

Group 8

SCS: Log Analysis and Attack Graph Visualisation



SAGE Dashboard

Design Report

Georgios Stournaras s2664747
Daan Luth s2737124
Danila Bren s2615908
Miglena Pavlova s2717972
Selin Mehmed s2666782

Supervisors:
Azqa Nadeem
Andrea Continella
Thijs van Ede

Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente
Bsc. TCS M11 Design Project

Abstract	4
1. Introduction	4
2. Domain Analysis	5
2.1 Client and other Stakeholders	5
2.2 SAGE	5
2.3 Current Solution	5
2.4 Research	6
3. Requirements Engineering	7
3.1 Agile Project Management Approach	7
3.2 Weekly Sprints	7
3.3 Requirements Formulation	7
3.4 Requirements Prioritisation	7
3.5 User epics	7
4. Design	8
4.1 Initial Design	8
4.2 Final Design	8
4.2.1 Global Design	8
4.2.2 Dashboard Design	10
4.2.2.1 Structure	10
4.2.2.2 Sidebar	10
4.2.2.2.1 Search	11
4.2.2.2.2 History	11
4.2.2.2.3 Sorting/Filtering	11
4.2.2.3 Attack graph view	11
4.2.2.3.1 Graph	12
4.2.2.3.2 Details	12
4.2.2.3.3 Episode info	12
4.2.2.3.4 Timeline	13
4.2.2.4 Overview	13
4.2.2.5 Thumbnails	14
4.2.2.6 Settings	14
4.2.2.7 Miscellaneous	14
4.2.3 Design Choices	15
4.2.3.1 Implementation of user epics	15
4.2.3.2 Sidebar	15
4.2.3.3 Tab navigation	15
4.2.3.4 Attack Graph Syntax	15
4.2.3.5 Attack Graph + Timeline	16
4.2.3.6 Urgency	16
4.2.3.7 Overview	17

4.2.3.8 No access control	17
4.2.3.9 Urgency settings not affecting the attack graphs	18
4.2.3.10 Look and feel + Logo	18
4.2.3.11 Recommendation System	18
5. Testing	19
5.1 Unit Tests	19
5.2 User Acceptance Testing	19
5.2.1 Testing with Internal Users	19
5.2.2 Testing with External Users	19
5.2.3 Measuring Non-Functional Requirements	20
5.3 Results	20
6. Future Work	21
6.1 Open Requirements	21
6.2 Documentation	21
6.3 UI Testing	22
6.4 Code Quality	22
6.5 Documentation	22
7. Evaluation	23
7.1 Collaboration within team	23
7.2 Communication with client and supervisor	23
7.3 Issues and Challenges	23
8. Bibliography	24
Appendix A - Requirements	25
Appendix B - Test Plan & Results	28
Appendix C - User Acceptance Testing & Results	41
Appendix D - UI Mockups & Screenshots	48
Appendix E - Logo	52
Appendix F - Sprint Reports	53

Abstract

This report is for the Design Project course of the University of Twente. For the project, the "SAGE Dashboard" system was created for the SCS Department of the University. The report contains details about the design process, including the domain analysis and research, the requirement formulation process, the initial and final design, along with explanations on the design choices and system structure.

1. Introduction

The Semantics, Cybersecurity & Services (SCS) team at the University of Twente works on innovative online services with improved quality through context-alignment and reduced security and privacy threats. They work on methods and techniques for requirements conceptualization, architecture design, and model-driven engineering of service systems. The group also develops service ontologies and service composition frameworks to realise semantic interoperability and meaningful enterprise services. To protect these services against cyberattacks, they develop algorithms and protocols that secure the underlying IT infrastructure and help to detect or thwart attacks.

Runtime monitoring is essential to identify and respond to cyberattacks. As such, several tools exist to collect activity logs by monitoring applications and operating systems. However, such logs are usually hard to process and understand, because of the high volume of alerts contained in them.

The SAGE (IntruSion alert-driven **Attack Graph Extractor**) tool can reconstruct attack graphs from logs, in order to more easily visualise the different steps conducted during a cyberattack. SAGE assists analysts in discovering the steps how the attack was conducted and which services were exploited so they can take appropriate action.

Even though the clutter that analysts have to navigate in order to study attacks is reduced, the tool generates a lot of attack graphs. Being a CLI tool, it does not provide any navigation through the attack graphs.

This project aims to extend the functionality of SAGE and provide a dashboard that takes as input a set of activity logs, reconstructs attack graphs, and visualises the resulting graphs, allowing analysts to navigate through different graphs and compare their features. In addition to that, the dashboard will also provide a way to prioritise graphs, highlighting the ones that are more important to the analyst.

2. Domain Analysis

SAGE is designed to assist security analysts, therefore the graphical user interface (GUI) we are developing will target security analysts as users. Security analysts prevent cyber threats by analysing previous attack attempts, conducting risk assessments, and implementing protective measures. Their responsibilities include monitoring security alerts and investigating attackers' actions. That is why security analysts need a tool for visualising the attack graphs, highlighting the detected attacker strategies. By understanding their roles, responsibilities, and needs, our GUI strives to make their work more efficient with its intuitive interface.

2.1 Client and other Stakeholders

The clients are Azqa Nadeem (one of the authors of SAGE), Thijs van Ede and Andrea Continella. Other stakeholders are SIEMENS, and (indirectly) network infrastructure attackers. The main problem to be solved by the proposed system is that analysts use up time searching through attack graphs that are not interesting to them. Hence, the main objective is to present “interesting” graphs to the analyst, and to allow them to search down to specific ones if they know what they are looking for. Therefore, the client’s interests are typically centred around ease of use and efficiency. Analysts should not have to “fight” the tool to get the information they are looking for. This also includes the fact that the interface should visualise the data in a clear and intuitive way.

2.2 SAGE

SAGE [\[1\]](#) utilises S-PDFA (Sequential Probabilistic Deterministic Finite Automaton) machine learning models to analyse security alerts and generate attack graphs, which show the possible attack paths that attackers took (or could take).

The tool takes in network intrusion detection system alert logs, and generates sequences of “episodes”, which correspond to single actions attackers took, such as scanning, data exfiltration, or credential brute force attacks. Those episode sequences are used to generate attack graphs, which are then rendered using graphviz dot, and stored on disk.

2.3 Current Solution

A security analyst must be able to easily find the graphs that they are looking for and quickly glean what might be happening in the network. Unfortunately, the current solution makes it too time consuming to do either; graphs are generated into a directory as PNG files and are very convoluted, and there is no good way to navigate between the different graphs or see what graphs might be most relevant to the security analyst. In addition, when the graphs contain multiple attackers and actions, they become even more cumbersome and hard to read.

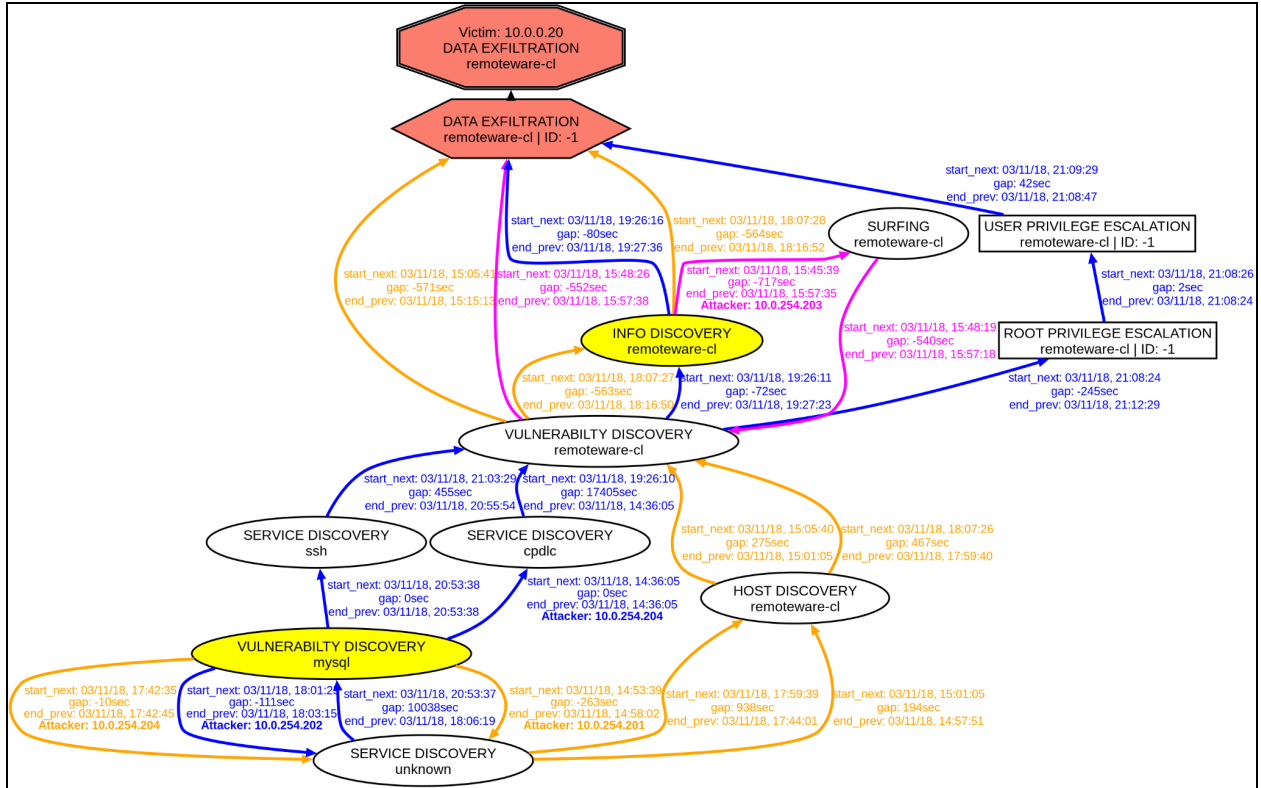


Figure 2.1: Sample output of the SAGE tool.

2.4 Research

To develop our system, we consulted the original SAGE paper [1] and a prior approach in creating a dashboard for the SAGE tool [2]. These provided us with inspiration for a general structure of the dashboard and ideas for a summary of the data. In addition, we also explored the security dashboard in the research of Bedhammar and Johansson [3], which inspired the idea of the timeline implementation.

3. Requirements Engineering

3.1 Agile Project Management Approach

Agile methodology enabled us to realise our project by breaking it down into manageable tasks, allowing for flexibility and adaptability throughout the process [\[5\]](#). Through iterative cycles, namely our weekly sprints, we were able to regularly review and adjust the requirements.

3.2 Weekly Sprints

Regular meetings with the client were conducted every Thursday, which structured the project into 7 sprints. Each sprint consisted of an initial planning stage, where requirements were formulated and prioritised. Furthermore, GitHub Issues were created in the repository that facilitated the development stage. The weekly meetings served as a sprint review, ensuring continuous adjustments.

3.3 Requirements Formulation

The requirements were formulated using the S.M.A.R.T. framework, ensuring they were Specific, Measurable, Achievable, Relevant, and Time-bound [\[6\]](#).

3.4 Requirements Prioritisation

The functional requirements were separated into sections for the different views we have. Each section of requirements was prioritised using the MoSCoW method. This way the essential features were addressed first, “should-haves” and “could-haves” were deprioritized, whereas certain requirements that were outside the project scope were identified early on. This method of categorization and prioritisation helped us organise our development process.

3.5 User epics

While the requirements helped us gain a clear understanding of the project's goals, it was essential to explore the end-user perspective as well. The following user epics provided us with a clear understanding of what the user expects from the system:

1. As a security analyst, I want to be able to find the relevant paths from the graphs when I know what I am looking for.
2. As a security analyst, I want to be able to find relevant information on an attack by analysing the graphs when I do not know what I am looking for.

4. Design

Our solution is the SAGE Dashboard. A web-app that encapsulates the SAGE tool and provides users with an intuitive interface to upload and process their network logs with SAGE, navigate the attack graphs produced, access details per episode, isolate attack paths within graphs, provide attack timing information and prioritise and rank graphs based on the user's preferences.

4.1 Initial Design

After a few brainstorming sessions and studying the SAGE paper, we came up with a few mockups for the dashboard. The main problem that we needed to solve was choosing how to visualise the information. However, the issue was that it was hard for our clients to decide if the visualisation technique was good or not without first seeing it in action. Therefore, we needed to use sample data in a working visualisation to adequately demonstrate potential ideas and started the implementation of the dashboard a bit earlier than expected and with less planning, as having a working prototype was required to continue the requirement capturing process.



Figure 4.1: (Left) Early concept of the dashboard. (Right) Early prototype of SAGE Dashboard.

The first version of the SAGE Dashboard featured hard-coded sample data, one loaded experiment and attack graphs. The user could select the attack graph and it would appear in the viewing area. Details for the nodes were provided when clicking on them. A preliminary version of searching/filtering the graphs was also available.

4.2 Final Design

4.2.1 Global Design

The system is meant to be used by (potentially multiple) analysts working with the same data (NIDS logs from their internal networks). The SAGE tool is a bit difficult to install on a Windows workstation, and so we chose to implement the system as a web app.

On the server-side, we chose Python to interface with SAGE; this was the obvious choice as SAGE is written in Python. A downside of using Python is that it can be slow, but preliminary testing suggested that this would not cause a bottleneck later.

We also opted to use the Flask library to implement the web server and API, because of the ease of development it offers, and because of prior familiarity within the team.

The whole system is meant to be deployed as illustrated in the following diagram:

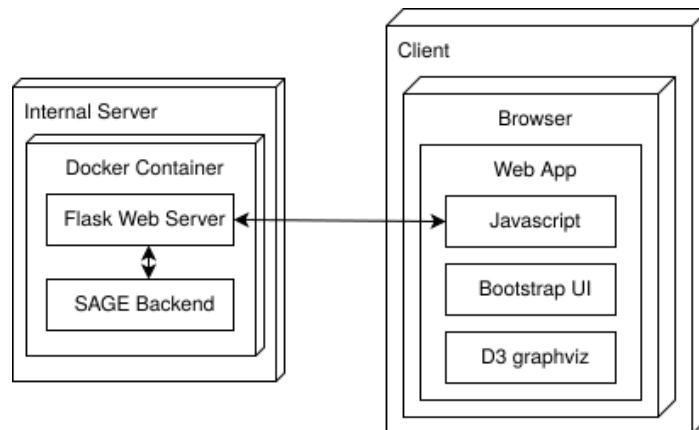


Figure 4.2 : The expected deployment scenario

The client runs a web app which uses Bootstrap 5 and JavaScript for the page layout, and communicates with a relatively small backend which interfaces with the SAGE tool, and stores its output.

The client then downloads these results from the server and uses JavaScript and d3.js to visualise the attack graphs and related analytics.

The backend is wrapped in a docker container for ease of deployment, but can also be run directly on a (Linux) server. The Flask web server can be run in development mode, but should be deployed to a WSGI server in production. We don't use HTTPS to secure the communication between the server and client, but an SSL certificate could be added if the data sent across (NIDS logs, experiment results) are deemed important enough.

The backend allows our system to hook into SAGE before the AG generation, and groups the episode sequences into attack graphs, each of which has a number of "attack attempts", grouped by attacker IP. This allows us to render the graphs with our own syntax and format. Throughout our system we are using the term "experiment" to refer to a single run of SAGE. An example of a generated episode object is:

```
{'start': 1541284006.515861, 'end': 1541284012.524229, 'stateID': -1, 'micro':  
'DATA_MANIPULATION', 'port': 80, 'service': 'http', 'signatures': ['ET WEB_SERVER  
Possible SQL Injection Attempt UNION SELECT']}
```

It contains a start and end timestamp, attack stage, port, and NIDS signatures.

4.2.2 Dashboard Design

4.2.2.1 Structure



Figure 4.3: Annotated dashboard layout

The main structure of the dashboard consists of the sidebar on the left, the viewing area on the right and the tab view on top. Most actions are initiated from the sidebar and the viewing area adapts to whatever has been selected.

4.2.2.2 Sidebar

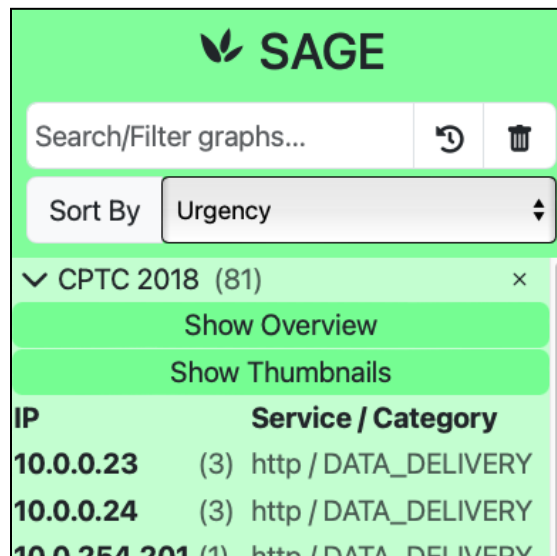


Figure 4.4: The dashboard sidebar.

The sidebar contains the list of loaded experiments created by the user (for example CPTC 2018). Each experiment can be expanded to reveal the list of attack graphs. Clicking on an attack graph will create a new tab and open it in the viewing area. Under each experiment, buttons for the overview and the thumbnail view are provided. Users can also rename and delete experiments from the sidebar.

4.2.2.2.1 Search

Using the Search/Filter bar, users can filter attack graphs based on a number of attributes like number of attack attempts, IPs, services exploited, type of attack, timestamps and more. Users can also combine attributes to create more complex queries. The search bar provides suggestions for completing a query based on the data of the expanded experiments.

4.2.2.2.2 History

The Search/Filter bar has a history button that allows users to see up to 5 of their last queries and quickly reapply them. Searches can also easily be deleted with the delete button.

4.2.2.2.3 Sorting/Filtering

Users can sort their graphs based on a number of ranking techniques like based on Urgency (link), ascending victim IP, Graph Centrality and number of Attack Attempts.

4.2.2.3 Attack graph view

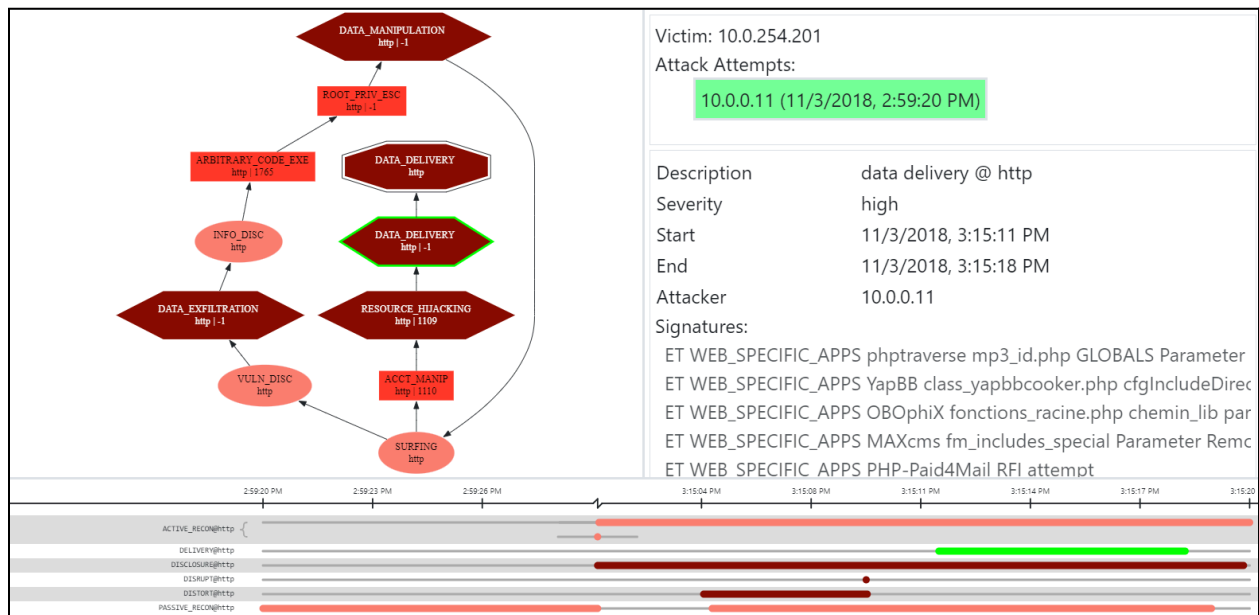


Figure 4.5: The attack graph view that opens when a user selects an attack graph from the sidebar.

4.2.2.3.1 Graph

The graph view shows an attack graph produced by SAGE. The attack graph is located on the left side of the screen (normally just to the right of the sidebar). It is both zoomable and movable for easy navigation.

Each node represents an episode in the attack process, and contains information about the type of attack and the service targeted. If multiple edges are present, that indicates that there were multiple attempts and/or multiple attackers.

4.2.2.3.2 Details

On the top right of the viewing area, general information about the attack graph is shown, including the victim IP and all attack attempts. If there are multiple attempts and/or multiple attackers, they will be grouped by their IP. Clicking on an attempt, will highlight their path on the attack graph. When an attack path is highlighted the timeline at the bottom is also adjusted to show only the timing of this specific attempt.

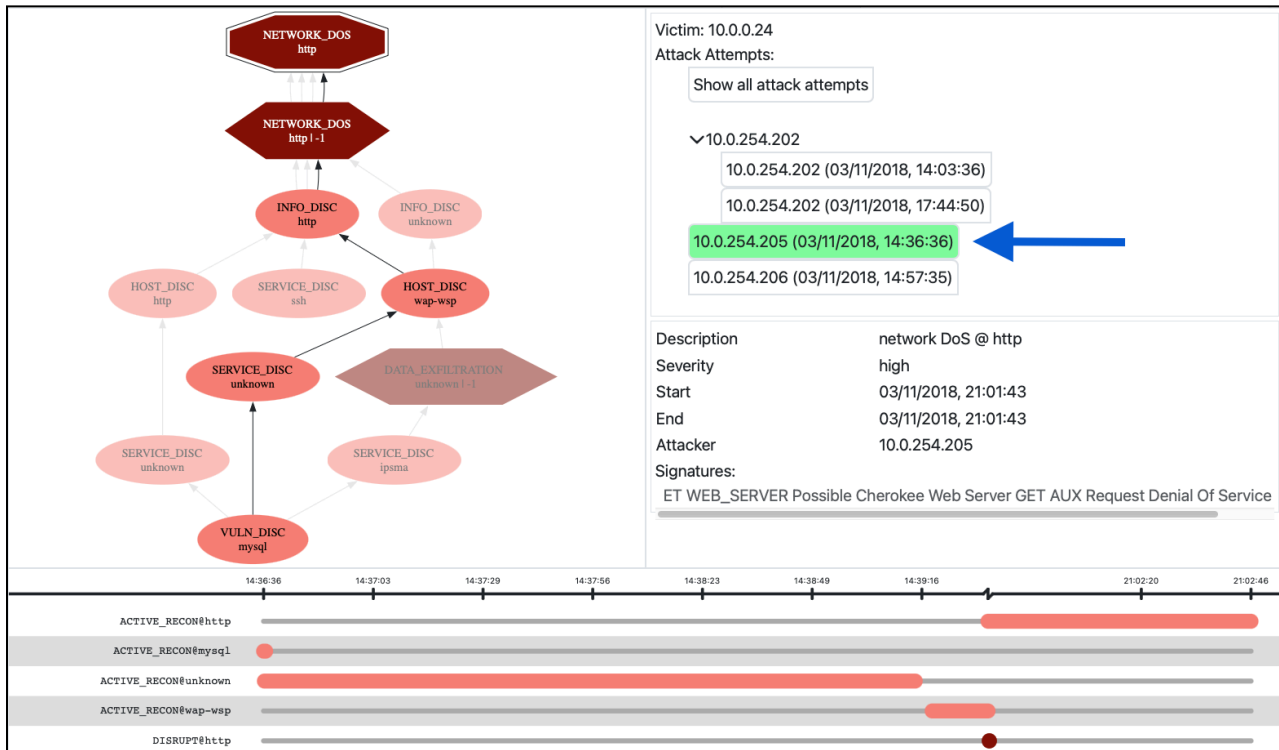


Figure 4.6: Clicking on an attack attempt, highlights it in the attack graph and changes the timeline accordingly.

4.2.2.3.3 Episode info

Clicking on a node in the attack graph will select and show detailed information about the episode on the right. Such information includes descriptions, severity levels, timestamps and signatures. The episode will also be highlighted on the timeline.

4.2.2.3.4 Timeline

The timeline shows the timing of the events within the attack graph. An attacker can repeat multiple steps and do multiple steps at the same time. The timeline groups episodes by attack type and service. It provides a time scale at the top, and the ability to set custom timestamp windows at the bottom. Alternatively, users can zoom in and out and scroll the timeline. Clicking on an episode will also select the corresponding one in the attack graph and trigger the same information to appear.

4.2.2.4 Overview

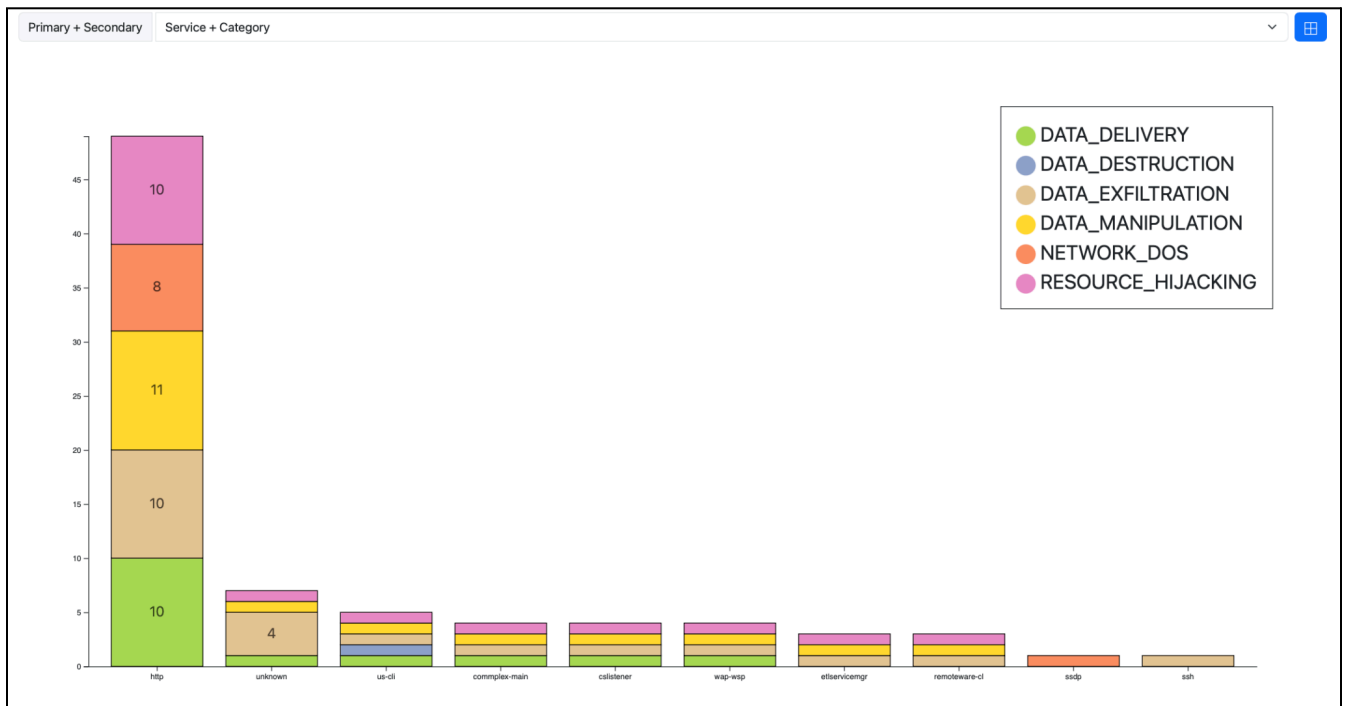


Figure 4.7: The overview for an experiment, here the Service + Category chart.

The overview provides a summary and statistics for an experiment. Users can select between different charts that provide insight on the data in a graphical and intuitive way. The charts are interactive, allowing users to click on axes, slices and the legend to apply filters and narrow down the available attack graphs. This also works the other way around; users can type filters on the search bar of the sidebar and narrow down the data displayed in the overview. The overview also adapts based on the available data, to provide meaningful charts to the user.

4.2.2.5 Thumbnails

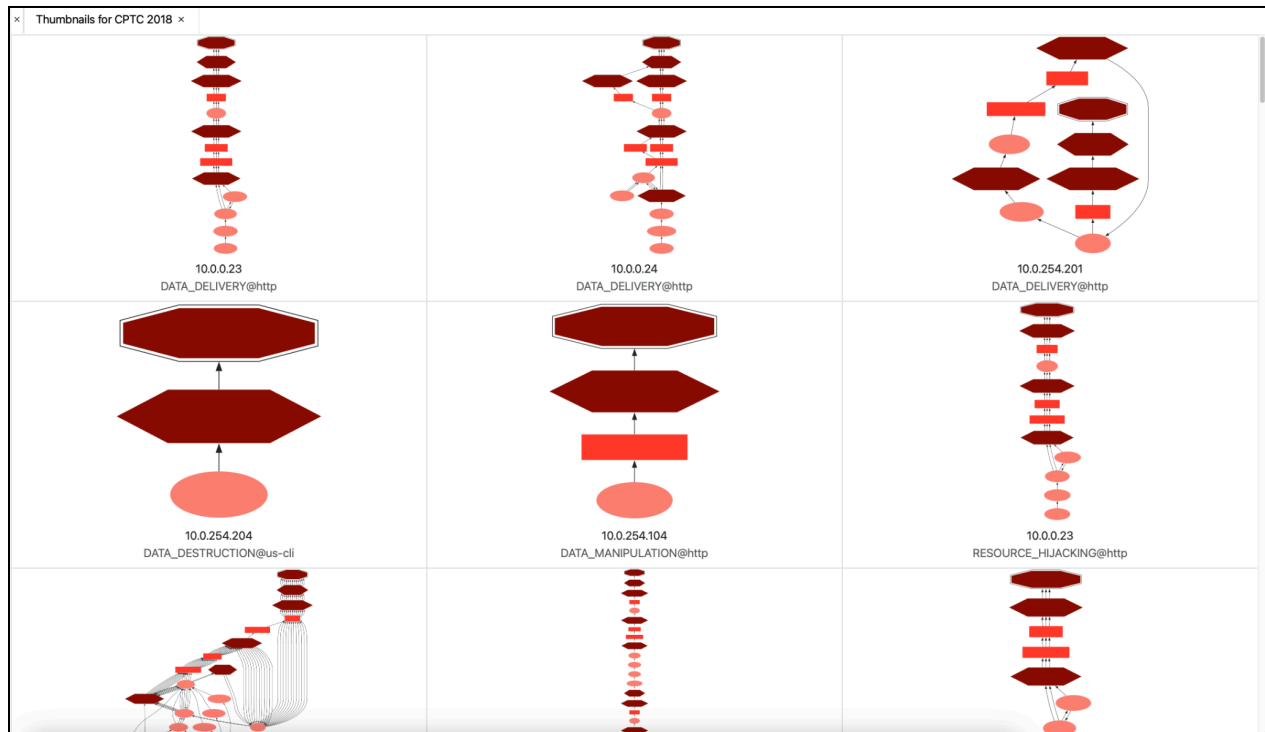


Figure 4.8 : The thumbnail view showing attack graphs for an experiment.

The thumbnail view allows users to view the same attack graph list as in the sidebar, but in an expanded graphical way. The view contains a grid of attack graphs that have reduced thumbnails of the attack graphs. Clicking on one opens the same attack graph view.

4.2.2.6 Settings

The settings page allows users to set custom weights and change the urgency of different attack types, influencing the ranking when sorting by urgency. This allows users with different priorities in their systems to make the recommendation system work for them.

4.2.2.7 Miscellaneous

Some additional nice-to-haves features of the dashboard include:

- the tab system which allows users to quickly navigate through previous views,
- the views being resizable to accommodate for different screen resolutions and aspects,
- and, finally, a dark mode option.

For the full list of features, please refer to the manual.

4.2.3 Design Choices

4.2.3.1 Implementation of user epics

As mentioned before, we had two user epics for this system:

1. As a security analyst, I want to be able to find the relevant paths from the graphs when I know what I am looking for.
2. As a security analyst, I want to be able to find relevant information on an attack by analysing the graphs when I do not know what I am looking for.

Our solution to these is:

1. Searching and filtering attack graphs on specific attributes like the IPs, timestamps, services and more and being able to isolate attacker paths in an attack graph.
2. The Overview provides a summary and statistics for an experiment that provides the analyst with insights regarding the attacks that can lead to a further more elaborate search of the attack graphs and paths. In addition, the recommendation system, which ranks the graphs by urgency, sorts attack graphs based on the analyst's preferences, which can be modified in the settings menu.

4.2.3.2 Sidebar

We based the dashboard on the structure of the SAGE tool. The SAGE tool creates an experiment that contains all the attack graphs when a set of logs is uploaded. Therefore, the highest object in the hierarchy is an experiment. So, our sidebar shows all the loaded experiments first. Since almost all functionality of the dashboard is provided for an experiment or a contained attack graph, we thought that the sidebar should always be visible and act as an anchor to navigate between experiments and attack graphs.

4.2.3.3 Tab navigation

The implementation of a tab-style navigation was the most appropriate for our application. Since the sidebar is always visible and acts as a starting point and should always be accessible, the idea of having a page-like navigation did not make sense. Instead, every action that starts from the sidebar creates a new tab while keeping all previous actions still available for quick access to the user. With this in mind, a universal "back button" navigation system would not be easy to implement. We had some feedback from the clients that this would be a handy feature, but ultimately jumping the user from different tabs would be rather confusing.

4.2.3.4 Attack Graph Syntax

The syntax of the attack graphs is the same as the one provided from the SAGE tool. However, for enhanced visibility, we augmented it with a consistent colour scheme, where the nodes are coloured shades of red, with the darkness indicating the urgency of the associated episodes. The

urgency is also indicated by the shape of the node; these shapes were taken from the SAGE paper's AG syntax.

In order "root node, high, medium, and low urgency":



Figure 4.9: The attack graph syntax for the dashboard

We started with green-blue-red as the colours, but this did not make much sense, as urgency is not a categorical variable, and distinct colours give the impressions of distinct events. Also, using green for the low-urgency nodes gave the impression that these were indicating something unimportant or even good, which is misleading. We now use these three shades of red to indicate urgency, which we ensured to be clearly distinguishable.

4.2.3.5 Attack Graph + Timeline

A challenge we were faced with when deciding the main method of displaying the attack information was that an analyst is both interested in the steps of an attacker and also how they were executed. The SAGE tool combines timing and attempt information all within one graph, and that hinders its visibility. And the problem arises with repeated steps and concurrent steps. Everytime we tried adapting the graph to accommodate these issues we ended up making a sort of "timeline graph" but it doesn't provide glanceable attack information. So, using ideas from our research, we decided to combine an attack graph and timeline view. The attack graph contains the set of steps with unique nodes, regardless if they are repeating, and the timeline shows the repeating and concurrent steps, grouping them by attack type and service. If the analyst wishes to know more information about a step/episode, clicking on the timeline brings up the information.

4.2.3.6 Urgency

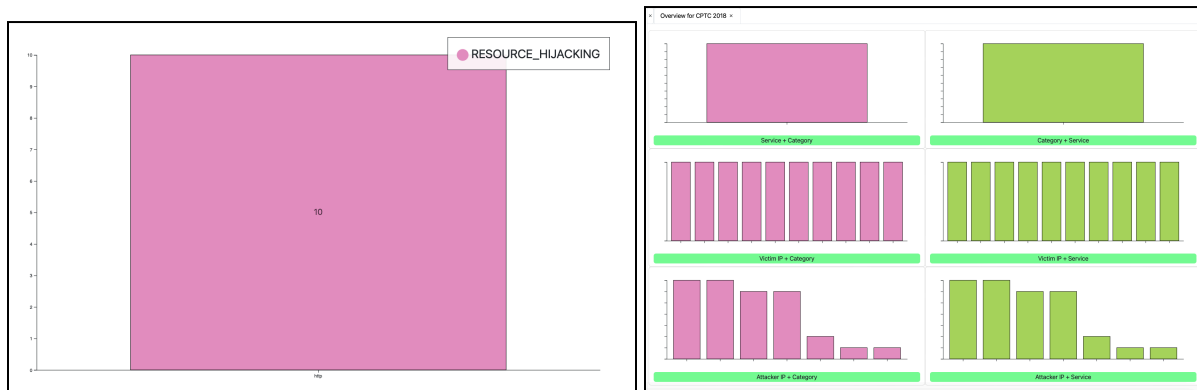
For our second user epic, we needed a way to prioritise graphs in the sidebar. For the first versions of the dashboard, we ranked them based on ascending victim IPs. We went through multiple definitions and applications of a ranking by urgency with our client, as there are not any universal definitions for it. Ultimately, our client provided us with their definition of what is considered an urgent attack graph, and we implemented this formula. For extra flexibility, users can modify the weights and urgency levels of each attack, but not the implementation.

Note that the urgency settings only affect the graph list, and not the shapes of rendered graph nodes. This is because this feature would require completely re-rendering each graph shown before the settings update, which is too computationally expensive. The alternative - only rendering graphs that were created after the settings update correctly, could confuse the user.

4.2.3.7 Overview

The overview is meant to summarise and provide statistical intel on an experiment's data. The data available has three dimensions; IP, service and attack type. There does not exist a simple method to show all dimensions at once, that wouldn't require a 3D graph, which would be difficult to navigate. So, we show two dimensions per chart. The user can select one of six data orders: 'service + attack category', 'category + service', 'victim IP + category', 'victim IP + service', 'attacker IP + category', and 'attacker IP + service'. These orders are based on questions the analyst may have, such as "which service was attacked the most?" (service + category), or "which victims were attacked on subnet 10.0.0.0/24?" (search 'ip==10.0.0.0/24' and select 'victim IP + ...').

Another issue we encountered was that when the user would click on a slice of the bar in a chart, to select a victim IP and a service (for example, 10.0.0.4 and "http"), the overview would apply filters in the search bar to narrow down the attack graphs. As we've explained, the filters also influence the overview, so we would get a giant solid square in the resulting chart. There technically is nothing wrong, but it is wasted screen space. To solve this, we created a similar thumbnail view that we used for the attack graphs for all the six different charts. This view is shown automatically to the user when a square chart is detected, along with a message informing the user why the view changed. The clients really liked this new view so we added a button in the overview that would show the thumbnails on command.



Figures : (Left) Filter combination results in a square of one attacked service and attack type combination. (Right) The dashboard automatically switches to the grid view to show the user that other visualisations are available.

4.2.3.8 No access control

The dashboard has no users, no logins and it's technically not built with security in mind in terms of access control. Everything is meant to be running on a company internal network, so there is no need to include authentication and authorization functionality. This line of reasoning also caused us not to enable HTTPS in the final version, although that could be done fairly straightforwardly.

4.2.3.9 Urgency settings not affecting the attack graphs

Changing the urgency weights and levels in the settings menu only affects the recommendation system and the ranking of the graphs. While the client expressed that the original attack graphs still displayed their original urgency per attack type, changing them would require a complete re-rendering of all attack graphs. That would change their layout and look and would cause confusion to the user, especially if they are using the Thumbnail view of the graphs. Re-rendering all graphs at once may also be computationally infeasible while keeping the UI responsive.

4.2.3.10 Look and feel + Logo

We went for a modern look and feel that is simple and efficient, using the resources from bootstrap. The main colour of the dashboard and of most the buttons is green. Since our attack graphs include a lot of red, a complimentary colour like green would help differentiate the UI from the attack graphs. In addition, “sage” is also a plant, which is green.

The SAGE Dashboard logo is inspired by the leaves of a sage plant. For the bigger logo, the three leaves represent the different urgency levels (LTR; high, medium and low) and the corresponding colours that we chose, all being grounded by the green sage colour. Other logo variants and the process of realising the current one can be found in Appendix [E](#).

4.2.3.11 Recommendation System

The recommendation system works to order the graphs in the graph list (and thumbnail grid) by one of the following metrics:

- “Urgency”: For this metric, the average severity over all nodes in the graph (not counting duplicates) is used to order the graphs, with the highest score at the top. This is the formula we used:

$$U_g = (|V_{high}| * W_{high} + |V_{med}| * W_{med} + |V_{low}| * W_{low}) / |V|$$

This is the basis of our “when the analyst doesn’t know what to look for”, but there is no universal urgency metric or formula. So, one of our clients, a security analyst, provided us with what they define urgency as, given the SAGE attack graphs.

- “Graph Centrality”. For this metric, the “betweenness centrality” of each node in the graph is computed, and multiplied by the severity score (low,medium,high) of the node. The average of these values is then used to order the graphs (highest score at the top).

This metric is intended to prioritise “bottlenecked” attack graphs, where analysts could then apply targeted measures to stop the attack at the bottleneck.

We used Brandes’s Algorithm to compute betweenness centrality of a node [\[4\]](#).

- “Victim IP”, which orders the graphs by victim IP (i.e. “1.2.3.4” comes before “1.2.4.3”).
- “Attack Attempts”, which orders graphs by the number of attack attempts (descending).

5. Testing

For our system we ran two categories of tests; Verification and Validation. Verification, to make sure that the agreed requirements were implemented and function correctly, and Validation to make sure that we are building the system that the client wants.

For the Verification testing, we ran automated unit tests and manual scenario-based checks, which we also ran with the client in the weekly meetings. For the Validation testing, we ran User Acceptance Testing (UAT) with internal and external users.

5.1 Unit Tests

We made JavaScript unit tests that cover:

- The tab management system.
- The generic highlighting system (used in the graph view, histogram, and graph list)
- Common utility code (date-timestamp conversion, IP-string comparison).
- Search/filter pattern parsing and autocompletion-related code.

And Python unit tests that cover:

- Common utility code.

5.2 User Acceptance Testing

For our User Acceptance Testing (UAT), users tested the main functionality of the Dashboard, excluding the installation. We tested with internal and external users. For both types of users the process was as follows: they were provided a version of the SAGE Dashboard along with the manual and while testing they expressed comments and notes about their experience, which we documented. Afterwards, we processed these comments and categorised them into bugs, feature requests or out-of-scope.

5.2.1 Testing with Internal Users

UAT that was conducted with internal users used a subset of our client team. Internal users were already familiar with the Dashboard and were also part of the development process, so we did not conduct any onboarding.

We followed an exploratory approach; users were provided with the Dashboard and the manual, and were allowed to use the system however they desired, making notes of their experience.

5.2.2 Testing with External Users

For our external users, two cybersecurity PhD students were selected. Our external users were not security analysts or familiar with SAGE and its syntax, so a quick onboarding was provided. Unlike our internal users, we used a scenario-based approach for their testing. This way, the intuitiveness of the system could also be tested, something the internal users did not qualify for since they already knew how the system works.

5.2.3 Measuring Non-Functional Requirements

At the end of both sessions, we asked users some questions about their experience that would help us determine if the non-functional requirements were completed. Users responded using a likert scale (Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree).

5.3 Results

Most of the feedback from Internal and External UAT was positive, highlighting how the major interactions with the Dashboard were intuitive and provided useful functionality. The majority of the negative feedback was mostly on discovered bugs and some visual inconsistencies, with lastly, some requested features. The external users provided almost the same feedback as the internal users, with some comments being rejected as they arose from the users not being familiar with the SAGE output and terminology.

The identified bugs and inconsistencies were scheduled to be resolved in the following sprint, and the feature requests to be further discussed with the clients. More information about the results and the testing process can be found in Appendices [B](#) and [C](#).

6. Future Work

While we managed to create a sufficient product that was accepted by our clients, there are some areas left incomplete and opportunities for extending the functionality of the dashboard. To address this, we've ensured comprehensive documentation and designed key components of the codebase to facilitate easy modification.

6.1 Open Requirements

The following (Should/Could level) requirements are left unfulfilled:

7. The dashboard should have a navigation bar to navigate through the different views.
 - Having universal back and forward arrows would not work with our current tab system, and this tab system largely serves the same purpose as a navigation bar would.
18. The graph nodes could follow a hierarchical structure.
 - Graphs are rendered using graphviz in the browser, and we were (are) not sure how to reliably enforce such a hierarchical structure in the graphviz language.
19. The graph view could compare attack paths and highlight differences.
 - This was not done due to time constraints.
20. The graph view could automatically detect duplicate attack paths.
 - This was not done due to time constraints.
21. The graph view could display multiple attack graphs.
 - This was not done due to time constraints.
30. The timeline could have an attacker perspective which displays multiple swimlanes for all the victims a given attacker targeted.
 - This was not done due to time constraints.
40. The overview chart could be normalised so that all bar slices are clearly distinguishable.
 - This was not done because we could not think of a good way to do this while still keeping the semantic meaning of the barchart slice heights. The situation that this requirement was meant to prevent can also be resolved by using the search feature.

Of these, (19), (20), (21), and (30) represent major features that were not implemented.

6.2 UI Testing

Automatic UI testing was considered, but not implemented due to time constraints. As such, we are reasonably sure that the UI code works correctly for the common use cases, but bugs may be found with more extensive testing.

6.3 Code Quality

The code is divided into modules (graphviz, timeline, ui, overview, etc.) which can be reused, but the code structure within some of the modules (most notably the timeline) is more complex than necessary.

Common code was extracted into the “utils” namespace, but some patterns could still be abstracted over, such as accessing episodes within a graph object.

6.4 Documentation

Documentation was generated from JavaScript docstrings using JSDoc. To the generated documentation we added technical overviews and call graphs for the frontend and backend. A sequence diagram was also included to show typical client-server interactions.

7. Evaluation

7.1 Collaboration within team

Our team was divided into two teams:

Management: Megi, George, Selin - Front-end design, requirements, communications, planning, documents, testing.

- Wireframing, CSS, requirements specification with client, creating tasks/user stories for Development team, documentation (project), test plan, conducting user testing.

Development: Daan, Danila, Selin - Back-end design, implementation.

- Implementation of requirements, (code) documentation, testing.

The established division of labour within our team has demonstrated its effectiveness in both the creation and fulfilment of project requirements. However, it was observed that the roles assigned to each member closely mirrored their own inclinations. It was noted during our discussions that it potentially hinders the interchangeability of roles. Additionally, this kind of assignment works to decrease learning potential, because everyone tends to do what they were already good at.

7.2 Communication with client and supervisor

To manage requirements and get regular feedback from the client we scheduled weekly one hour long meetings with our supervisor and clients. During the final weeks of the project our main client was not available, so instead we scheduled meetings with Thijs van Ede or Andrea Continella.

During those meetings we went through requirements that were not marked as complete previously and discussed design choices for features to be implemented. During that review, new requirements could be added to improve the product. In most cases we had mock-ups of how the feature could look like so that our supervisor could indicate their preferences and suggest changes or possible improvements.

7.3 Issues and Challenges

SIEMENS could have been part of the UAT (User Acceptance Testing), but due to planning issues, we were not able to conduct testing with them. External UAT used PhD students, but the best scenario would have been other security analysts.

While we were developing the dashboard, a change in FlexFringe (a dependency of SAGE) caused our docker container build process to break. We tried to resolve this, but the change also broke SAGE itself, and so we decided to not use the latest version of FlexFringe for the final system.

8. Bibliography

- [1] Nadeem, A., Verwer, S., Moskal, S., & Yang, S. J. "Alert-driven attack graph generation using S-PDFA". *IEEE Transactions on Dependable and Secure Computing*, 19(2). (2021).
<https://doi.org/10.1109/tdsc.2021.3117348>
- [2] Díaz, Sònia Leal, et al. "Critical Path Prioritization Dashboard for Alert-Driven Attack Graphs." Universidad Carlos III de Madrid & Delft University of Technology, 19 Oct. 2023.
<https://arxiv.org/pdf/2310.13079>
- [3] Bedhammar Jennifer, and Oliver Johansson. "Visualisering Av Cybersäkerhetsangrepp." Linköpings Universitet, 1 June 2020.
<https://www.diva-portal.org/smash/get/diva2:1448554/FULLTEXT01.pdf>
- [4] "Betweenness Centrality", The University of Texas at Austin. 2014-2018. Accessed: 20 Mar. 2024. https://iss.odan.utexas.edu/?p=projects/galois/analytics/betweenness_centrality
- [5] "Agile Project Management for Software Teams." Atlassian, 2024. Web. Accessed: 05 Feb. 2024. <https://www.atlassian.com/agile/project-management>.
- [6] Mannion, Mike, and Barry Keepence. "SMART Requirements." *ACM SIGSOFT Software Engineering Notes*, vol. 20, no. 2, Apr. 1995, pp. 42–47, doi:[10.1145/224155.224157](https://doi.org/10.1145/224155.224157).
- [7] "5.2.17 'Service Validation & Testing.'" *ITIL® Foundation ITIL 4 Edition*, TSO, Norwich, 2019, pp. 160–161.

Appendix A - Requirements

Functional

Dashboard View

Must Have

1. The dashboard must open the Overview for the latest experiment by default.
2. The dashboard must display the Recommendation System in the sidebar as a list.
3. The dashboard must display the Overview as a button option for every experiment in the sidebar.
4. The dashboard must open an attack graph view and a timeline view when a certain graph is chosen from the recommendation sidebar.
5. The dashboard must accept JSON logs that contain Suricata alerts as input, extract attacker/victim IPs, destination ports, alert signatures and timestamps, run SAGE, and display the results.

Should Have

6. The dashboard should have an attack graph filter with a dropdown of possible filters existing in data and an option for the analyst to type filters themselves.
7. The dashboard should have a navigation bar to navigate through the different views.
8. The dashboard should have an option to remove an experiment from the Sidebar.

Could Have

9. The dashboard could have a search history in the filter dropdown.
10. The dashboard could be able to change the scale of the graph, info and timeline windows.
11. The Sidebar could have a 'Show Thumbnails' button, filling the screen with small previews of the attack graphs, with their IP as the title, in a grid-like structure similar to video platforms.

Will Not Have

- Allowing the user to input logs not in the suricata format.
- Being able to import the data from and to the dashboard.
- Comparing between experiments.

Attack Graph View

Must Have

12. The graphs must visualise severity with different shades of red filling the node and with a uniform shape (low - circle, medium - square, high - hexagon).
13. The graph view must display information about the root (objective) node by default.
14. The graph view must display the following information about a node when it is clicked: Description, Severity, Attackers, Signatures.
15. The graph view must show the paths that different attackers took.

16. Each graph node must display the attack stage and targeted service information within its shape.
17. The graph view must highlight the steps and the path of an attacker, reducing the opacity of the rest of the graph, and display targeted victim hosts, when a particular attacker is clicked from the details tab of the current attack graph.

Could Have

18. The graph nodes could follow a hierarchical structure.
19. The graph view could compare attack paths and highlight differences.
20. The graph view could automatically detect duplicate attack paths.
21. The graph view could display multiple attack graphs.
22. The graph view could display the State ID of a node.

Timeline View

Must Have

23. The timeline must highlight the corresponding node and display their information when a certain event's duration is clicked.
24. The timeline must have swimlane labels based on the macro attack stage and service being attacked.
25. The timeline must show all the attackers' timeline by default.
26. The timeline must have horizontal lines separating swimlanes.
27. The timeline must be able to show the timing of events.
28. The timeline must show using sub-swimlanes overlapping attacks when grouped.

Should Have

29. The timeline should have a filter based on time intervals.

Could Have

30. The timeline could have a victim perspective which displays multiple swimlanes for all the attackers the victim was targeted by.

Recommendation System

31. The recommendation system must allow the analyst to choose between *urgency*, graph *centrality*, *victim IP* and *attack attempts* to prioritise attack graphs.
32. The user must be able to set custom values for the classification of low, medium and high urgency events.
33. The user must be able to set per attack stage a (custom) low, medium and high urgency.

Overview

Must Have

34. The overview must be generated when clicking on the overview button of an experiment, using the experiment's data.

35. The overview must generate a combined bar chart that shows the service attacked (x-axis) and the type of attack (y-axis).
36. The overview chart must create search parameters that automatically search attack graphs in the sidebar, when a certain bar slice is clicked.
37. The overview must allow for filtering of the data that are shown in the bar chart.

Should Have

38. The Overview should have options to display service attacked + type of attack, type of attack + service attacked, Victim IP + type of attack, Victim IP + service attacked, Attacker IP + type of attack and Attacker IP + service attacked.

Could Have

39. The Overview could have the option to display the available bar charts in a thumbnail grid style when only one bar and one slice (1x1 square) is detected in the default view.
40. The overview chart could be normalised so that all bar slices are clearly distinguishable.

Non-functional

Performance

41. The dashboard must respond to user interactions within 2 seconds (The dashboard is fast and responsive).
42. The attack graph view must load within 5 seconds.
43. The timeline view must load within 3 seconds.

Usability

44. The dashboard interface must be intuitive, and analysts must be able to navigate between views seamlessly.
45. Buttons must have descriptive text labels indicating their action.
46. Icons must have intuitive representations of their associated actions.
47. Shapes and colours of the Nodes must have intuitive representations.

Compatibility

48. The dashboard must function consistently across major web browsers including Chrome, Firefox, and Safari.

Maintainability

49. The code must follow standard practices for easier maintenance and future development.
50. There must be documentation to help future developers understand how the system is built and how it works.

Appendix B - Test Plan & Results

1. Unit Tests

For JavaScript, found in /test/tests.js


Test Method	Purpose
ui/tabs	Check that creating, switching, and closing tabs works correctly, and the tab context remains valid.
ui/tab_listeners	Check that the "on_tab_hide" and "on_tab_show" listeners work as intended.
ui/tab_fuzzing	Generate many random inputs to the tab system and check if the behaviour is correct.
ui/highlighting	Check that the "highlightable", "highlight", and "unhighlight" function yield calls to the appropriate event handlers.
utils/assert	Check that the "assert" function throws an error when an assertion in the code fails.
utils/date_timestamp	Check that converting from timestamp to date and back yields the same value.
utils/deep_equals	Check that the "deep_equals" function works as intended.
utils/IP_compare	Check that the IP comparison works as intended for IPv4 and IPv6.
utils/UID	Checks that the "UID" method doesn't return the same value twice.
utils/fire_events	Checks if the "fire_input", "fire_resize", etc. methods work as intended.
utils/populate_extra_keys_twice	Regression test, checks if populate_extra_keys is idempotent (it should be).
search	Verify that tokenisation and parsing work, and also check some search suggestion related functions.

For Python, found in test/test.py

Test Method	Purpose
test_mkdir_rmdir	Check that "mkdir" creates nested directories if needed, and "rmdir" removes directories recursively.
test_json_on_disk	Check that "load_json_on_disk" and "update_json_on_disk" work as intended.
test_random_names	Empirically check that the likelihood of generating the same name from "random_name" is not too large.

2. Verification Testing

Requirement	Input	Expected	Result
Dashboard View			
1. The dashboard must open the Overview for the latest experiment by default.	N/A	When at least one Experiment is loaded, the Overview is displayed automatically.	Pass
2. The dashboard must display the Recommendation System in the sidebar as a list.	N/A	The sidebar is populated by attack graphs which are sorted based on the Recommendation System (Sort By).	Pass
3. The dashboard must display the Overview as a button option for every experiment in the sidebar.	N/A	The sidebar displays the experiment title and the Overview button below it.	Pass
4. The dashboard must open an attack graph view and a timeline view when a	User clicks on an attack graph from the sidebar.	A new tab is opened in the viewing area and the selected attack graph and timeline (only of one	Pass

certain graph is chosen from the recommendation sidebar.		attacker) is shown.	
5. The dashboard must accept JSON logs that contain Suricata alerts as input, extract attacker/victim IPs, destination ports, alert signatures and timestamps, run SAGE, and display the results.	<p>Network logs in the Suricata format are uploaded using the</p>  <p>button and an experiment name is given by the user.</p>	A new experiment is created in the sidebar and after SAGE has completed processing, all the attack graphs are made available.	Pass
6. The dashboard should have an attack graph filter with a dropdown of possible filters existing in data and an option for the analyst to type filters themselves.	The user clicks on the search bar.	A list of available commands drops down.	Pass
8. The dashboard should have an option to remove an experiment from the Sidebar.	The user clicks on the X next to the experiment row and selects Delete from the deletion prompt.	The dashboard confirms the deletion with the user and removes the experiment.	Pass
9. The dashboard could have a search history in the filter dropdown.	The user clicks on the history button in the search/filter bar.	A drop down appears with the user's previous queries.	Pass
10. The dashboard could be able to change the scale of the graph, info and timeline windows.	The user grabs on the columns of the graph view, info view and timeline view.	The windows are resized.	Pass

11. The Sidebar could have a 'Show Thumbnails' button, filling the screen with small previews of the attack graphs, with their IP as the title, in a grid-like structure similar to video platforms.	The user clicks on the "Show Thumbnails" button under an experiment.	A new tab is opened with thumbnail previews of the attack graph list.	Pass
Attack Graph View			Pass
12. The graphs must visualise severity with different shades of red filling the node and with a uniform shape (low - circle, medium - square, high - hexagon).	User clicks on an attack graph from the sidebar.	The attack graph node colours and shape match the severity.	Pass
13. The graph view must display information about the root (objective) node by default.	User clicks on an attack graph from the sidebar.	The root/objective node is selected by default.	Pass
14. The graph view must display the following information about a node when it is clicked: Description, Severity, Attackers, Signatures.	User clicks on an attack graph from the sidebar.	Description, Severity, Attackers and Signatures are displayed for the specific node.	Pass
15. The graph view must show the paths that different attackers took.	User clicks on an attack graph from the sidebar.	Graph contains multiple edges that belong to different attackers and/or attempts.	Pass
16. Each graph node must display the attack stage and targeted service	User clicks on an attack graph from the sidebar.	All the nodes in the attack graph display attack stage and targeted service information.	Pass

information within its shape.			
17. The graph view must highlight the steps and the path of an attacker, reducing the opacity of the rest of the graph, and display targeted victim hosts, when a particular attacker is clicked from the details tab of the current attack graph.	User clicks on an attempt from the details on the right of the attack graph.	The attacker's attempt is highlighted in the attack graph.	Pass
22. The graph view could display the State ID of a node.	User clicks on an attack graph from the sidebar.	The nodes contain the State ID.	Pass
Timeline View			Pass
23. The timeline must highlight the corresponding node and display their information when a certain event's duration is clicked.	User clicks on a timeline event.	The matching node in the graph is highlighted and information about the node is displayed on the right.	Pass
24. The timeline must have swimlane labels based on the macro attack stage and service being attacked.	N/A	The timeline groups attacks based on the attack stage and service attacked.	Pass
25. The timeline must show all the attackers' timeline by default.	User clicks on an attack graph from the sidebar	The timeline shows all the attacks.	Pass
26. The timeline must have horizontal lines separating swimlanes.	N/A	The timeline separates the attack+service groups with horizontal gradient swimlanes.	Pass

27. The timeline must be able to show the timing of events.	N/A	Timestamps legends are shown at the top of the timeline.	Pass
28. The timeline must show using sub-swimlanes overlapping attacks when grouped.	N/A	Sub-swimlanes are shown if there is event overlap.	Pass
29. The timeline should have a filter based on time intervals.	The user can set manual timestamps at the bottom of the timeline	The timeline is scaled based on the intervals set.	Pass
Recommendation System			Pass
31. The recommendation system must allow the analyst to choose between <i>urgency</i> , graph <i>centrality</i> , <i>victim IP</i> and <i>attack attempts</i> to prioritise attack graphs.	User clicks on Sort By: Urgency, Graph Centrality, Victim IP, Attack Attempts.	Attack graphs are sorted accordingly.	Pass
32. The user must be able to set custom values for the classification of low, medium and high urgency events.	The user clicks on the settings icon.	The settings page appears and the user can set values for the weights of <i>low</i> , <i>medium</i> and <i>high</i> urgency	Pass
33. The user must be able to set per attack stage a (custom) low, medium and high urgency	The user clicks on the settings icon.	The settings page appears and the user can change the urgency of the attack stages.	Pass
Overview			Pass
34. The overview must be generated when clicking on the overview button of an experiment, using the experiment's data.	User clicks on an experiment's overview button.	The overview opens in a new tab for this experiment.	Pass

35. The overview must generate a combined bar chart that shows the service attacked (x-axis) and the type of attack (y-axis).	N/A	The overview generates the combined bar chart.	Pass
36. The overview chart must create search parameters that automatically search attack graphs in the sidebar, when a certain bar slice is clicked.	The user clicks on a bar chart slice.	The search/filter bar is inputted with filters that match the slice.	Pass
37. The overview must allow for filtering of the data that are shown in the bar chart.	User types filters in the search bar.	The overview adapts to the filtered data.	Pass
38. The Overview should have options to display service attacked + type of attack, type of attack + service attacked, Victim IP + type of attack, Victim IP + service attacked, Attacker IP + type of attack and Attacker IP + service attacked.	The user clicks on the Primary + Secondary drop-down menu in the Overview.	The user can select between service attacked + type of attack, type of attack + service attacked, Victim IP + type of attack, Victim IP + service attacked, Attacker IP + type of attack and Attacker IP + service attacked.	Pass
39. The Overview could have the option to display the available bar charts in a thumbnail grid style when only one bar and one slice (1x1 square) is detected in the default view.	The user clicks on a slice that will create a 1x1 square.	The overview automatically switches to the grid view and notifies the user.	Pass
40. The dashboard must respond to user interactions within 2	We used the CPTC 2018 and CPTC 2017 datasets that	The dashboard remains responsive and does not lag.	Pass

seconds. (The dashboard is fast and responsive) 41. The attack graph view must load within 5 seconds. 42. The timeline view must load within 3 seconds.	generate a lot of attack graphs.	Views load instantly.	
48. The dashboard must function consistently across major web browsers including Chrome, Firefox, and Safari.	Dashboard was tested with the latest versions of Chrome, Firefox, Safari and Chromium.	Dashboard functionality remained consistent.	Pass

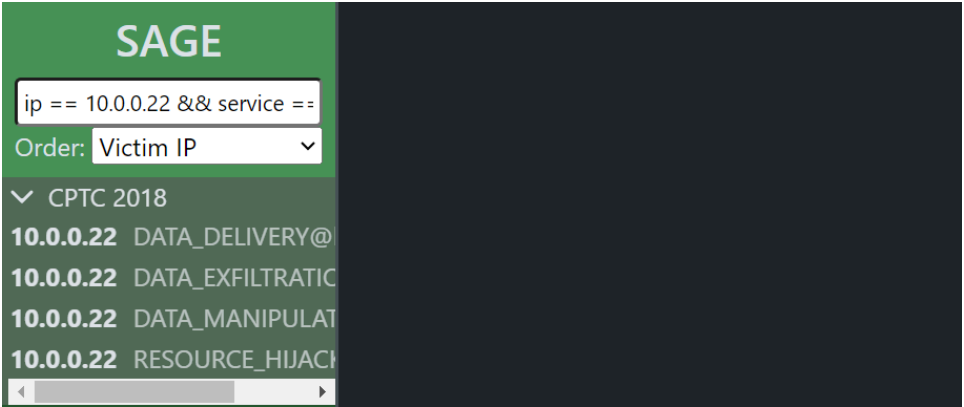
3. (Internal) Scenario-based testing

Scenario: The user wants to analyse attacks on IP address 10.0.0.22 on the HTTP service

Task: The user must be able to filter attack graphs on IP and service

Actual: The user types this filter in the searchbar 'ip == 10.0.0.22 && service == http' and a list of experiments is displayed in the left sidebar

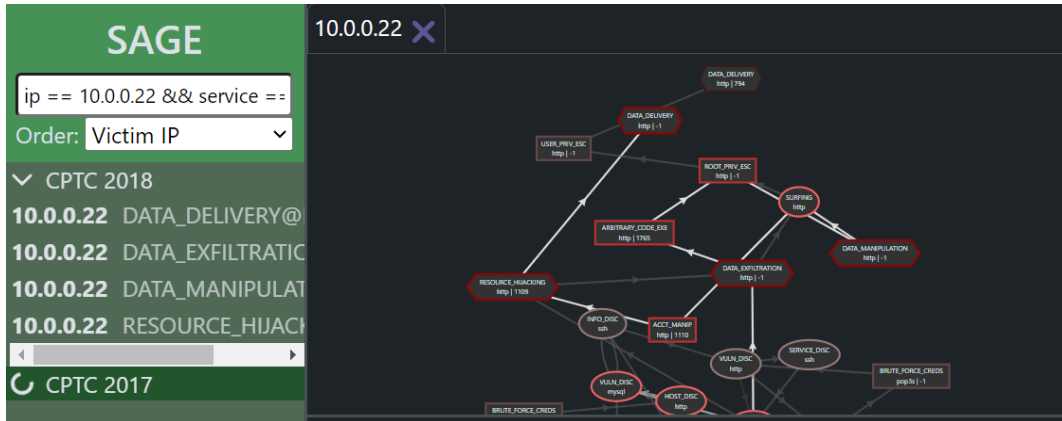
Evidence:



Task: The user must be able to view attack graphs for each experiment

Actual: The user clicks on an attack graph from the sidebar and it is displayed in the content area

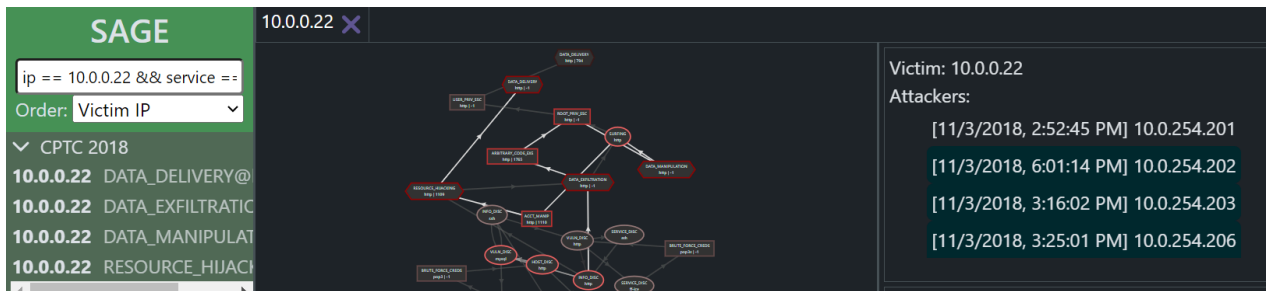
Evidence:



Task: The user must be able to see information about each attack graph

Actual: Along with the attack graph, the following information about the experiment is displayed in the right side of the content area: Victim IP and a list of Attackers IPs

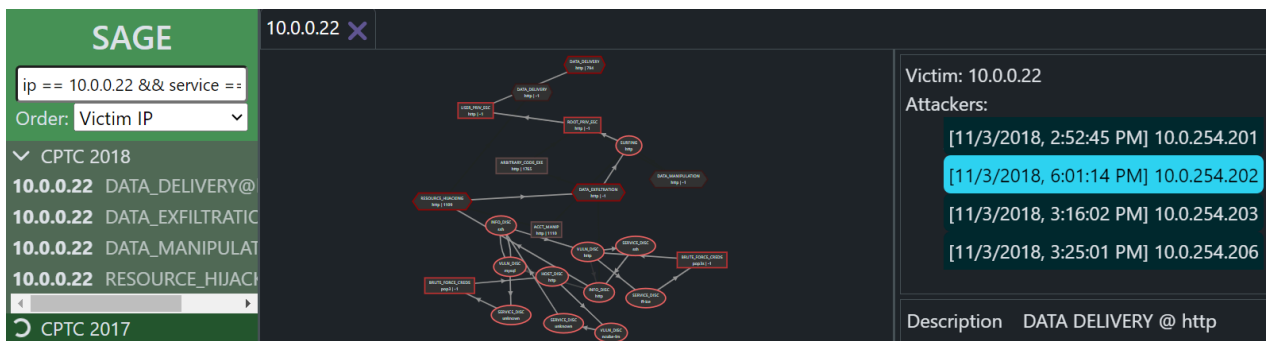
Evidence:



Task: The user must be able to see each step an attacker took

Actual: The user clicks on a certain attacker's IP from the attackers list on the right and this attacker's steps and path are highlighted in the attack graph

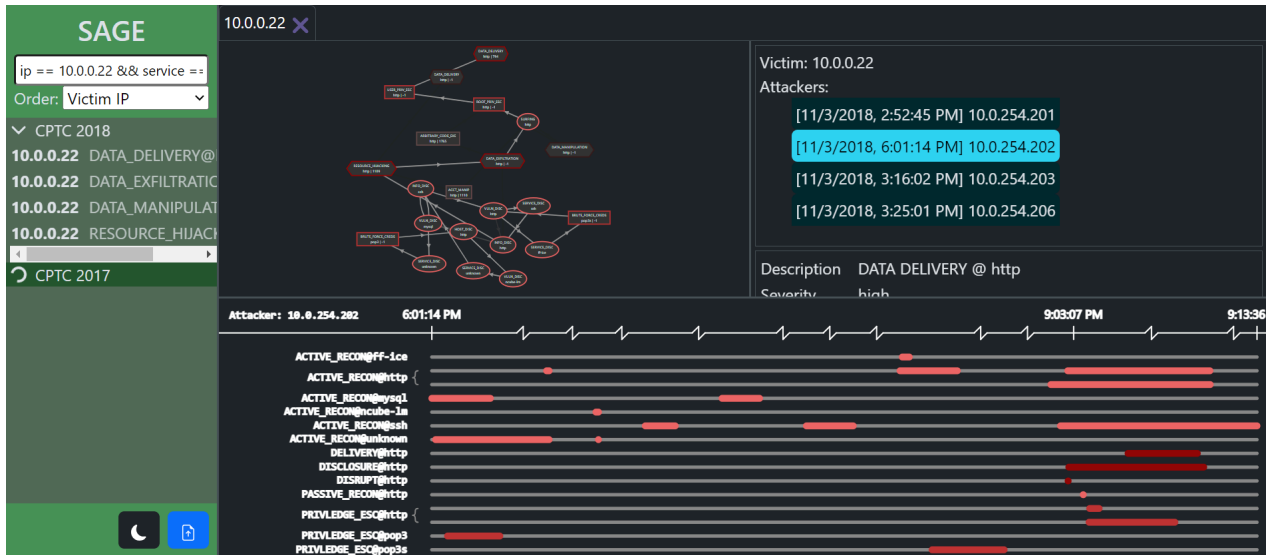
Evidence:



Task: The user must be able to see the timing of each step

Actual: The user clicks on a certain attacker's IP from the attackers list on the right and a timeline is displayed showing the attacker's path and the duration of episodes.

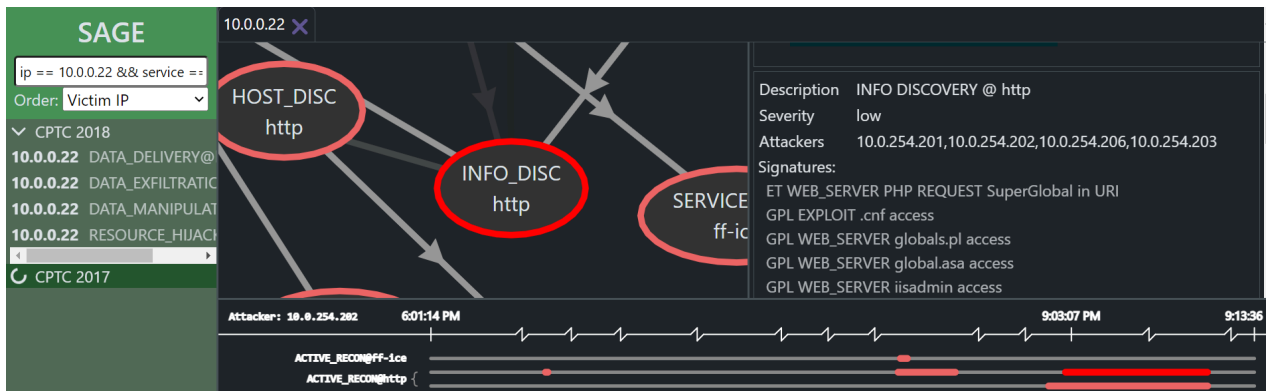
Evidence:



Task: The user must be able to see information about each episode

Actual: The user clicks on a node from the graph or on an episode's duration from the timeline and the following episode information is displayed in the right side of the content area: Description, Severity, Start, End, Attacker, Signatures.

Evidence:

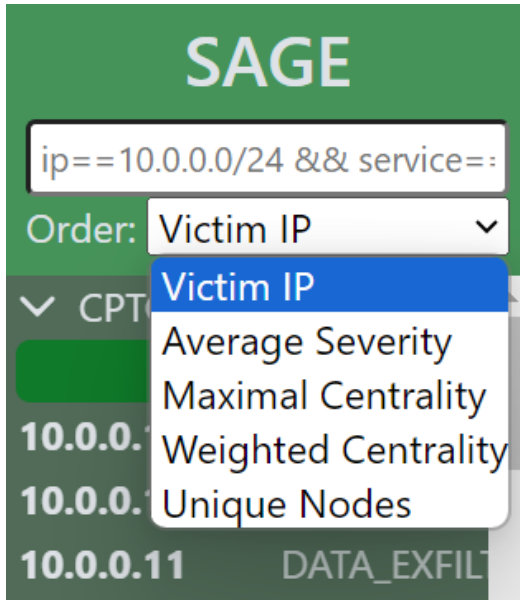


Scenario: The user isn't looking for anything specific and wants to get a recommendation based on urgency.

Task: The user wants the most urgent attack graph to be positioned on top of the list

Actual: The user can sort the list of attack graphs in the sidebar on the left by: Average Severity, Maximal Centrality, Weighted Centrality or Unique Nodes

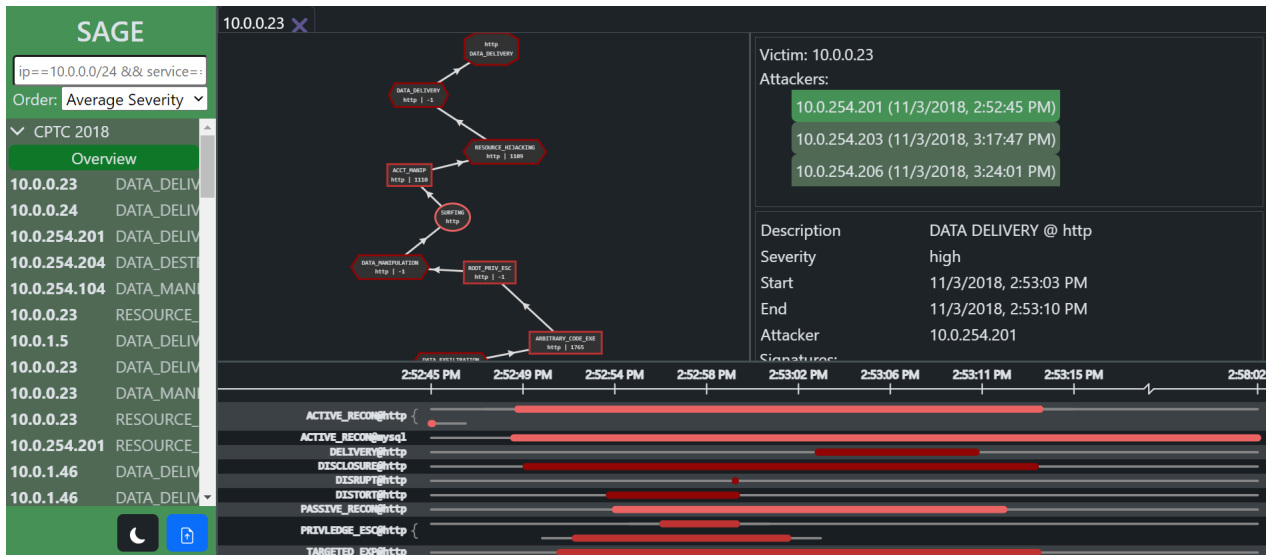
Evidence:



Task: The user wants to analyse the most urgent attack graph

Actual: The user can click on the attack graph on top of the list and the most urgent graph can be analysed from the content area. The attack graph on top depends on what the user chose as definition of 'urgency': Average Severity, Maximal Centrality, Weighted Centrality or Unique Nodes

Evidence:



Scenario: The user wants to identify the most targeted services in the latest experiment and analyse the corresponding attack graphs.

Task: The user must be navigated if they are not looking for anything specific

Actual: An Overview is displayed in the content area on launch, showing the most targeted services and the type of attacks targeting them.

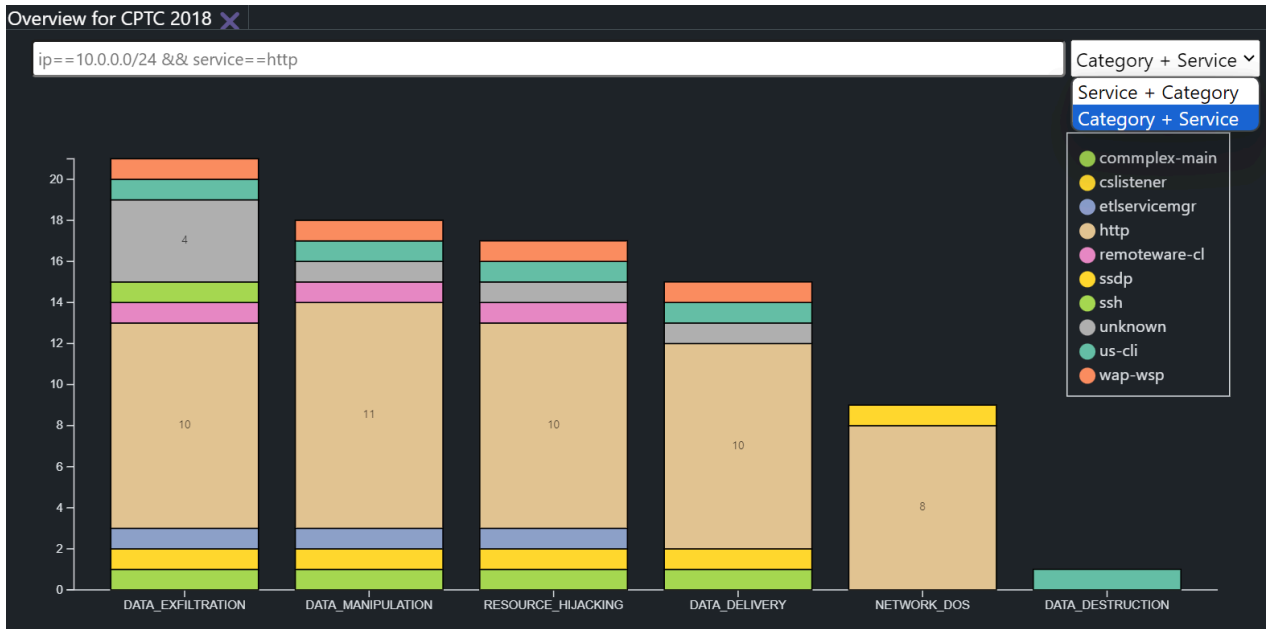
Evidence:



Task: The user must be able to see most targeted services or prevalent types of attacks

Actual: The user can choose between two views of the Overview: 'Service + Category' or 'Category + Service'

Evidence:



Task: The user must be able to see a list of the corresponding attack graphs.

Actual: The user can click on a bar slice from the bar chart and a list of attack graphs is displayed in the sidebar on the left filtered by the corresponding category and service.

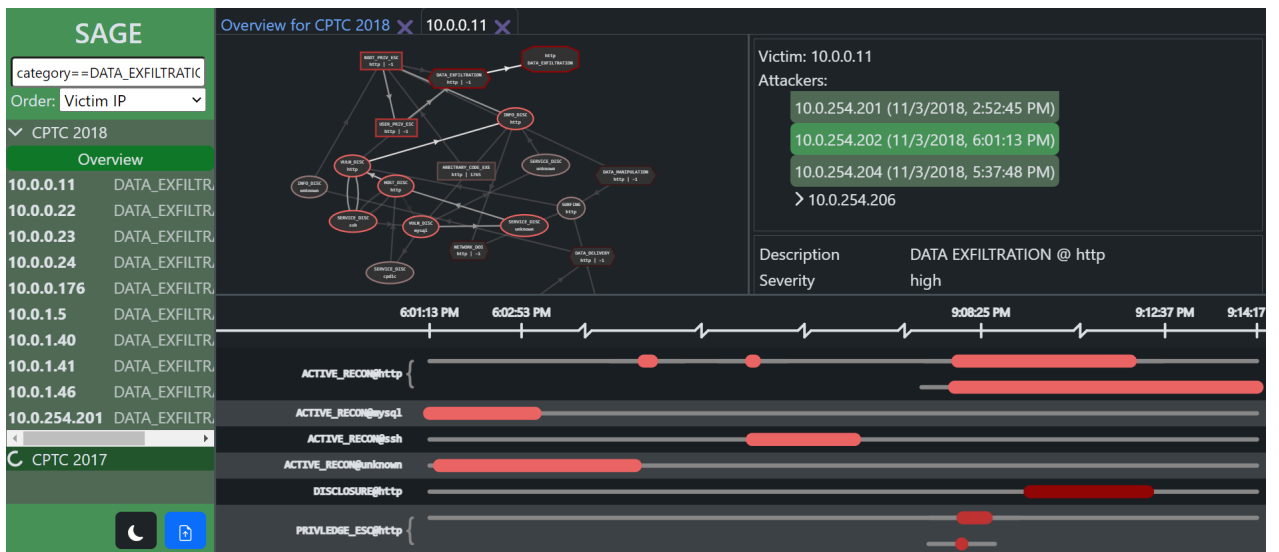
Evidence:



Task: The user must be able to analyse the corresponding attack graphs.

Actual: The user clicks on an attack graph from the sidebar and it is displayed in the content area

Evidence:



4. User Acceptance Testing

Check User Acceptance Testing document.

Appendix C - User Acceptance Testing & Results

Introduction

In order to test the effectiveness of the SAGE Dashboard, we conducted User Acceptance Testing (UAT).

The users will be testing the main functionalities of the system, except the installation. A computer with the latest version of the Dashboard will be provided during testing.

In addition to testing the Dashboard, the effectiveness of the user manual will also be tested.

Test Users

We conducted UAT with internal and external users. For our internal user, a stakeholder was selected. They have been part of the development process, but have never used the system freely.

For our external users, two cybersecurity PhD students were selected. Unfortunately, they did not have prior knowledge of SAGE and its attack graph syntax thereof, but they should be able to provide feedback about the intuitiveness of the system. As our external users are not security analysts, tasks will be provided to them as inspiration on how to use the Dashboard. This is not to test if the tasks can be completed, but if the process of completing them is intuitive.

Test Materials

- Computer running SAGE Dashboard
- User Manual for the SAGE Dashboard
- Notepad

Procedure

Internal Users

- Test users will be provided with the test Computer and User Manual.
- The conductors will run an onboarding demo, explaining and showing the functionality of the SAGE Dashboard.
- The test users will freely use the SAGE Dashboard noting down potential comments and giving feedback. This will give us more insights into how intuitive the system is and if it meets the user's expectations.

External Users

- Test users will be provided with the test Computer and User Manual.
- The conductors will explain the target user type that the external users will be testing as, including what they would want the system to do.
- Tasks will be provided to the users and feedback will be collected about their experience.

Collected Feedback

UAT with Internal Users (Exploratory approach)

Action	Type (positive, bug, feature, out-of-scope)	Comment
Overview Order by	Positive	'It's good'
Opening Attack Graph View	Positive	'Quite intuitive'
Navigating through the Timeline	Positive	'Quite intuitive'
Deselect node by double clicking on it or on the timeline duration	Feature	'Would've been nice'
When a graph is open highlight its row in the list on the left	Feature	'Would be nice for visual confirmation'
Recognising '-1' to be the state id	Out-of-scope	'Not intuitive'
Automatic application of settings	Positive	'Very nice'
Automatic application of settings	Feature	'Would be nice to have a popup'
Zooming in Timeline with scroll wheel	Positive	'Very nice', 'Quite nice', 'Well done!'
Timeline filter	Positive	'Works well'
Search bar filter syntax	Positive	'Intuitive'
Grouping by IP in attacker list	Positive	'Intuitive'
Responsiveness	Positive	'Very good'
Settings tooltips for explanation	Positive	'Good'
Setting custom weights	Feature	'Wasn't intuitive that it's only for sorting and doesn't alter visualisation', 'Maybe change icon'

Close all tabs button	Bug	'Makes the user think it's going to close the whole page', 'Could be larger', 'Could have a label'
Reset all in settings puts dark mode	Bug	'Reset all in settings puts dark mode'
Highlighting a node	Feature	'When clicking on a Timeline item the node could stay highlighted in green and not with halo effect', 'Once selected, keep it highlighted with green'
Severity same as urgency	Bug	'Not intuitive since in setting is called severity weights and in order by is called urgency', 'Consistency between the naming would be nice'
Opening multiple tabs of same graph/overview	Bug	'Not necessary'
Drag and scroll in timeline instead of a scroll bar	Feature	'Should be described in the Manual', 'A bit confusing'
Overall	Positive	'Very intuitive'
Number of attack attempts in list of graphs	Bug	'Not too intuitive'
Sort settings	Feature	'Settings could be alphabetically sorted'
	Bug	'Consistency with capitalised letters'
Description in graph list clickable	Feature	'Description should also be clickable'
Hovering on Timeline item shows highlighted node	Positive	'Nice feature'
Search bar suggestions in dropdown	Positive	'Very nice'

UAT with External Users (Task-Scenarios approach)

	Action	Type (positive, bug, feature, out-of-scope)	Comment
Scenario	Analyse attacks on IP address 10.0.0.0/24		
	Search bar suggestions dropdown	Positive	'Very nice'
	Search bar query hint	Bug	'The query in search feels like the query is entered while it is a just hint on the syntax'
	Finding the goal of the attacker	Positive	'Intuitive'
	Hovering on a Timeline item highlights the corresponding node	Positive	'Very nice'
	Dragging the Timeline	Feature	'It is a bit confusing to scroll the Timeline since it doesn't have scrollbar but you can drag'
Scenario	Find the graph with the most attack attempts		
	Number of attack attempts in list of graphs	Positive	'It is intuitive'
Scenario	Find an individual attempt		
	Grouping by IP in attacker list	Positive	'Makes sense'
	Selecting on an attacker from the list to highlight a singular path	Positive	'Intuitive'
	Showing all paths button	Positive	'Intuitive'
Scenario	Find the graph with the most attempts from category Resource Hijacking		
	Search bar filter on category	Positive	'Intuitive'
	Description in graph list clickable	Feature	'Intuitive but entire line could be clickable'
Scenario	Analyse the Overview		

	Changing Order in Overview	Bug	'Order isn't an intuitive name, maybe use X + Y'
Scenario	Go back to the initial Overview		
	Clearing the filter	Positive	'Intuitive'
Scenario	Analyse the most urgent graph		
	Sorting by urgency	Positive	'Intuitive'
	Setting the severity weights without confirmation	Bug	'Did it apply anything?'"
Scenario	Filter the Timeline on the interval 3:20pm 11th of March to 6 pm 11th of March		
	Timeline filter application	Bug	'Applying isn't intuitive', 'No apply button'
	Reset selection	Positive	'Intuitive'
Scenario	Close all tabs efficiently		
	Close all tabs button	Positive	'Intuitive'
Scenario	Switch to Dark Mode		
	Dark Mode button	Positive	'Intuitive'
	Distinguishing colours in Dark Mode	Out-of-scope	'Dark red is hard to distinguish on timeline especially on dot-like small durations'
Scenario	Resize the view windows		
	Visualisation that resize is possible	Bug	'It would be nice to have an on mouse visualisation that resize is possible'
Scenario	Go back to a previous page		
	History view	Feature	'Good but not intuitive'
	Back button	Feature	'Would be nice'

	Opening duplicate tabs	Bug	'So many tabs', 'If the tab is already open don't need to open again but switch to this open one'
--	------------------------	-----	---

Testing Non-Functional Requirements

The users were given the following statements regarding the non-functional requirements of the system. They were able to rate each of them from *Strongly Disagree*, *Disagree*, *Neutral*, *Agree* to *Strongly Agree* based on their perception of the system's performance.

Both internal and external users answered the same and their answers were grouped.

The dashboard is fast and responsive.

[Strongly Disagree] [Disagree] [Neutral] [Agree] [**Strongly Agree**]

The dashboard interface is intuitive.

[Strongly Disagree] [Disagree] [Neutral] [**Agree**] [Strongly Agree]

Navigating between views is seamless.

[Strongly Disagree] [Disagree] [Neutral] [**Agree**] [Strongly Agree]
 Pretty seamless?

Buttons have descriptive text labels indicating their action.

[Strongly Disagree] [Disagree] [Neutral] [**Agree**] [Strongly Agree]

Shapes and colours of the Nodes have intuitive representations.

[Strongly Disagree] [Disagree] [Neutral] [**Agree**] [Strongly Agree]

Conclusion

Based on the collected data, most of the feedback was positive, highlighting how the major interactions with the Dashboard were intuitive and provided useful functionality. The majority of the negative feedback was mostly on discovered bugs and some visual inconsistencies, with lastly, some requested features. The external users provided almost the same feedback as the internal users, with some comments being rejected as they arose from not being familiar with the SAGE output and terminology.

Bugs to be resolved:

- 'Close all tabs' button will be modified to be more intuitive regarding its functionality.
- The reset weights button in the settings toggles dark mode on and off.

- Inconsistent use of 'severity' and 'urgency' when referencing either.
- Tabs for the same view can infinitely be opened.
- Format of the attack graph list not too intuitive, legend will be added at the top.
- Inconsistent capitalisation in settings menu.
- The search bar displays sample queries by default, but it's not clear if it's a hint or an applied filter.
- The different charts being named "Order" is not clear, but will be changed to X+Y or similar.
- It's not immediately clear that the inner views can be resized, additional thickness of the borders and a on-hover resizing indicator will be added.

Features to be implemented:

- Being able to deselect a node in the attack graph.
- Highlighting the selected attack graph in the Sidebar.
- Having a popup when changes are saved in the settings menu.
- Adding an 'Apply' button for the Interval Filter in the Timeline
- When selecting a node in the attack graph, the node and the timeline item should remain highlighted.
- A scroll bar in the timeline.
- Alphabetically sorting the settings menu.
- The entire row in the attack graph list in the Sidebar should be clickable, not just the IP.
- More intuitive search history

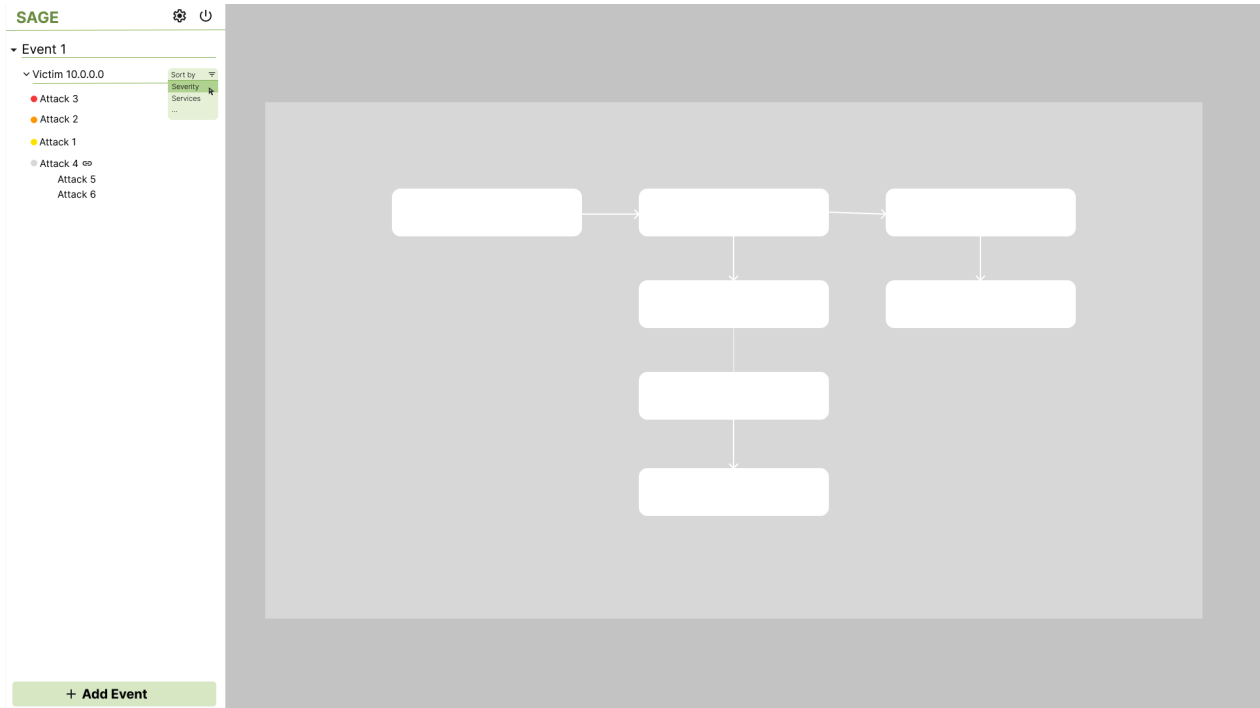
Features to be discussed further:

- Whether the settings menu should alter the weights and severity on both the attack graphs and the ranking, instead of just the ranking. In the latter case, more consistent tooltips to inform the user about what the weights alter.
- Universal back button, since the current implementation uses tabs, a back button is not applicable in some contexts.

Out-of-scope/ Will not be addressed:

- The dark red colors not being visible in dark mode. The current colour scheme has been designed with the client and stakeholders and is designed to have high contrast between the other shades of red. While the shade change is not considered urgent, the small red items in the timeline issue can be resolved with the aforementioned fix of keeping the items highlighted plus the zooming in.

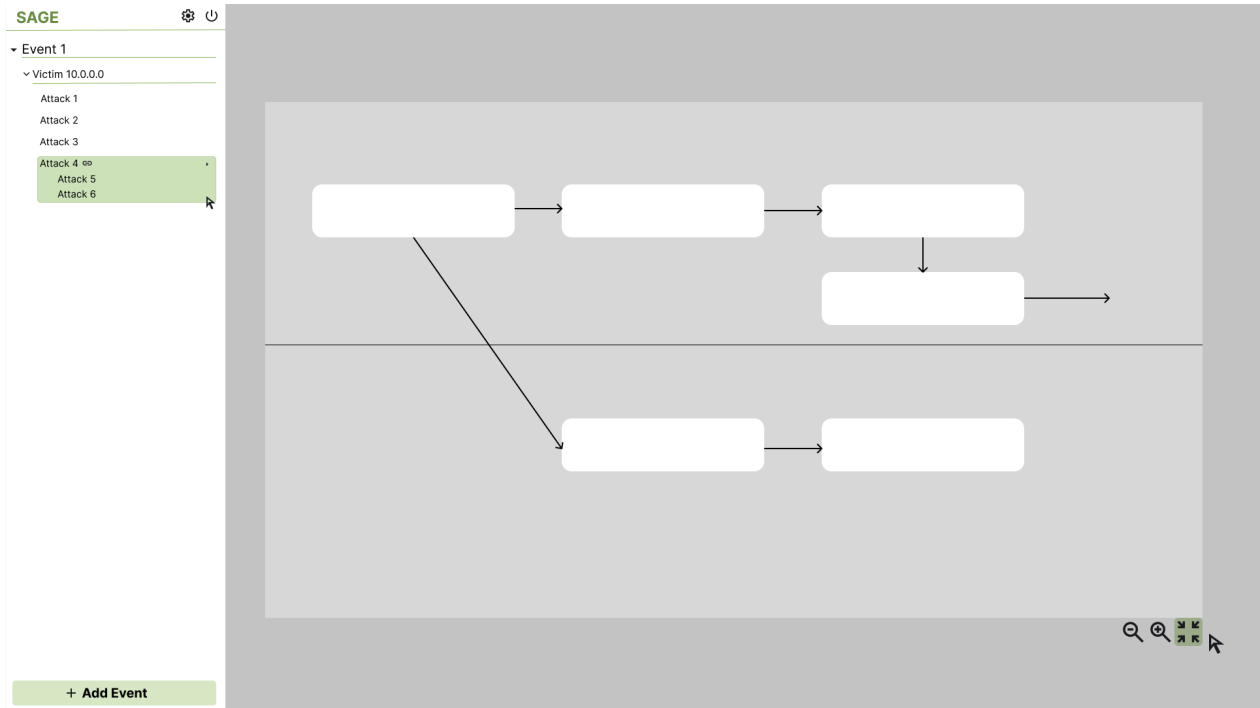
Appendix D - UI Mockups & Screenshots



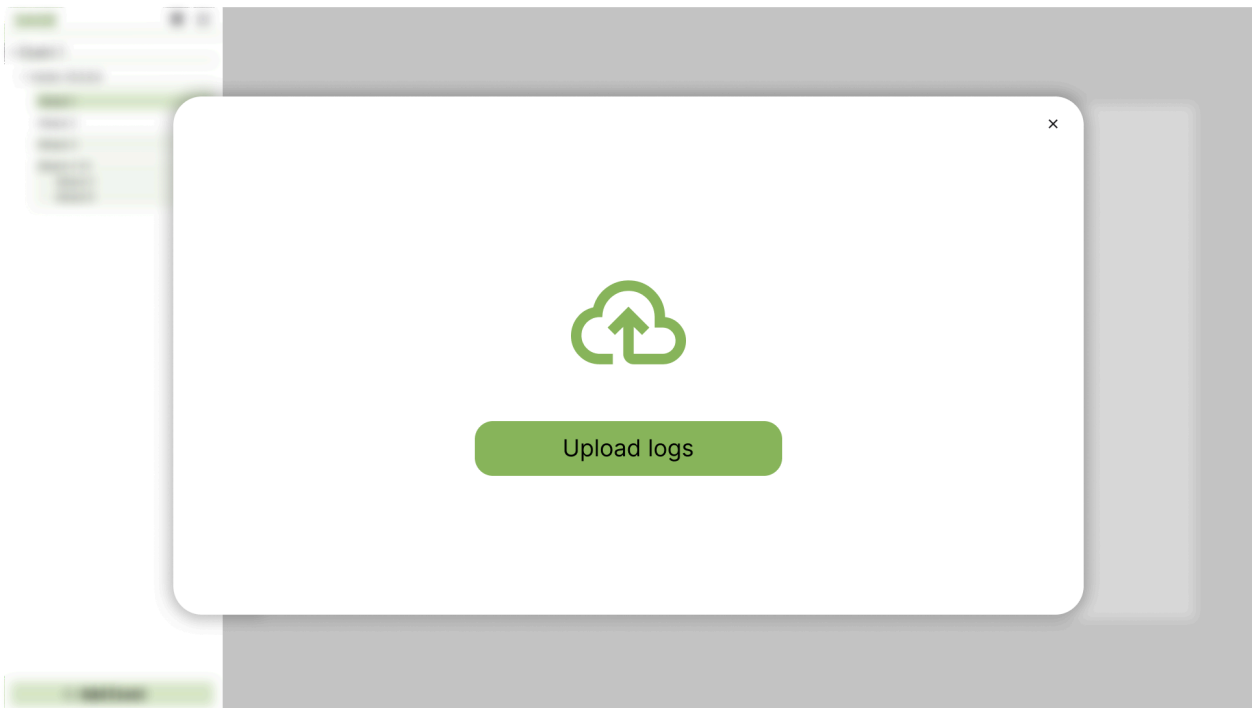
1) Main view with an attack graph open and the sidebar sorts the attack graphs by severity/urgency.



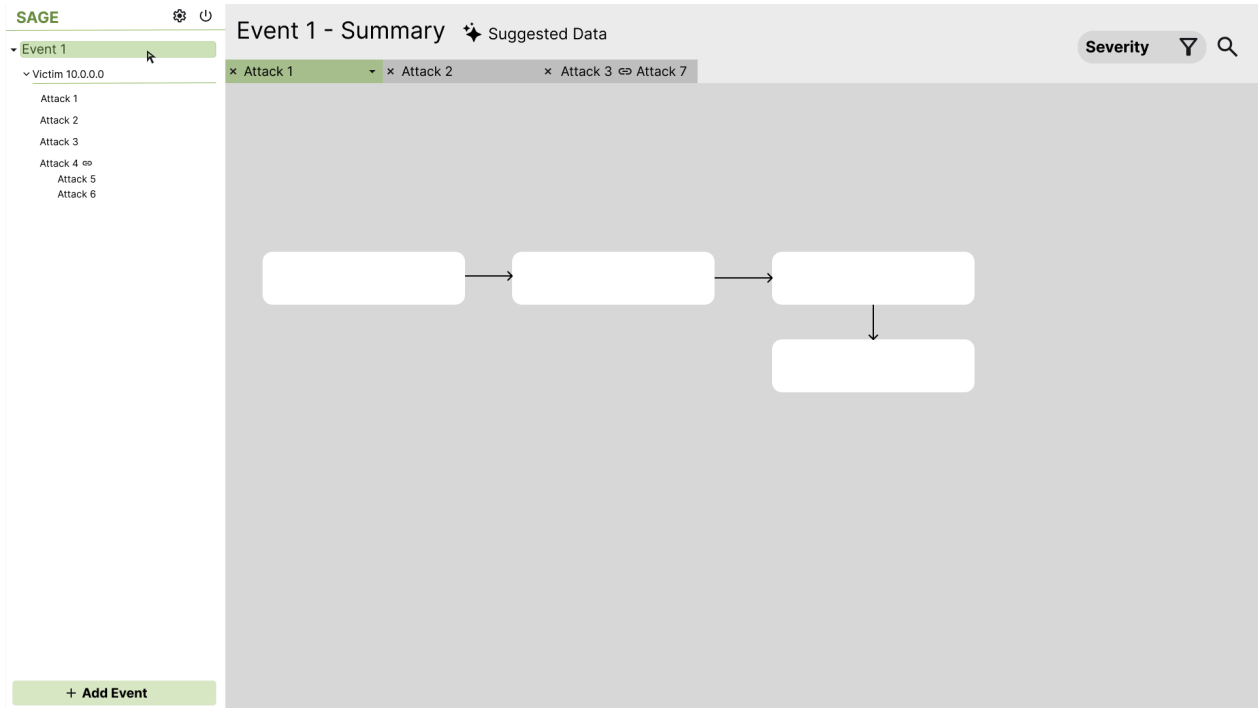
2) Alternate view showing a timeline-like view that combines multiple attack graphs.



3) Other version of a multiple attack graph view.



4) Upload log screen concept.

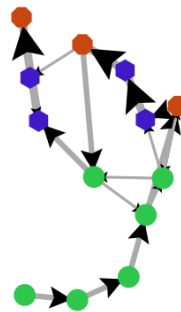


5) Concept of the recommendation system that was realised as the overview later.

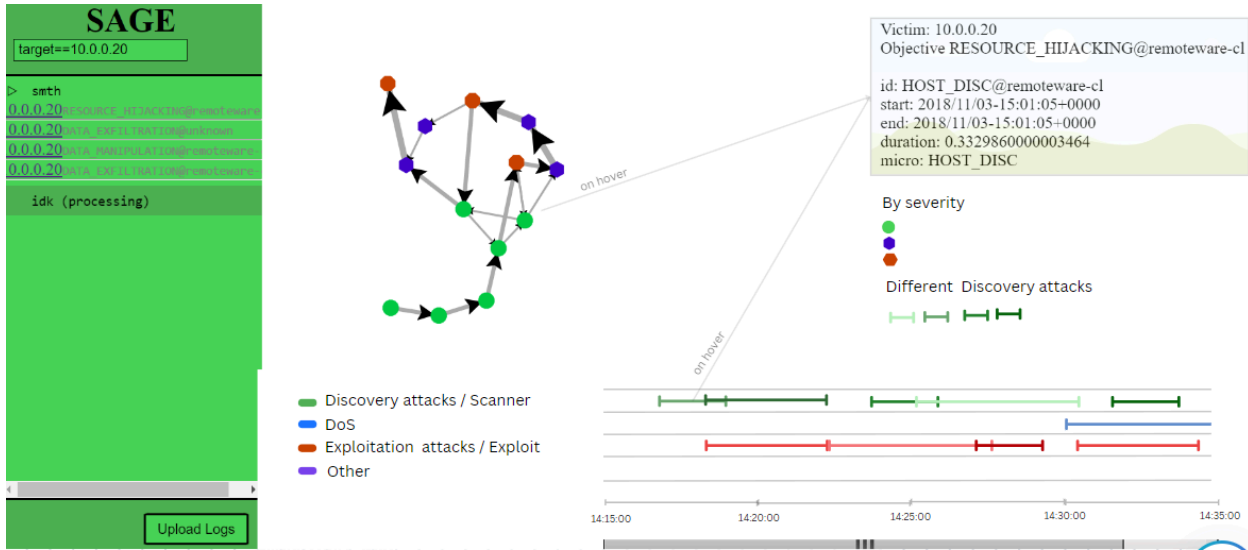


Victim: 10.0.0.20
Objective RESOURCE_HIJACKING@remoteware-cl

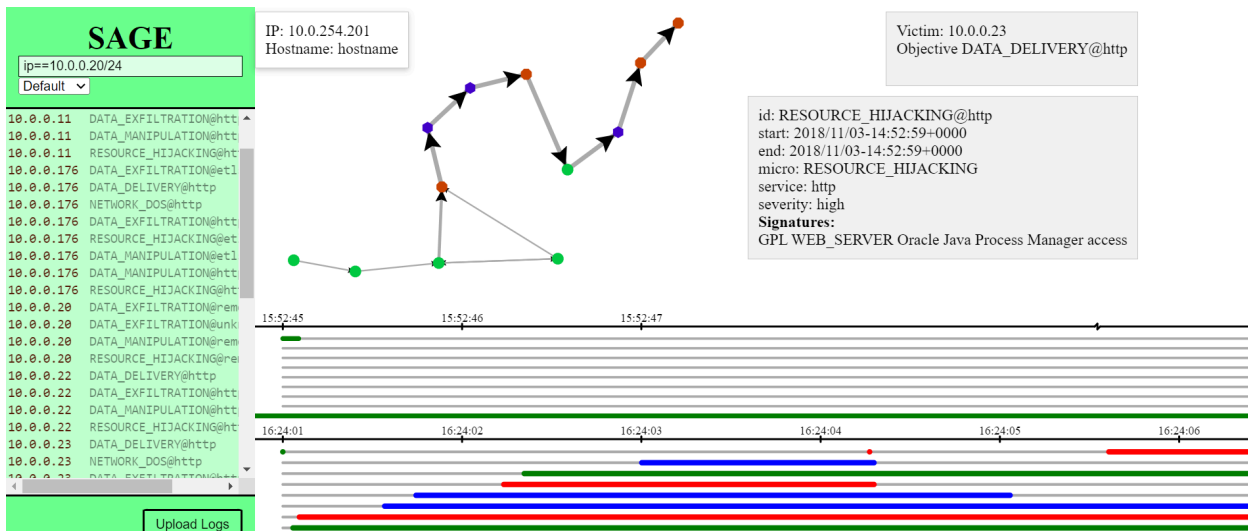
id: SERVICE_DISC@unknown
start: 2018/11/03-14:53:39+0000
end: 2018/11/03-14:57:51+0000
duration: 252.4556219999995



6) Screenshot of the first working prototype.



6) Concept of what a timeline could look like on the prototype.



7) First version of the timeline.

Appendix E - Logo



1) Multiple versions of the logo including the realisation process, favicon and text and non-text final versions.

Appendix F - Sprint Reports

Week 1 #09.02

Initial kick-off meeting, domain was discussed, establishing communication(Mattermost, GitHub) and meeting frequency, introduction to SAGE. Notes* on preliminary requirements:

Graphs:

- Show steps attacker took
- Must be able to show overlapping actions somehow
- Which services?
- Nodes with info like ip address

Recommendations:

- Filter out graphs in order to show what is most important (which may vary between organisations)
- How to visualise: e.g. drop down list

Queries:

- Filter by things like hostname or IP address
- How to search
- One type of log as input is good for now

Week 2

Based on the information from the SAGE paper and the previous meeting, further functionality notes were made to be discussed with the client and a few UI/UX mockups were created. The client expressed that they were not interested in mockups as they wanted something tangible that worked with data, otherwise they couldn't provide feedback.

Development of an MVP began based on the preliminary requirements.

Week 3 - Sprint 1 #29.02

The MVP was discussed with the client, the client was now more willing to provide feedback. The following requirements were created, marked off are the requirements that were already completed with the first MVP.

We also made changes to the way we implement the requirements: Meeting notes > Requirements > GitHub issues, this way it was clear what needed to be implemented, who was implementing it and when it was done.

Another improvement was getting more feedback by utilising Mattermost more frequently.

Requirements v.1

Must Have Dashboard

- The dashboard must accept network logs as input, run sage, and display the results.
- The dashboard must have a recommendation system that prioritises graphs based on what is the most critical for the security analyst.
- The dashboard must have an overview of the data, where it can help the analyst on what to look for.
- The dashboard must display information about node - attack stage, service, numeric identifier 'id' for context
- The dashboard must visualise severity with different shades of red.
- The dashboard must display information about the root (objective) node by default
- The dashboard must display the steps of a specific attacker and which victim hosts it targeted
- The dashboard must display the steps of a specific attacker and which victim hosts it targeted

Attack Graphs

- The graphs must show each step the attacker took.
- The graphs must either show attacks on all services (or/and?) attacks on each service separately.
- The graphs must display nodes information on hover - action description, attackers.
- The graphs must visualise repetitions on the attacker(s) step(s) (e.g loops using multiple edges).
- The graphs must visualise paths clearly.
 - The system must highlight the steps and the path the attacker followed when the user clicks on a certain attacker.
- The graphs must follow a certain structure e.g. tree structure with root nodes at same level on top

Timeline

- The timeline must have swimlane labels of the macro attack stage.
- The timeline must be able to show the timing of steps.
- The timeline must have a clear way of visualising overlapping steps (grouping by type when overlap?).
- The timeline must show how the attacker is doing the same action multiple times.
- The timeline must display multiple swimlanes for all the attackers the victim was targeted by.

Recommendation system

- The recommendation system must normalise on graph size when sorting by 'total severity' by scaling between 0 and 1

Should Have

- The timeline should have a filter based on time intervals.

- The dashboard should have an attack graph filter with a dropdown of possible filters existing in data and an option for the analyst to type filters themselves.

Could Have

- Be able to change the format of the graph (top-down, left-right)
- Being able to import the data from and to the app.
- The dashboard could have a filter based on attacker perspective
- The dashboard could have a filter based on victim perspective

Will Not Have

- Allowing the user to input logs not in the suricata format.

Week 4 - Sprint 2 #07.03

We were asked to think of an Overview that would display relevant information for the whole dataset on entry point. This request for yet another 'view' of the GUI gave us the idea to structure the requirement in such a way that we have separate MoSCoW requirements for each 'view'. We also migrated to Bootstrap, as per our clients' request to make the UI more modern and also provide consistency between browsers.

Requirements_{v.2}

Dashboard View

Must Have

- The dashboard must display the Overview as a button option in the sidebar.

Should Have

- The dashboard should have a search history in the filter dropdown and a delete history function.
- The dashboard should have a navigation bar to navigate through the different views.

Attack Graph View

Must Have

- The graph view must display information about the root (objective) node by default.
- The graph view must highlight the steps and the path of an attacker, including repetitions, and display targeted victim hosts, when a particular attacker is clicked from the details tab of the current attack graph.

Could Have

- The graph view could have an option to select the service(s) to analyse.
- The graph view could compare attack paths and highlight differences.
- The graph view could automatically detect duplicate attack paths.
- The graph view could display multiple attack graphs.

Timeline View

Must Have

- The timeline must show using sub-swimlanes overlapping attacks when grouped.

Should Have

- The timeline should have a filter based on time intervals.

Could Have

- The timeline could have a victim perspective which displays multiple swimlanes for all the attackers the victim was targeted by.

Recommendation System

Must Have

- The recommendation system must allow the analyst to choose between *victim IP*, *total severity*, *average severity*, *graph centrality* and *weighted centrality* to prioritise attack graphs.

Overview

Must Have

- The overview must be generated by (waiting on answer about the scope).
- The overview must take into account parameters passed by the search bar
- The overview must generate a pie chart of the attacked services.
- The overview must expand each slice of the pie chart into a histogram of the attack methods on the clicked service.
- The overview must link the attack method with the attacked service and generate a query that is inputted in the search bar and redirects the user to attack graphs that match the criteria, when clicking on the specific bar in the chart.

Week 5 - Sprint 3 #14.03

Discussed overview prototypes. The client had many different ideas on how the Overview could look like and they were all different from our prototype. In the end we agreed on using combinations of bar charts to display 2 out of the 3 dimensions (IP, Service, Attacker), with the slices being clickable and narrowing down the attack graph list.

We had a clear idea for the Overview so we could update the requirements with more detailed ones. Highlighted in green are the new requirements emerging from the meeting with our client.

Requirements^{v.3}

Dashboard View

Must Have

- The dashboard must open the Overview for the latest experiment by default.

- The dashboard must display the Overview as a button option for every experiment in the sidebar.

Could Have

- The dashboard could be able to change the scale of the graph, info and timeline windows.

Will Not Have

- Comparing between experiments.

Attack Graph View

Must Have

- The graphs must visualise severity with different shades of red filling the node and with a uniform shape (low - circle, medium - square, high - hexagon).
- Each graph node is highlighted on hover with a different colour for each severity.
- The graph view must highlight the steps and the path of an attacker, reducing the opacity of the rest of the graph, and display targeted victim hosts, when a particular attacker is clicked from the details tab of the current attack graph.

Timeline View

Must Have

- The timeline must show the 1st attacker's timeline by default, if there are multiple attackers.
- The timeline must have horizontal lines separating swimlanes.

Should Have

- The timeline should have a filter based on time intervals.

Could Have

- The timeline could have a victim perspective which displays multiple swimlanes for all the attackers the victim was targeted by.

Recommendation System

- The recommendation system must allow the analyst to choose between *victim IP*, *average severity*, *maximal centrality*, *weighted centrality* and *unique nodes* to prioritise attack graphs.

Overview

Must Have

- The overview must be generated when clicking on the overview button of an experiment, using the experiment's data.
- The overview must generate a combined bar chart that shows the service attacked (x-axis) and the type of attack (y-axis).

- The overview chart must be normalised so that all bar slices are clearly distinguishable.
- The overview chart must create search parameters that automatically search attack graphs in the sidebar, when a certain bar slice is clicked.
- The overview must allow for filtering of the data that are shown in the bar chart.

Week 6 - Sprint 4 #21.03

From this Sprint onwards, optimisation and bug fixing was prioritised, and new features were kept to a minimum. This sprint we also conducted User Acceptance Testing with both internal and external users, which generated a lot of feedback, mostly finding UI-related bugs and inconsistencies which were scheduled to be fixed. No major new features were requested by the users, although we could have probably asked them about this more directly.

Week 7 - Sprint 5 #28.03

In this sprint we focused on fixing all the bugs from the User Acceptance Testing, and implementing the quality-of-life features discussed during the user testing. These features were too small to be included in the requirements list. We did discuss two final *could* have requirements, and how they should be implemented. We agreed not to be implementing major features from this point on, and to only fix bugs.