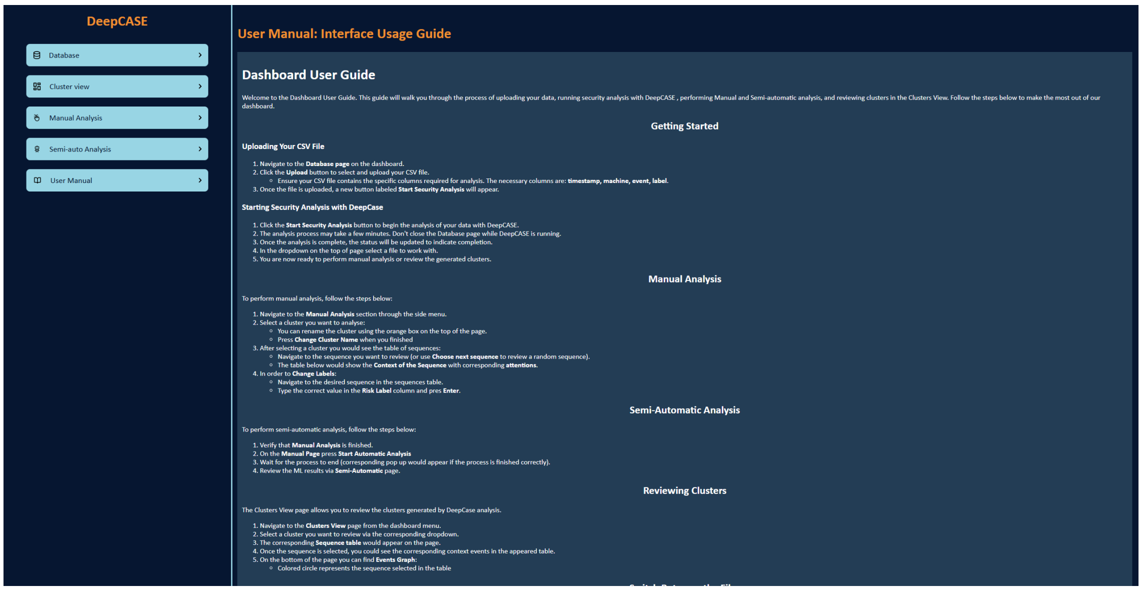
FIGURE 8: Semi-automatic analysis page

On this page the user can view the result of the semi-automatic analysis. It is very similar to the cluster view and manual analysis page but it shows less information and is less interactive making the page more clear and less cluttered. This might be useful to some users.
A user can select any cluster using the select a cluster button (**element 1**) which sequences will then be displayed in the top table (**element 3**). Again, the columns in this table are toggleable using the toggle columns button (**element 2**). A user can select a sequence using the radio buttons in the left most column of the table (**element(4)**. The context of that sequence will then be displayed in the bottom table (**element 6**) which columns are also toggleable (**element 5**).
For a more in depth explanation about the elements contained on this page please look at either the description of the cluster view page or the manual analysis page as all the elements contained here are also present there.

Elements:

1. Button to select a cluster
2. Button to toggle columns in element 3 on or off
3. Table showing the sequences contained in the selected cluster
4. Radio buttons to select a sequence
5. Button to toggle columns in element 6 on or off
6. Table that shows information about the selected cluster

5.6 User Manual



User Manual: Interface Usage Guide

Dashboard User Guide

Welcome to the Dashboard User Guide. This guide will walk you through the process of uploading your data, running security analysis with DeepCASE , performing Manual and Semi-automatic analysis, and reviewing clusters in the Clusters View. Follow the steps below to make the most out of our dashboard.

Getting Started

Uploading Your CSV File

1. Navigate to the **Database page** on the dashboard.
2. Click the **Upload** button to select and upload your CSV file.
    - Ensure your CSV file contains the specific columns required for analysis. The necessary columns are: **timestamp, machine, event, label**.
3. Once the file is uploaded, a new button labeled **Start Security Analysis** will appear.

Starting Security Analysis with DeepCase

1. Click the **Start Security Analysis** button to begin the analysis of your data with DeepCASE.
2. The analysis process may take a few minutes. Don't close the Database page while DeepCASE is running.
3. Once the analysis is complete, the status will be updated to indicate completion.
4. In the dropdown on the top of page select a file to work with.
5. You are now ready to perform manual analysis or review the generated clusters.

Manual Analysis

To perform manual analysis, follow the steps below:

1. Navigate to the **Manual Analysis** section through the side menu.
2. Select a cluster you want to analyse:
    - You can rename the cluster using the orange box on the top of the page.
    - Press **Change Cluster Name** when you finished
3. After selecting a cluster you would see the table of sequences:
    - Navigate to the sequence you want to review (or use **Choose next sequence** to review a random sequence).
    - The table below would show the **Context of the Sequence** with corresponding **attentions.**
4. In order to **Change Labels:**
    - Navigate to the desired sequence in the sequences table.
    - Type the correct value in the **Risk Label** column and pres **Enter**.

Semi-Automatic Analysis

To perform semi-automatic analysis, follow the steps below:

1. Verify that **Manual Analysis** is finished.
2. On the **Manual Page** press **Start Automatic Analysis**
3. Wait for the process to end (corresponding pop up would appear if the process is finished correctly).
4. Review the ML results via **Semi-Automatic** page.

Reviewing Clusters

The Clusters View page allows you to review the clusters generated by DeepCase analysis.

1. Navigate to the **Clusters View** page from the dashboard menu.
2. Select a cluster you want to review via the corresponding dropdown.
3. The corresponding **Sequence table** would appear on the page.
4. Once the sequence is selected, you could see the corresponding context events in the appeared table.
5. On the bottom of the page you can find **Events Graph**:
    - Colored circle represents the sequence selected in the table

FIGURE 9: User manual page

This page shows a quick guide to the dashboard guiding the user through the steps of using it. This can be used to quickly teach new users how to interact with the dashboard or as reference for more experienced users. Including this page will raise the accessibility of the dashboard and we therefore choose to include it as that was one of our quality requirements.

6 Global design

6.1 Database Schema

In this part, we will go through the decision process of how the database in our project came to what it is. For the DeepCASE module, it is not necessary to use a database. However, we have chosen to save it in a database, because this allows for better modifyability and a better overview of the system – especially when trying to visualize it in a dashboard.

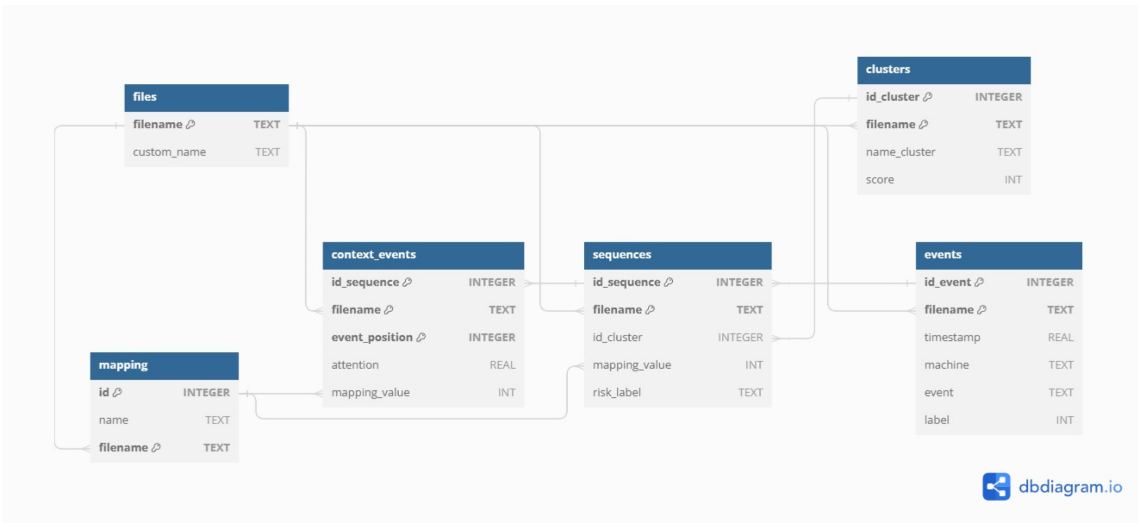


FIGURE 10: Database Schema

6.1.1 Our Approach

For the development of the database, we have used SQLite [6], which is a simple low-configuration database setup. This allows us to shift our focus to development, instead of configuration. SQLite has been experienced as efficient (enough) and suitable - i.e. reasonable speed - for our purposes. For production use, it is good to consider other database types, like PostgreSQL. PostgreSQL is more robust at handling greater volumes of data. However, SQLite is suitable for local use, which is the scope of the project right now.

6.1.2 Graphical Overview & Structure

The database schema, as shown in Figure 10, serves as a blueprint for the structured storage and retrieval of security events, which are needed for DeepCASE. Each table within the schema is designed with specific constraints to ensure data integrity and consistency. The primary and foreign key constraints enforce a strong relational structure.

6.1.3 Conclusion

The implementation of our database schema is crucial for the creating of our dashboard. Opting for a database, despite DeepCASE's standalone functionality, affords us modularity and clarity, particularly with dashboard visualisations.
While SQLite's simplicity has facilitated our development with sufficient efficiency for current demands, we acknowledge PostgreSQL's superiority for handling larger datasets, which we may consider for future scalability.

Overall, the integration of our database with DeepCASE creates a robust framework for security event analysis. It ensures that our system remains agile and prepared for the evolving landscape of data and security management. The schema depicted in Figure 10 is more than a static design; it's the baseline for an even better dashboard.

6.2 Design Patterns & File Structure

Within the source code, standard patterns occur to enforce modular behavior. This design has been particularly useful when implementing file structures between different imports. Underneath, we will explain further design patterns and demonstrate specific global design choices.

6.2.1 Design Patterns

Let us start with one of the most important patterns, which is the MVC-Pattern (Model-View-Controller) pattern.

MVC-Pattern If we go through each different part of the pattern, then we start with the M from Model. The model handles all the data interaction, which has been realized using the DAO (Data Access Object) Pattern. This module handles all the queries, where it works together with the database.py to actually return results. Then we continue to the view, which is responsible for representing data interaction to the user. The templates and static encapsulate core design components. However when it comes to dynamic pages, Dash allows for an intuitive way for the user to interact with the data. Lastly, we have the controllers. The controller is the intermediary between the view and the model. The 'pages' directory is responsible for the interaction of all the pages. These will handles the requests and perhaps callbacks on requests.

DAO-Pattern The DAO pattern has been used in our application to encapsulate database logic. It provides an abstraction layer between the application logic and the database, making data communication and manipulation easier. This has been used in our DAO class.

- **Data Persistence:** Methods like `save_input` and `save_sequencing_results` allow for the processing of input files and sequencing data, storing them in the database.
- **Data Retrieval:** Functions `get_initial_table` and `get_context_per_sequence` allow data fetching, offering a simplified interface to retrieve data without revealing the complexity of the underlying database queries.

Observer Pattern The DeepCASE dashboard uses the Observer pattern through Dash's callback system, which allows UI components to react to state changes without direct coupling. Callbacks (acting as observers) respond to user interactions by updating the UI, maintaining the interface's reactivity. This design is useful for scalability and clean separation of concerns.
There are plenty of unmentioned patterns, such as builder pattern. But many of these patterns are automatically used to create a more effective code base. This section is here to explain the most import design patterns and demonstrate that during development, design pattern have been kept in mind.

7 Detailed design

In this section we will give a detailed explanation of the workings and design chooses made in the most important parts of the dashboard. This will hopefully give a better understanding of the system and make it easier to maintain for further developers.

7.1 DeepCASE integration

In order to integrate DeepCASE with the operational framework we have designed a certain system architecture. Such integration includes design and implementation of several modules that would ensure successful cooperation between DeepCASE module and the web application and database. A detailed schema of the integration of different modules and general data flow is presented in Figure 11.

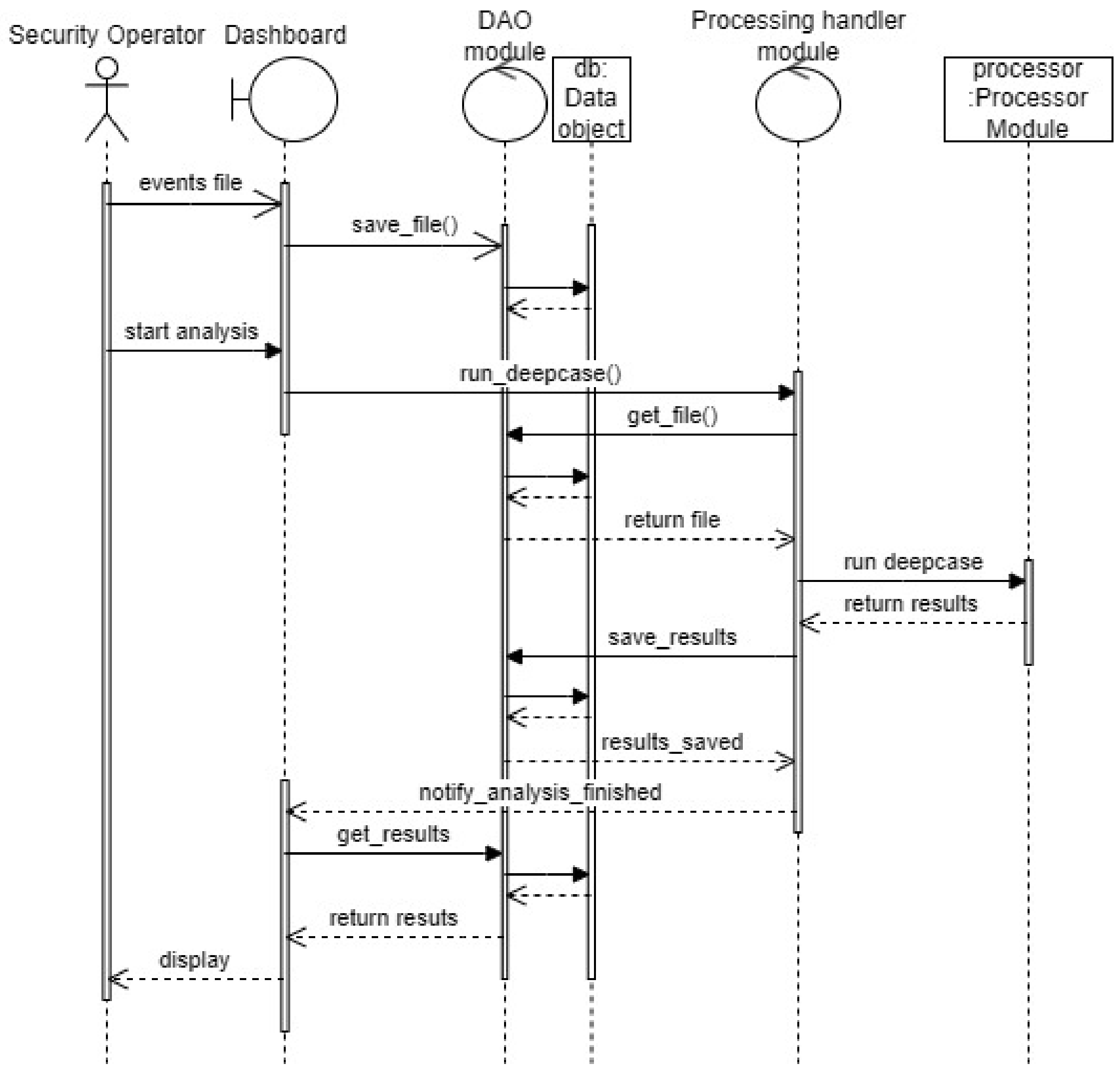


Figure 11: Sequence Diagram

Process Module Process module represents the core functionality of the DeepCASE module, as well as handles step by step analysis operations.

- **Sequencing:** Organises input events into sequences by extracting the main event and defining corresponding array of context events.
- **ContextBuilder training:** Performs model training to determine attention scores of the events.
- **Clustering:** Clusters sequences based on the events and attention scores.
- **Manual analysis:** Labels the generated clusters according to predefined algorithm, returns `tensor.Torch` objects [7].
- **Semi-Automatic Analysis:** Predicts risk labels based on the results of Manual analysis.

Processing Handler Module The Processing Handler module ensures correct execution of all the process steps. The module is also responsible for secure and consistent data management; and interaction via Data Access Object (DAO), mainly by storing intermediate and final analysis results.

Data Access Object (DAO) Module The purpose of DAO module is to manage database operations, including storing, updating and retrieving data necessary for both backend processes and front end visualizations. DAO operates using `pandas.Dataframe` objects. [4]

7.2 Front End in Dash

To create the front end we used a python library called Dash which is designed to be a quick and easy way to make dashboard web apps. This library lets you use CSS and a HTML-like language written in python. Dash provides the possibility to easily integrate graphs and table in the web app and provides objects to run a web server.

7.2.1 HTML

The page is split into two parts: the navigation bar and the content container. The way Dash works is that a certain part of the screen is static, in this case the navigation bar and the other part is dynamic, the content container. So when a link is activated to switch to another the web page only the content container will change. This is done by linking a URL with a python file using the *register_page* method and then making the script inside that file assign a new value to the *layout* variable. This value is a python array that closely resembles the way an HTML file would be written.
The navigation bar being static has the upside that it does not have to be coded into every page but it also has the downside that it is not adjustable per page. An example of this is that we wanted to highlight the link to the current page in the navigation bar. However, this was not possible due to it's static nature. A quick and dirty way to solve this would be to move the navigation bar into the dynamic section of the page and hard code it for every page to have a different highlighted link.

7.2.2 CSS

Regular CSS is used to do the styling for the front end. The stylesheet is a file called *styling.css* in the assets folder. Every node in the HTML-like python array can be assigned a *classname*-parameter. This works the same as the *class*-attribute in HTML.

7.2.3 Color scheme

A dark blue and orange color scheme is used. We went through several design iterations before we landed on these colors. These colors were chosen to give the dashboard a clear and professional look. To follow this palette strictly four variables have been assigned in the style sheet. Namely:

1. primary-color: #FF8E00 🟧
2. secondary-color: #84D5E6 🟦
3. tertiary-color: #001633 ⬛
4. quarniary-color: #FF8E00 ⬜

Of course quarniary should be called quaternary but we used the way it is spelled in the code to avoid confusion.

7.3 File-structure

In the code-base we aimed for maintainable, readable, scalable, clear and organized file hierarchy by sorting different parts of our application into separate folders and files.
For the app pages, we split the files into distinct parts based on specific design aspects, HTML elements, callbacks and resources. This allowed to prevent duplicated code.
The data folder contains of the files pertaining to the database. Except the data access object which is outside this folder as it acts more as a intermediary.
Then we have the processing folder, this is were DeepCASE run. This can be accessed as well with an object: the processor access object.
Lastly, we have a test folder with some junit test were applicable.
Further we have chosen to not heavily rely on interfaces in our code base. Instead, we import files with shared methods, which has proven effective, especially in Python where interfaces are not enforced by the language. Our step-by-step approach to adding functionality by different members has proven that there is a certain level of readability, scalability and maintainability of our code base. If one part of the code needs to be changed, a small change was sufficient.

7.4 Variables

There are a few different kinds of variables, namely, client-based global variables, server-based global variables, constants, and temporary variables. In this section we will talk about front-based usage of those variables. In other parts of our code standard practice is applied.

Constants In the front end, constants are used to prevent duplicate code. However, these have a different callback and table. This is necessary to make dash work, since every element needs a unique id. To solve this we made a suffix for them. In the code we use them as input to generate a table that will appear multiple times.

Local variables Local variable are used as expected.

Client-based global variable At this point the website is only tested using local hosts with only one device accessing said host[1]. This means that for all client-based variables we could have used global variables. However, it appears that when the app gets deployed, you get unexpected behaviour. Therefore we use a store element, this a client based variable, it remembers which cluster or row is selected, in order to know which data to adjust. So most variables where you think a global one is used, is client-based.
There is one exception, that is for the variable that determines if the deep-case is processing. We made it such that only one process is going on. For example to make sure a file cannot be manipulated while DeepCASE is running. An improvement on this would be to use the status variable in database more extensively such that only one process can happen at the same time on one file. A thing that needs to be kept in consideration while doing that is that, the DAO keeps track of the selected file, this shouldn't be change during a process. So a solution need to be found for that.

7.5 Threads

Dash has a bug for multi-page apps. This bug causes all callbacks to get interrupted when the user switches pages. For example if DeepCASE is running it will get interrupted as well. This might cause the server to crash.
Therefore we run these processes in different threads to secure these callbacks can finish. This is not a 100% fix simply because the user can, with help from a program, click faster and let the program crash. This approach makes it so that the feedback from the process not always reaches the user. However, This can be fixed by choosing a different platform then Dash to run the app on. Thread fix is in our eyes the best solution with the chosen platform.

7.6 Selection of events

The client had the request to let the dashboard give suggestions on the most relevant sequences to label. We have split this task up in two parts, namely choosing the next cluster and next sequence. This is done because the security operator can handle one cluster at a time and then go to the next one instead of jumping all over the place.
Firstly we implemented this algorithm based on picking a random sequence or cluster. This would be useful to get a taste of the data that is present. However, It would be nicer for the security operator to have sequences presented that bear some relevance. Therefore we created an algorithm.

Cluster We select the clusters where either information is missing or which are very risky, so that the security operators are aware that those exist. This algorithm is rather slow, because the system calculates the security label for each cluster. This can be resolved by having the cluster risk labels stored in the database. Another options, is to keep intermediate result in a client variable.

Sequence For the sequence we select all the unique combinations between, the machine, risk label, and kind of event. Additionally, we choose to select the security labels with a negative value. Those negative values are there because there is not enough information to give it a proper risk label and it needs to be seen by security operators. After the selection is made, only one suggestion is given, this is determined randomly. The assumption is that the label might be changed and thus not be seen anymore by the algorithm. In order to make this process more deterministic, it can be kept track of in a client-based variable. This would be a further improvement.

8 Testing the System

When creating a dashboard for the DeepCASE module, it is important to keep functionality in mind. This functionality and user flow should be formulated in a document, such that the rules and guidelines are clear for the development. This document are the requirements. In this section we will formulate tests to see whether these requirements have been fulfilled.

8.1 Testing Approach

The testing approach included the implementation of unit, integration and user testing methods. The unit and integration testing are mainly applied to process and database management operations, while front end and dashboard are mainly tested during the user testing phase. The DeepCASE module was not tested assuming that the security tool works without flaws. The reason to assume this is that it was already extensively tested during it's development.

8.1.1 Unit testing

The aim of unit testing is to ensure the correct functionality of the database, since the database is one of the most crucial parts of the system. The testing was performed with standard unit test Python library and a dummy database. All the tests accessed the database from memory in order to avoid erasing or modifying already existing data. The test covered both internal and external functions of the database, checking the correct implementation of all the queries.
Several faults were discovered in the process. One of the bugs was related to returning the sequences according to specified id: the responsible method returned the necessary entry multiple times instead of one. The reason of this bug was insufficient check of join constraints in SQL query. An extra check of equality between `sequence_id` and `event_id` should be added to avoid repetitive results.
Another issue was related to the data types of columns representing `risk_label` or `score` of sequences across the table. While all of the columns used to store the score of sequence were of type "INT", the column `risk_label` in "sequences" table had "TEXT" data type. The issue of mismatched data types could potentially lead to errors in data representation, therefore it was decided to set all columns storing the risk labels of sequences to "INT" type.

8.1.2 Integration testing

After testing the database via unit testing, integration testing is necessary. All the modules work in cooperation with each other and are dependant on the correct functionality of dependant modules. Hence, in order to address potential flaws in the system we perform Top-Down integration testing. The details of the testing are described in Figure 12.
The internals of the system: Process, Process Handler, and DAO modules, were tested using the standard unit test Python library using patches to imitate the work of integrated modules. No issues arose with the integrated workflow of the modules on both higher and lower levels of the system. The DeepCASE module itself was not tested explicitly.

8.1.3 User Testing

In order to test the usability of the dashboard, we performed user testing. Since the primary goal of the project was to create a user-friendly dashboard that would integrate all

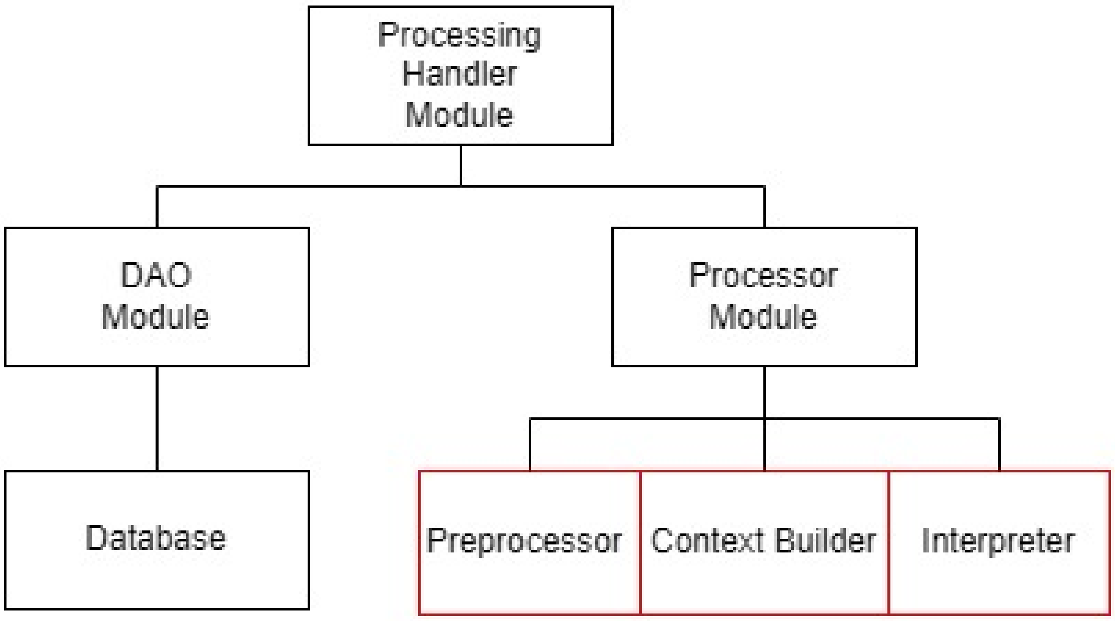


Figure 12: Integration Testing Diagram

functionality of DeepCASE, the objective of user testing was to receive feedback regarding both appearance, and accessibility of the dashboard. The users were required to perform the following tasks:

1. Upload the file using the database page.
2. Run the DeepCASE analysis.
3. Inspect the generated clusters on the manual page.
4. Inspect the sequences in a single cluster on the manual page.
5. Inspect context events of sequences on the manual page.
6. Change risk values of the sequences on the manual page.
7. Rename the cluster on the manual page.
8. Run the Semi-automatic analysis.
9. Inspect the sequences and its context events on a the semi-auto analysis page.
10. Inspect the clusters on the cluster view page.
11. Upload a new file.
12. Switch between the uploaded files.

Testing on usability of the dashboard provided a lot of valuable feedback.
As follows from Table 1, both system operation and design choices were sufficient, except some minor issues that were addressed and fixed further in the development.

Feedback	**Solution**
Analyze button disappears from Database page when switching between pages.	The button "Start Security Analysis" is now always present on the page but is disabled on corresponding DeepCASE run status.
"Enter" command should change the cluster name.	Added option to change cluster name: now both "Change cluster name" and "Enter" command work.
"Start Automatic Analysis" button is not robust against multiple clicks.	Added popups indicating that server is busy running Semi-Automatic analysis.
Automatic analysis button does not disappear when the analysis is finished.	Added functionality where button gets disabled once analysis is finished.
It is unclear when Database page is loading	The browser tab enters "Uploading.." mode when the page is loading the table.

Table 1: Feedback Received from User Testing and Presented Solutions

8.2 Requirements Testing

The following tables describes all functional and non-functional requirements specified and approved by supervisor at the proposal stage. The requirements are divided into the categories according to MoSCoW model [3]. The status of each requirement is indicated in corresponding column with 0-1 scale, where 0 represents "Not Done", 0.5 - "Still in Progress", and 1 - "Completed".

Functional Requirements

Requirement	Progress	Remarks
Must Have		
As a user, I want to upload data in a format accepted by the DeepCASE module.	1	
As a user, I want to be able to train the DeepCASE module on the uploaded data.	1	
As a user, I want to be able to score individual events. (Manual analysis)	1	
As a user, I want the DeepCASE module to give a score to uploaded data (Semi-automatic analysis)	1	
As a user, I want the system to store the scores of the clusters.	1	
As a user I want to be able to inspect security event sequences. (Security event sequences)	1	
As a user I want to be able to see which events in the sequence are correlated. (Context builder)	1	
As a user, I want to be able to inspect individual events. (Interpreter)	1	
As a user, I want to be able to approve or disapprove the score given to a sequence. (Semi-automatic analysis)	X	Dropped Requirement
Should Have		
As a user, I want the results of the uploaded security files to be stored on the server and available for viewing by any user.	1	
As a user, I want the system to randomly suggest which sequences to rate during the manual phase.	1	
As a user, I want to have a basic manual integrated into the website.	1	
As a user, I want the system to visualize the clusters in a graphical way.	1	
Could Have		
As a user, I want to be able to filter sequences with specific parameters. The parameters could be time, IP address, type of events & risk values	1	
As a user, I want to have a basic guide on the website, so that working with the new system is easy.	1	
As a user, I want the system to have regular backups.	0	
As a user, I want the system to suggest the most relevant sequences by an algorithm. The algorithm should be based on specific metrics, (e.g., greatest difference in vectors)	1	
Will Not Have		
As a user, I want to connect it to real-time security loggers, so that the data can be analysed in real-time.	0	
As a user, I want to receive notifications about newly detected malicious events.	0	

Non-Functional Requirements

Requirement	Progress	Remarks
As a user, I want the dashboard and analysis to show results within a reasonable time frame (1 or 2 seconds).	1	
As a user, I want the dashboard to be understandable with minimal training.	1	
As a user, I want the dashboard to be intuitive.	1	
As a user, I want the dashboard to give me accurate and timely feedback when an error occurs.	1	
As a user, I want the system to not have any substantial freezes.	1	
As a user, I want the system to be robust when against unexpected inputs.	0.5	
As a user, I want the data shown in the dashboard to be correct.	1	
As a user, I want the dashboard's load time to remain static with the addition of data to the database.	X	Keep in mind
As a user, I want the system to be compatible with all major browsers.	0.5	
As a user, I want the dashboard to be available 99% of the time.	1	
As a developer, I want the code base to be maintainable.	1	
As a developer, I want the code to be commented on and documented.	1	

9 Limitations And Further Improvements

The DeepCASE dashboard we have developed is a major step towards creating a user-friendly and interactive UI for the DeepCASE module. While it demonstrates great potential, further development is required to enhance its functionality. This section details the current features and identifies areas for improvement, facilitating future enhancements by other developers.

9.1 Database Page

Current Functionality: The system currently supports file uploads and subsequent execution of the DeepCASE analysis, which meets the basic operational expectations.

Issues and Proposed Solutions:

- **Upload Speed:** Uploads are slow on some devices, likely due to the handling of large files and server-side rendering limitations.
  Proposed Solution: Upgrade to a more powerful server to manage heavy computations more efficiently. Another solution is to rewrite the code to take a lazy approach to uploading/displaying data, so that it only uploads/displays as much as is needed.

- **Feedback Mechanism:** A feedback box appears after the interaction that might take a long time, intended to inform users that the process may take a while. However, this popup can be easily discarded and there is no way to know when the process finishes if the users switches to another page.
  Proposed Solution: Redesign the popup to be a status bar that is always present in the sidebar and let the user know when the operation finishes. Instead of using a popup overlay, a status bar would allow you to perform other actions in the meantime.

9.2 Cluster view

- **Graph Visualization:** The graph shows the sequences in the selected cluster, but it lacks more polish from the design and functionality perspective.
  Proposed Solution: Although it gets the job done, it could be further improved by adding the ability to hover over the point to see more information and by making the graph look more pleasing by updating the design and styling the filter buttons.

9.3 Manual analysis

- **Interface Clarity:** The interface shows a name change input box alongside a button labeled "Change cluster name." During tests, users mistook this for a direct action button rather than a submit button, leading to confusion.
  Proposed Solution: Redesign the dashboard to clearly differentiate between input fields and action buttons, perhaps by updating the button text to "Submit name change."

- **Risk Level Change Feedback:** It is possible to change the risk level of a given sequence, which meets the basic requirements. However, this function is not implemented intuitively.
  Proposed Solution: The system should be more responsive to user actions and clearly show when a change is made or when the user forgot to save the changes. When

changing the risk level of a sequence, a popup which does not obstruct the view/actions of the user should be shown instead.

- **Choosing Next Cluster:** The button that lets the system choose a next cluster based on an algorithm is not optimized and is deemed to be too slow for general usage. Also, it is not clear by which criteria the next cluster is chosen.
  Proposed Solution: Improve the performance of the algorithm, so that it takes a shorter time to find a cluster to score. Additionally, add an info box describing the manner in which the next sequence is chosen.

- **Choosing Next Sequence:** This feature was initially implemented with two "Choose random ..." buttons. Two additional buttons "Choose next ..." have been added to offer more choice for the user, but it introduces some design issues.
  Proposed Solution: It is possible to combine this into one button or have the ability to toggle between them to simplify the interface.

- **Button Functionality:** The button for initiating semi-automatic analysis is not completely fool-proof and it can be broken by specific actions made by the user while the analysis is still not complete, which could result in the need to restart the whole dashboard.
  Proposed Solution: Implement state management to prevent re-initialization of the process and to provide clear instructions to users about maintaining their session until the analysis is complete.

9.4 Semi-automatic Analysis

- **Similarity to Cluster View:** The current page of Semi-Automatic Analysis is very similar to the functionality of the Cluster View page.
  Proposed Solution: This could be solved by merging the two pages together, or thinking of a different unique way to show the semi-automatic step.

9.5 General dashboard

The system as a whole has some limitations and ways it can be further improved upon:

- **Loading Mechanism:** Whenever the system is executing something, it will change the page title to "Updating...". This is not that visible however, which can make it appear as not providing feedback to the user.
  Proposed Solution: Include a spinner whenever the system is executing something client-side or server-side.

- **Table row selection & column names:** Selection of a row is done through a radio button on the left side of each row, which can be unintuitive and clunky. Column names are hidden/toggled through a "Toggle columns" button, included automatically by Dash, the location of which cannot be changed.
  Proposed Solution: Each row could be made to be selectable, at any point along the row. As for the "Toggle columns" button, an icon could be given to the button to reduce clutter on the table. Additionally, there may be a way to change the position of this button using JavaScript, or by creating tables in a different way, besides Dash.

9.6 Conclusion

The aforementioned sections highlight several critical areas where the DeepCASE dashboard could benefit from enhanced functionality and improved user interaction. While the dashboard shows promise, it struggles with certain limitations, especially when managing large data sets [5]. We hope that the insights provided will guide effective improvements to the system's architecture and user interface.

10 Conclusions

As we wrap up this project, we want to thank Dr. Thijs van Ede for his invaluable help. Thijs' enthusiasm and helpful feedback during our weekly meetings greatly improved our work and understanding of the DeepCASE, which was necessary to create the dashboard. This project has been a great learning experience for all of us. We explored the complex world of security analysis, reaching new grounds of AI-assisted monitoring systems. This not only improved our technical skills but also helped us learn to cooperate, which was necessary to keep improving what we have.
We are proud of the dashboard we created. We went through all phases of development, working with a real stakeholder and were amazed by the self-management.
Lastly, we thank everyone who supported us, from the project group itself to our supervisor who pushed us to think deeply and creatively. The skills and knowledge we've gained from this project will surely help us in our future technical computer science life.
Group 1

References

[1] Store| dash for python| documentation plotly.

[2] Thijs van Ede, Hojjat Aghakhani, Noah Spahn, Riccardo Bortolameotti, Marco Cova, Andrea Continella, Maarten van Steen, Andreas Peter, Christopher Kruegel, and Giovanni Vigna. Deepcase: Semi-supervised contextual analysis of security events. *2022 IEEE Symposium on Security and Privacy (SP)*, May 2022.

[3] A. Hudaib, R. Masadeh, M. H. Qasem, and A. Alzaqebah. Requirements prioritization techniques comparison. *Modern Applied Science*, 12(2):64–65, 2018.

[4] pandas development team. pandas.dataframe - pandas 1.5.2 documentation, 2023. Accessed: 2024-04-25.

[5] Plotly. Performance. `https://dash.plotly.com/performance`, 2024. Accessed: 2024-04-17.

[6] Python Software Foundation. sqlite3 — db-api 2.0 interface for sqlite databases, 2023. Accessed: 2024-04-25.

[7] PyTorch. Pytorch documentation, 2023. Accessed: 2024-04-25.

A Manual

This guide provides step-by-step instructions on how to install and run the Docker-deployed system. Please follow the steps below to set up your environment correctly.

A.1 Installing Docker

1. Download Docker for your operating system from the official Docker website: `https://www.docker.com/products/docker-desktop`
2. Run the installer and follow the on-screen instructions to complete the installation.
3. Once installed, open a terminal or command prompt and verify the installation by running: `docker -version`. This command should display the Docker version installed on your system.

A.2 Running the Dockerfile

1. Navigate to the directory containing your Dockerfile.
2. Build the Docker image by running: `docker build -t deepcase-dashboard .`
3. Once the image is built, run the container by executing: `docker run -p 8050:8050 deepcase-dashboard`.

A.3 Accessing the Application

1. Open your web browser.
2. Enter `http://localhost:8050` in the address bar.
3. Press Enter, and you should now have access to the application running on your local machine.

A.4 Further Assistance

1. For further questions, please consult the user manual available, which can be found on the deployed dashboard. This manual provides detailed information on system functionalities.

A.5 Using Test Files

1. Sample test files, such as `sample_data.csv`, can be found in the `data` folder within your repository.
2. You can use these files, to start running Deepcase.

B Source code

You can find a git repository containing the source code to the dashboard using the following url: https://github.com/Kristupasc/DeepCASE-Dashboard