Sharing and Automation for
Privacy Preserving Attack Neutralization

(H2020 833418)

D4.9 Demonstrator for tracking provenance in visual analyses, final version (M30)

Published by the SAPPAN Consortium

Dissemination Level: Public

H2020-SU-ICT-2018-2020 – Cybersecurity

Document control page

Document file: Deliverable 4.9
Document version: 1
Document owner: Robert Rapp (USTUTT)

Work package: WP4
Task: T4.5 tracking of analytical provenance
Deliverable type: Demonstrator
Delivery month: M30
Document status: ☒ approved by the document owner for internal review
☒ approved for submission to the EC

Document History:

Version	Author(s)	Date	Summary of changes made
0.1	Robert Rapp	2021-10-26	Preliminary document sent out for internal review
0.2	Robert Rapp	2021-10-27	Updates based on internal feedback
1	Robert Rapp	2021-10-29	Finalise document for submission

Internal review history:

Reviewed by	Date	Summary of comments
Gabriela Aumayr	27-10-2021	grammar
Franziska Becker	27-10-2021	Technical, grammar and spelling

Executive Summary

This final report updates the initial report D4.8 and was created in the scope of task T4.5 “Tracking of analytical provenance” in the SAPPAN dashboard. Analytical provenance is used within the SAPPAN dashboard, a visual analytics web interface for security operation centre (SOC) analysts to handle incidents in the SAPPAN project. This deliverable illustrates the use of analytical provenance in the SAPPAN context. It shows how captured interactions during an analysis lead to different representations, which the analysts can use to comprehend recorded analysis sessions for further reuse. For this purpose, we explain the foundations, the recording and outcoming graph visualisation, and the live session player in detail. A summary and the future work section show further improvement with incident response playbooks out of the scope of this deliverable and summarise the results.

This deliverable focuses first the motivation for provenance and frame it in the SAPPAN Context. After a look into related work, the requirements and user of analytical provenance in the dashboard explained. This is followed by the implementations details contain, the system architecture and a provenance information flow. The implementation for recording analysis session and view analysis sessions are described in detail. As last the evaluation of the approach is explained and the deliverable is summarised.

Table of Contents

1 Introduction

In general, work documentation is time-consuming but aims to persist information that stakeholders further reuse to comprehend decisions or analyse the provenance of actions. One part of the SAPPAN project is a dashboard for security operation centres to analyse incidents visually. These analyses are the basis for analytical provenance, and the aim is to capture the analysis process. Understanding the reasons for user insights and their manner is most relevant and challenging for analytical provenance. In this deliverable, we use analytical provenance in Ragan's understanding and refer to the history of changes to data, system state, user interaction, and human understanding during analysis [5]. The information for analytical provenance is focussed on and limited to human operators in a visual analytics system like the SAPPAN dashboard. To expand the SOC analysts' opportunities within the SAPPAN dashboard, we created a tool to record the analysis and use the recorded data to visualise the sequence of user activities. This approach allows analysis sessions to be interpreted and understood by both humans and machines, making them comparable and suitable for various applications. SOCs can use the results of the human-based analysis to improve processes where necessary.

2 SAPPAN Context

As described in the Grant Agreement, this deliverable expands the SAPPAN dashboard's functionalities with provenance information. The deliverable itself is part of work package 4, which is about managing and automating threat intelligence. This deliverable is mainly related to the SAPPAN dashboard in work package 6, which serves as a testbed for implementing tracking of analytical provenance. The basic functionality of the dashboard and visualisations are a prerequisite for this task to capture analytical provenance information. SOC analysts use such dashboards in their incident response pipeline to analyse security alerts generated by the detection pipeline and define response actions for further handling. The work performed for *D5.8 Sharing response handling information* is about sharing playbooks that describe response actions. This deliverable describes a similar graphical representation for analysis sessions that represents actions that are part of response playbooks. The SAPPAN dashboard cannot depict the complete analysis process, as a playbook can describe it, but the representation loosely couples both approaches for comparison.

3 Related work

In the context of cyber security, analysts use a diverse range of tools to extract meaning and insight from data e. g., in order to assess the validity of an alert. Visual analytics aims to support users by combining human and machine capabilities. The capturing of interactive data exploration and the human reasoning process can be a valuable application for users, called analytical provenance.

Ragan [5] and colleagues defined an organisation framework by studying different types of analytical provenance and their purposes:

Recall	Maintaining and recovering memory and awareness of the current and previous state of analysis.
Replication	Reproducing the steps or workflow of a previous analysis.
Action Recovery	Maintaining the action history that allows undo/redo operations and branching actions during analysis.
Collaborative Communication	Communicating and sharing data, information, and ideas with others who are conducting the same analysis.
Presentation	Communicating the insights or progression of the analysis with those who are not directly involved with the analysis themselves, such as the general public, upper levels of management, or analysts focusing on other areas.
Meta-Analysis	Reviewing the analytic processes themselves in order to understand and improve aspects of the analysis (such as process efficiency, training efficiency, or analytic strategies)

Table 1: Purposes to accomplish with provenance by Ragan [5]

Our approach comes with an implementation of a state-based tracking to recall the arrangement of view and the viewed data at the time an analysts views again a session. The processing of the collected provenance information allows the workflow of an analyst to be replicated. Analysts are connected via the SAPPAN sharing network to exchange playbooks. If the recording subsequently is part of the shared data, analysts can ex-change via the comments their insights, and the two representations in this prototype can be used not only for internal collaboration, like in reviewing analysis and optimising playbooks.

As research focussed on the capturing of user interactions, little work was investigated in sensemaking as it becomes the most challenging part. Xu et al. [6] show the challenges of sensemaking for analytical provenance by discussing uncertainty, semantic hierarchies, manual and automatic capture mechanisms and approaches for the visualisation.

An approach focusing on sensemaking is a browser-based online sensemaking tool developed by Nguyen et al. [3]. Their approach captures context information of GET requests and applies action types to URL parameters within a browser extension to generate insights. They resolved their captured data in a timeline visualisation and a browser view to enable analysts to comprehend browser sessions. Xu et al. [6] show the challenges of sensemaking for analytical provenance by discussing uncertainty, semantic hierarchies, manual and automatic capture mechanisms and approaches for the visualisation.

Although the perspectives on analytical provenance vary slightly in research, it often refers to capturing interaction within a visual analytics system. It tries to extract the human thinking process during analysis. However, other approaches like Gorth and Streefkerk [1] use systematic recordings of screenshots during an analysis, enriched with higher-level semantic information from users. That can be used for visually recalling an analysis and matches the idea of the *Live Session Player* in our approach. North et al. [4] came up with a sequence, namely perceiving, capturing, encoding, recovering, and reusing for analytical provenance functionality. Our approach shows the whole process from perceiving to reusing it for a provenance graph in the same understanding.

In real-world applications, data undergoes several transformation pipelines, and the analysis workflows involve many steps to formulate hypotheses and generate insight.

Heer and Shneiderman [2] emphasise two forms of data to capture: designed systematic capture (e.g., user interactions, screenshots of visualisations) and explicit user annotations, which we adapted to our approach.

4 Analytical provenance in the SAPPAN dashboard

We will deploy the SAPPAN dashboard as a visual analytics interface for human analysts to view endpoint and network data resolved in different visualisations. To reuse the analyst's actions during analysis, we extended the dashboard with functionalities that allow capturing these actions, persisting, and making them accessible via a graphical representation and a live session viewer.
The biggest challenge of analytical provenance is to track the user's insights that emerge during the analysis. The human thought process is one of the essential factors in conducting data analysis, as it involves visually interpreting mapped data and drawing conclusions based on the interpretation. A sequential mapping of an analyst's interactions provides insight into the transitions, i.e., which steps have led to a further step. However, the insights arise in the analyst's mind, i.e., outside the actual application, and are therefore not detectable via the web interfaces in which the analysis is carried out. Our requirement analysis in D2.3 figured out that analysts use several mostly browser-based tools in their investigations. While the SAPPAN dashboard aims to provide a collection of visualisations, the dashboard cannot replace the range of external tools used within a SOC. These tools are already established depending on the analyst’s workflow. The collection of our provenance data is limited to the analyst's interactions with the SAPPAN dashboard, and conclusions about specific decisions depend on different visualisations. Therefore, the implementation chapter is limited to analytical provenance within the dashboard, beginning with a basic workflow an analyst would run in the dashboard.

4.1 Analysts’ Workflow

The analysis of an alert we capture is only carried out under the condition that an automated system cannot resolve the decision about whether it is malicious or not. The analyst follows an abstract workflow for the analysis. At the start, the detection pipeline of a SOC sends an alert via e-mail to the analyst with related information like the suspicious process that was detected during automated process analysis by an endpoint sensor. Analysts initially can start by opening the view they need for their analysis tasks. Analysts can enter the related information like the global process identifier to request the endpoint sensor data or NetFlow traffic records. An analysis of endpoint data usually begins with the analysis of the process tree, where process hierarchies and process activities can be analysed. The analysis depends on whether the analyst has found something in the respective data or is looking for other indications from external sources. Analysts then finish their analysis by deciding whether the alert is an actual incident that will result in response actions or a false positive alert.

Analytical provenance can serve several purposes in this workflow:

- The layout collection can provide information about what different visualisations are used by analysts

- Furthermore, an analyst can trace the history of visualisation interactions and the manipulation of corresponding data
- A graphical representation of the provenance information allows sharing and discussing it with colleagues or comparing it with incident response playbooks that describe a sequence of actions for incident handling.

4.2 Requirements

To achieve these purposes within the SAPPAN dashboard, we had to consider the following requirements:

- Multi-visualisation support: The SAPPAN dashboard offers a flexible card-based layout that can be equipped and extended by the analyst with various visualisations for incident analysis in network and endpoint data.
- Analysts are no data visualisation practitioners: Complex visualisation techniques are not practical because they require additional effort and training to use or communicate it to others.
- Tracking of external tools: Analysts may use other tools in their workflow that are not part of the dashboard.
- Playbooks as the description of tasks: Playbooks can be viewed in the dashboard and are the most related to the workflow that the representation should be similar to.

The provenance information must contain data of multiple visualisations and their layout arrangement, which analysts have chosen for the analysis. Tracking of external tools requires grant access to the whole machine of an analyst, which is not a realistic use case in the scope of this project. The approach from Nguyen SensePath [3] tracks the GET requests and URL parameters of analysts' browser activity to get insights into their activity. Nguyen faced difficulties telling whether the participant was reading, thinking, or simply away from the analysis based on the information available from the browser. We figured out that we cannot content related to external tools such as used in Cyberchef [1]recipes. Because that would add misleading information into the provenance session, we have not followed that approach and focused on deep-linking. Deep-linking links directly to specific content, in contrast, to visit the page and navigate to the searched content.

[1] CyberChef: http://icyberchef.com/

5 Prototype

This chapter describes the technical implementation of our analytical provenance prototype. First, the architecture of the SAPPAN dashboard is explained where the provenance tool is a part. This is followed by an overview of the analytical provenance information flow between the frontend and backend. The chapter shows then the implementation details of 1) the data persistence and pre-processing of provenance data in the backend and 2) how we capture and represent provenance data in the frontend.

5.1 Architecture

The system architecture for this deliverable build on the architecture of the SAPPAN dashboard.

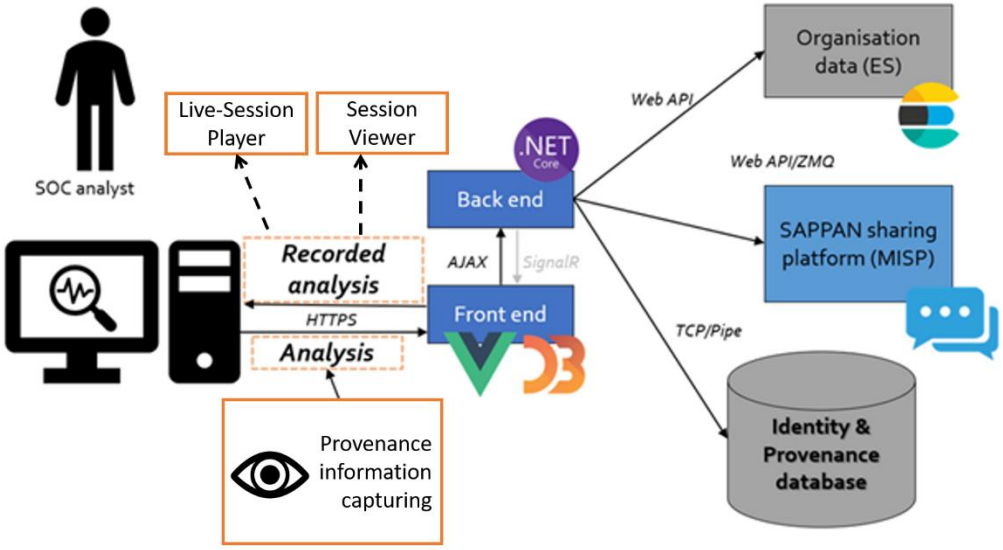

Figure 1: System architecture for analytical provenance

The architecture in Figure 1 shows the SAPPAN dashboard architecture with its connections to external sources. Technically, the provenance tracking implementation in the SAPPAN dashboard is split between the ASP.NET Core backend and the Vue.js front end. The front end is used to capture interactions from an SOC analyst and access recorded analysis sessions via a graphical session viewer. The back end takes the captured information and interacts with the database and external data sources. The front end and back end are loosely coupled through a REST API. The provenance database is used to store provenance data and identity information for analysts.

5.2 Provenance Information Flow

Figure 2 provides an abstract information flow of provenance data.

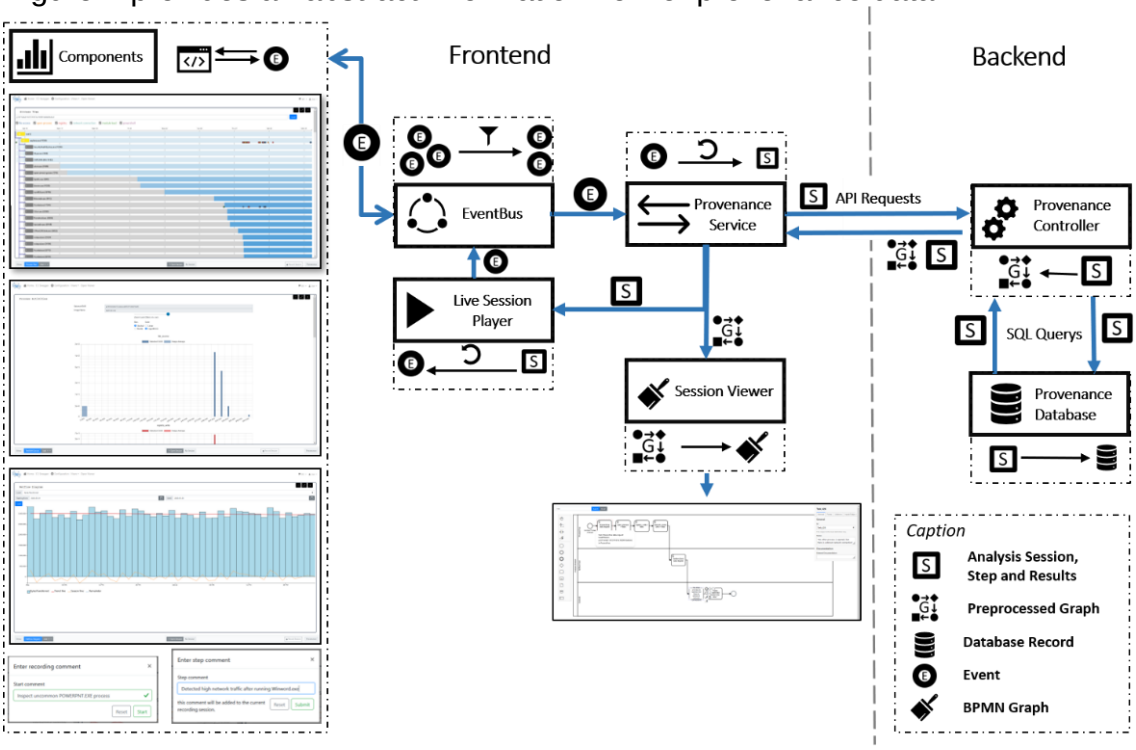

Figure 2: Provenance information flow

Figure 2 shows several dashboard *Components* that can be graphical views or the comment function. These components emit self-describing events to react to selected user interactions or reproduce interactions based on events. The *EventBus* receives all the emitted events and has defined reactions to these. It sends valid provenance events to the *Provenance Service*. The *Provenance Service* adds additional information, like the type of interaction, session-id or timestamp, before sending them via an API request to the backend. The P*rovenance Controller* receives analysis steps and saves them into the *Provenance Database* (SQL) or responds to the frontend requests. If the *Provenance Service* requests a session graph, the *Provenance Controller* fetches the requested session from the *Provenance Database* and converts it into a prepared format to draw the graph. Otherwise, an analysis session is returned. The *Provenance Service* distributes the response into the *Session Viewer* for a graphical representation or the *Live Session Player* for a step-by-step walkthrough. For each step, the *Live Session Player* triggers a frontend function by the *EventBus* that causes an execution of interaction in a component.

5.3 Implementation Details

This section shows the implementation details of the analytical provenance as part of the SAPPAN dashboard. Provenance Information Flow5.2 It starts with an introduction to the provenance related user interface changes to record sessions in the frontend, followed by an overview of the event processing used for recording. The next part explains the ASP.Net Core backend functions to persist, preprocess and return analysis sessions to the frontend. The last part of this section contains two different representations in the frontend for an analysis session.

5.3.1 SAPPAN Dashboard User Interface

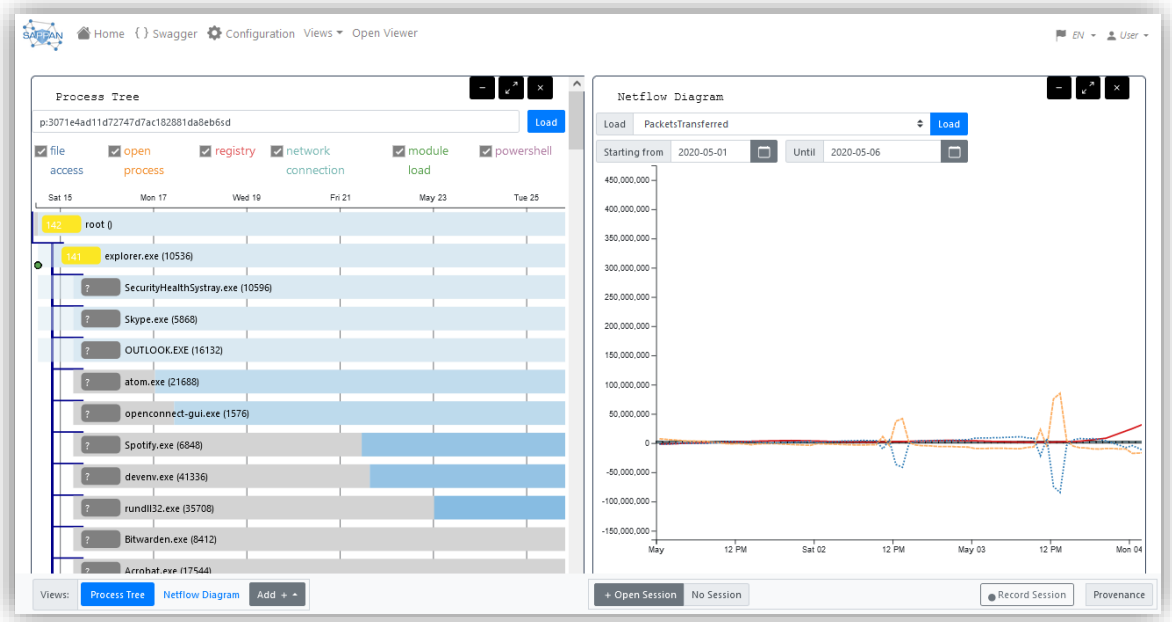

Figure 3: User interface during an analysis.

This user interface is a card-based grid written in with Vue.js, where a user can open, close, resize and arrange analysis views. These views visualise data from different sources like IP-NetFlow data or process data used in a SOC to analyse incidents. In Figure 3, the provenance bar can be viewed in the bottom right. It collections all interaction possibilities regarding analytical provenance to avoid confusion because they change depending on whether an analysis is recorded or played. To start tracking the analyst's activity, the analyst must be logged in and start the recording manually. This has the advantage that analysts can decide for themselves which incidents they consider relevant, which leads to a better quality of the recorded analysis sessions. Initially, an analyst can decide whether to retrieve an already recorded session in the *Live Session Player* or record a session. The *Live Session Player* takes place in the shown user interface, replaces the content with the layout of the recorded analysis session, and enables navigation possibilities in the provenance bar. By using the menu button "Open Viewer", the *Session Viewer* can be opened.

5.3.2 User interaction tracking for recording a session

The user has to push the “Record Session” button in Figure 4 to open a dialogue form that requests an initial comment for an analysis session. This comment describes the start of an investigation and will be saved in the provenance database. If the users press on start, the current layout is captured, and the *EventBus* begins to listen to changes.

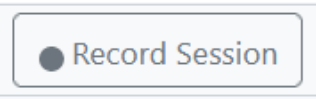

Figure 4: Record session button

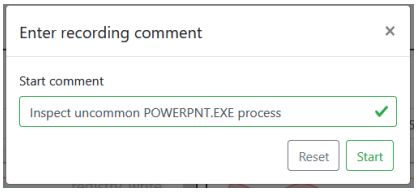

Figure 5: Analysts enters a initial comment to start recording

EventBus

With the *EventBus,* we can react to various actions, such as user interactions, through a publish-subscribe mechanism. It enables the communication between different dashboard components. The *EventBus* can hook into the visualisation rendering process to capture the implemented interaction functions of such advanced visualisations. Therefore, the *EventBus* can publish changes triggered by the analyst from any class, and *EventBus* recipients can subscribe in order to react to changes. These recipients filter relevant provenance information before adding it to an analysis. Because the *EventBus* is used in all dashboard components, it works like a hub for interactions triggered by analysts. Figure 6 shows the relation between the *EventBus* and the visualisation and layout components. A full list of recorded events is shown in Table 2.

Name	Description
Layout Tracking	
MoveEnd	Fires when the move is complete
ResizeEnd	Fires once resizing is complete
AddCard	Fires when an analyst adds a visual component
RemoveCard	Fires when an analyst closes a visual component
MinimizeCard	Fires when an analyst minimises a visual component
MaximizeCard	Fires when an analyst maximises a visual component
GraphViewer	Fires when the analyst opens the playbook graph viewer
ClearLayout	Fires when the analyst closes all views
Layout	Fires when the analyst begins their analysis and no layout was saved so far
Data Tracking	
LoadData	Fires initially when analysts press on load
LoadOpenTree	Fires when a node in the process tree is expanded
FilterEventType	Fires when an analyst applies an event type filter in the process tree visualisation
SelectProcess	Fires when an analyst clicks on a specific process
FetchChild	Fires when an analyst requests a child process that is not part of the process handle
EnrichDetails	Fires when the SHA1 value of a process is sent to VirusTotal API to enrich process information
ShowPlaybook	Fires when an analyst fetches a playbook from the backend
LoadHostData	Fires when analysts fetch specific host information regarding an incident
User Insights	
AddComment	Fires when an analyst enters a comment during the analysis

Table 2: All events used to track interaction during analysis.

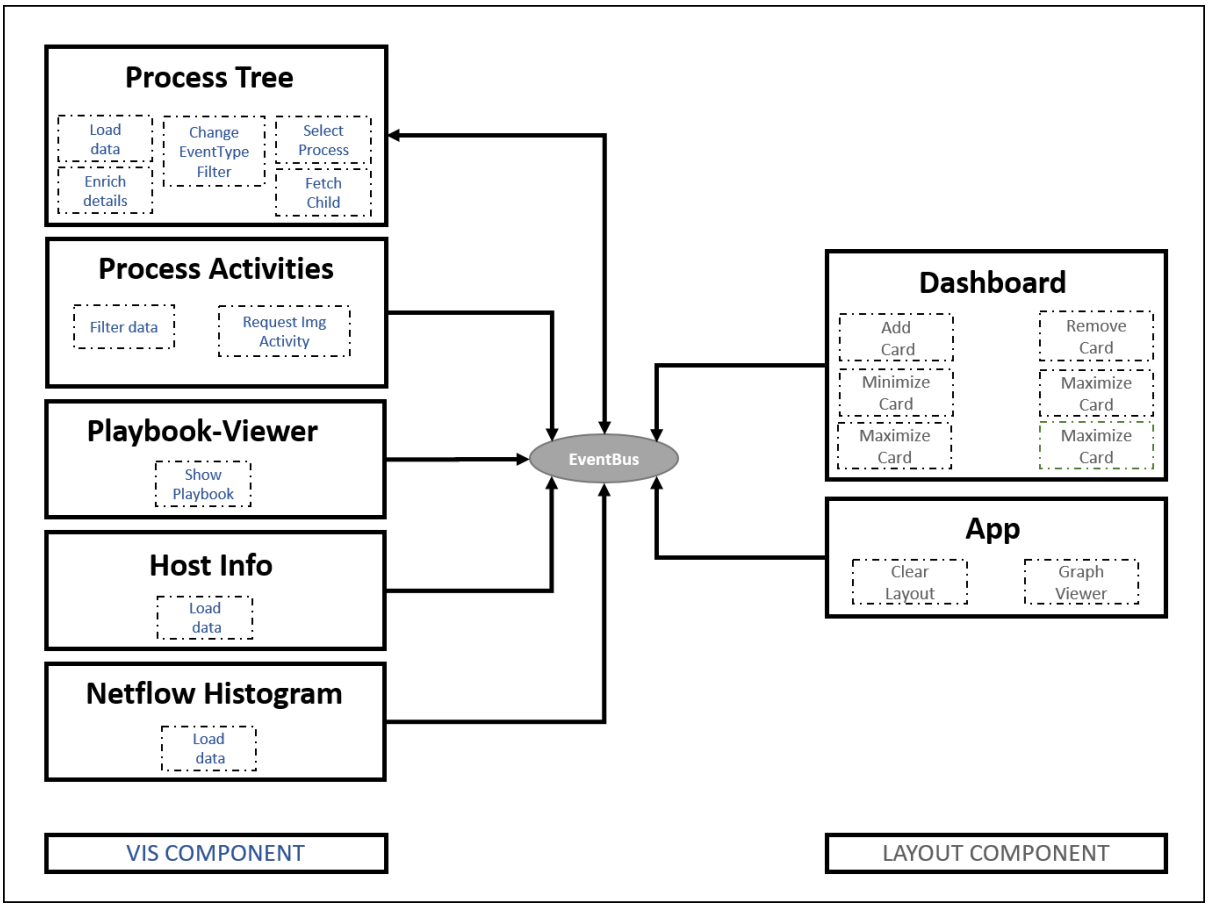

Figure 6: The EventBus in the SAPPAN dashboard behaves like a hub for user interactions.

In Figure 8, the typical use of the *EventBus* as a provenance information publisher is shown. An event needs to be self-descriptive so that the *Live Session Player* can re-build the situation. Relevant information like the component, action, description and request, and timestamp is filled with component and situation-related information and included in the event's payload. The event can then be subscribed up within the complete frontend application. Figure 9 shows how this event is received elsewhere to perform an action such as the function call recordData().

In this manner, each component has a set of defined interactions that are emitted in self-descriptive events and handled afterwards. The process tree view holds one implementation of external tool tracking. The traces of external tools are challenging to rely on or mostly hidden. We addressed this issue by enriching the shown process

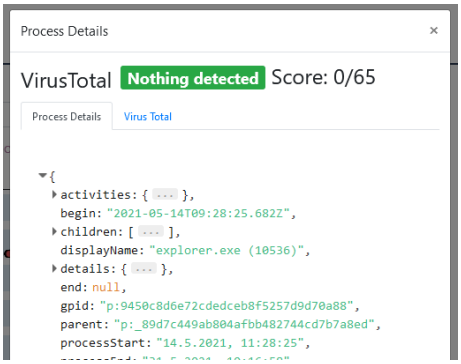

Figure 7: Analysts view the assessments from 65 security providers

```
EventBus.$on('LoadData', (payload) => {
    let event: any = payload;
    this.$data.data.push(event);
    localStorage.setItem("data", JSON.stringify(this.$data.data));
    this.recordData();
})
```

Figure 9: An event is received in component

```
EventBus.$emit("LoadData", {
    component: this.$options.name,
    timestamp: new Date(),
    action: "Get netflows",
    description: "Get netflows from "+ this.fromDate +" to " +this.toDate,
    request: this.api + "netflows/decomposition?from=" + this.fromDate
        + "&to=" + this.toDate
        + "&bucketSize=" + this.timeSpanToString(bucketSize)
        + "&periodLength=" + periodLength
```

Figure 8: An event is emitted in a component

information to optimise the analyst's workflow and enable tracking of the content that could lead an analyst to an insight. An API call fetches additional process information from an external tool called VirusTotal[2] that is frequently used in malware analysis. The use of the process tree view is a common starting point of an analysis, where an analyst can select a specific process to view the process details shown in Figure 7. The details are then enriched by an assessment of various security providers, which states how many of these providers consider the SHA1 value of the process to be malicious. Compared with Nguyen et al. [3] SensePath approach, this leads us to capture information about a specific process the analysts were interested in and know the response of VirusTotal. As a benefit, this avoids one additional step to an external tool in the analyst's workflow. This way can be adapted to other API's which are frequently used in SOCs.

If SOC analysts want to annotate something during analysis, they can use the comment function to persist it as a step comment. To stop an investigation, the button "Stop" in Figure 11 can be pressed. The click triggers the dialogue in Figure 10 to create an analysis result and save it in the database. The user interface stops recording the user activities and finalises an analysis session by adding the related result.

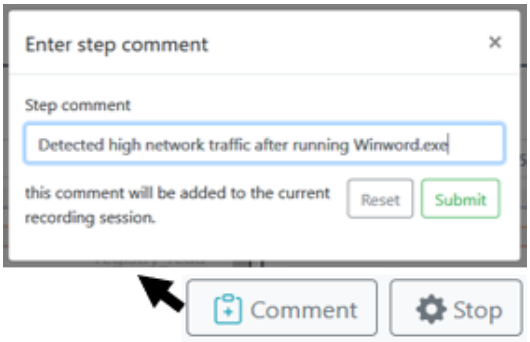

Figure 11: Analyst can add comments or stop session recording

Figure 10: If analysts found something the alert is malicious

5.3.3 ASP.Net Core backend

The backend receives all provenance events converted to analysis steps. The backend is implemented in ASP.NET Core and has a REST Interface to interact with the client and a native connection to the provenance database. The backend has access to response and recovery playbooks from WP4 via SAPPAN sharing system described in D5.8. These playbooks show a way to handle an incident and send it to the client to compare it with analysis sessions.

Provenance database

The back end natively interacts with the SQL *Provenance Database*, which stores the provenance data we record. Depending on the data model of recorded sessions, a relational database maps best to command pattern, the recorded data design of our provenance tracking solution. Command pattern is on a collection of objects representing the actions being performed. Figure 12 shows the entity relationship model of the provenance database. The model shows four tables named *Users*, *Analysis Sessions*

[2] https://www.virustotal.com/gui/

Analysis Steps and *Analysis Results*.

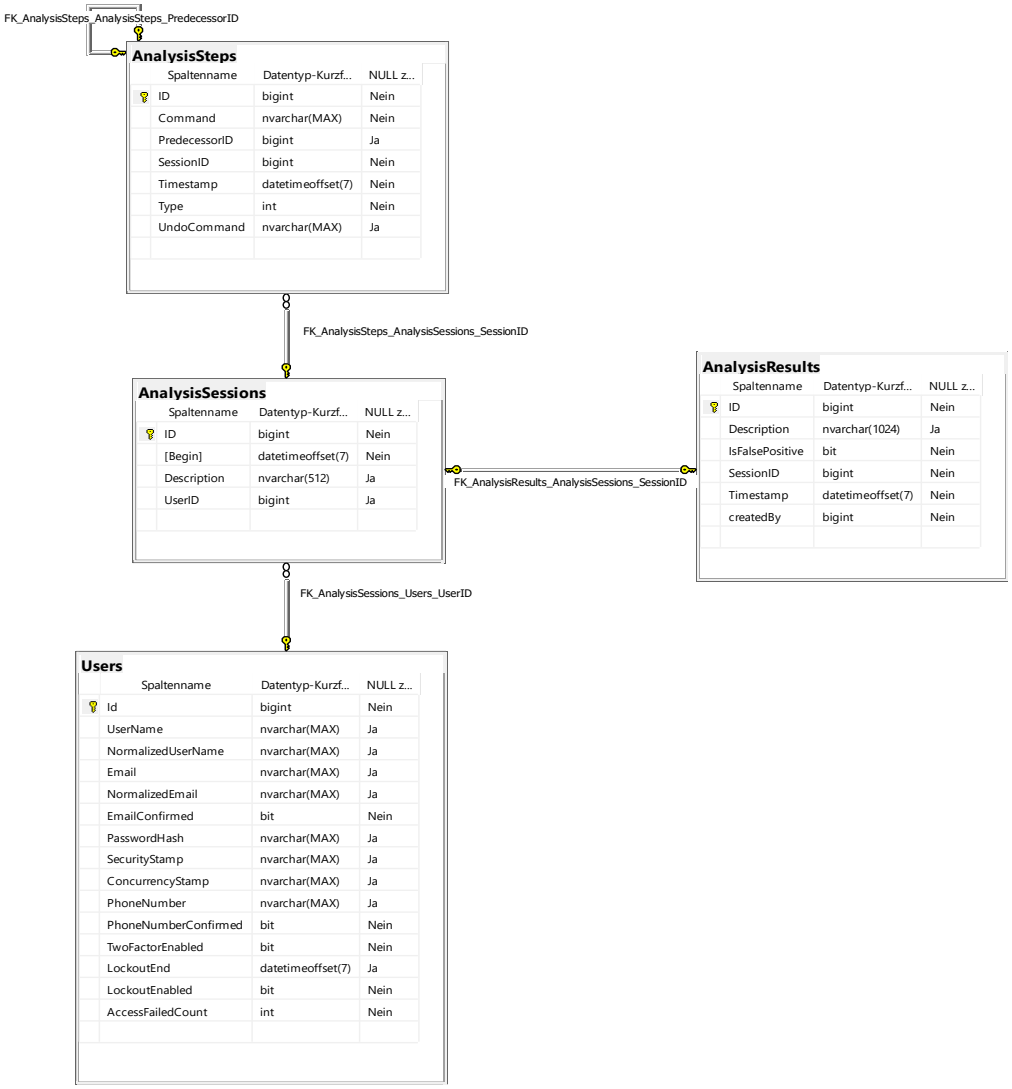

Figure 12: Entity Relationship Model of the *Provenance Database*.

The *Users* table is a representation of the ASP.NET Core Identity. It adds to the SAP-PAN dashboard user login functionalities. It manages common data for login and user management like user name, password, profile data and more. On the top of these columns, registration and the login form from Microsoft was implemented. Each SOC analyst has a user account to identify to whom actions belong to keep track of the user activities. For this reason, a user account is necessary for capturing provenance.
The tables *AnalysisSessions*, *AnalysisSteps* and *AnalysisResults* are essential for the persistence of the user interactions in a session. Each session, step or result has a timestamp of its creation to keep track of the sequence of actions and persist the analysis duration.
When an analysis session is created, it is always assigned to a user who has started recording the analysis session in the frontend. An analysis session consists of one to several analysis steps. Each step contains the action and information to which session it belongs to. Due to this structure, the composition of the analysis session is flexible. It can record sessions of any length and continue the recording of further steps at a later time. When an analysis is over, an analysis result is stored, which extends the session with the decision of the SOC analyst whether the incident report is a false positive or a false negative.

In the entity relationship model of the provenance database, the implementation of a specific command pattern with do and an undo stack is visible. These commands are generated by the views in the SAPPAN dashboard like the visualisations, the layout and user dialogues and are wrapped as a command in an analysis step. For example, if the user loads content for a visualisation the requests is saved as object in a step. The type of application object gives information about what will be saved in the command and is saved in the **type** attribute.
The analysis steps can be one of these types: *layout*, *layout manipulation*, *data*, *data manipulation* and *comments*. For each analysis, an initial layout is recorded and changes to that layout are further saved as analysis steps. For example, adding a visualisation component triggers the creation of an layout manipulation step, because it changes the initial layout. Similarly, a data request is saved for each component if new data is loaded in the visualisation. Data manipulations reflect the use of filters and changes in the viewed data so that the changes an analyst caused on the data can be rebuilt on top of the user's interaction sequence.

Commands can be in different formats like script languages, events or text notes. The type provides information on whether, for example, a user dialogue, a JavaScript command or the dashboard arrangement needs to change while executing the undo stack. If a command has a backward and forward implementation, undo and redo can be implemented by storing the chain of commands. These analysis steps are chained together with previous steps by the attribute *PredecessorID* to realise the undo stack.

Provenance controller

The *ProvenanceController* bundles all functions for recording and retrieving provenance information. The *ProvenanceController* communicates with the database to create, read and update sessions. This class also performs the pre-processing of the session graph. For this purpose, the contents of an analysis session are converted into a format suitable for the graphical representation, which means that the steps are sorted into specific views and labels, and task types are added directly.

5.3.4 Replay a recorded session

An analyst can open recorded analysis sessions in the *Live Session Player* or the *Session Viewer*. To get an analysis overview in the *Session Viewer,* the analyst can view sessions in graphical form. The *Live Session Player* allows the analyst to view different interaction steps of an analysis session within the dashboard.

To replay a session, the user has to push the “Open Session” button to open a dialogue which shows a list of sessions, displayed in Figure 15. This list of sessions is fetched from the backend via the *Provenance Service* and lists the already recorded sessions. The number of days shows the time passed since the session was created.

If the users press on open, the dashboard opens the *Live Session Player* in the bottom right in the provenance bar shown in Figure 13. Analysts can navigate within an open session by clicking the navigation elements in Figure 14.

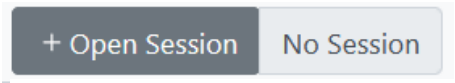

Figure 13: Open Session button and label of selected session

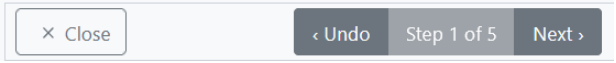

Figure 14: Analysts can navigate over a session via this navigation

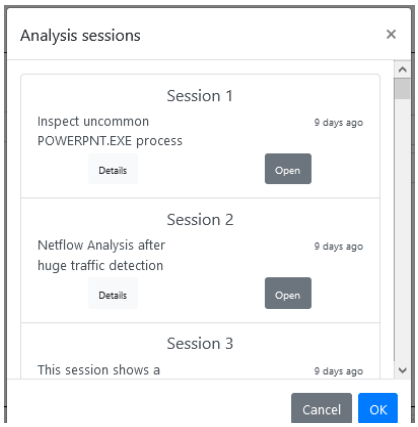

Figure 15: Analysis session list to select a recorded session

Live Session Player

The *Live Session Player* allows the analyst to view different interaction steps of an analysis session within the dashboard. The first *Layout* step rearranges the dashboard for viewing a recorded session. The user interface avoids that analysts can record a session during the replay of a session by hiding the “Record Session” button. The sequential structure of an analysis session then allows analysts to comprehend what an analyst has performed during the recording. The command type determines how an analysis step is mapped. For each of the following five command types, there is a specific way. The aim is to create a picture as detailed and comprehensible as possible, guiding the analyst through the recorded session.

Command Type	**Action in the session player**
Layout	Layout steps usually occur only once per analysis session and replace the current dashboard layout with the initial layout, i.e., which views are open and how they are arranged.
LayoutManipulation	Layout manipulations such as resizing, opening or closing views directly change the layout as the analyst did during the analysis.
Data	Data requests are distributed via the *EventBus* to the correct view, where an API call is made to retrieve the data and render the visualisation.
DataManipulation	Data manipulations are also distributed to the correct views via the *EventBus* and directly manipulate the visualisation or the underlying data there.
Comment	Comments are displayed in an overlay above the currently open session.

Table 3: Mapping of command types and actions.

If necessary additional notifications are sent to users in the upper right, to inform them that the execution expected some delays, what can be the case when data is requested from the backend. Asynchrony requests to the backend show always an loading spinner to inform the user that there will be content fetched. The *Live Session Player* additional triggers a notification, to avoid that analysts did not recognise the action that is usually triggered by pressing on a “Load” button.

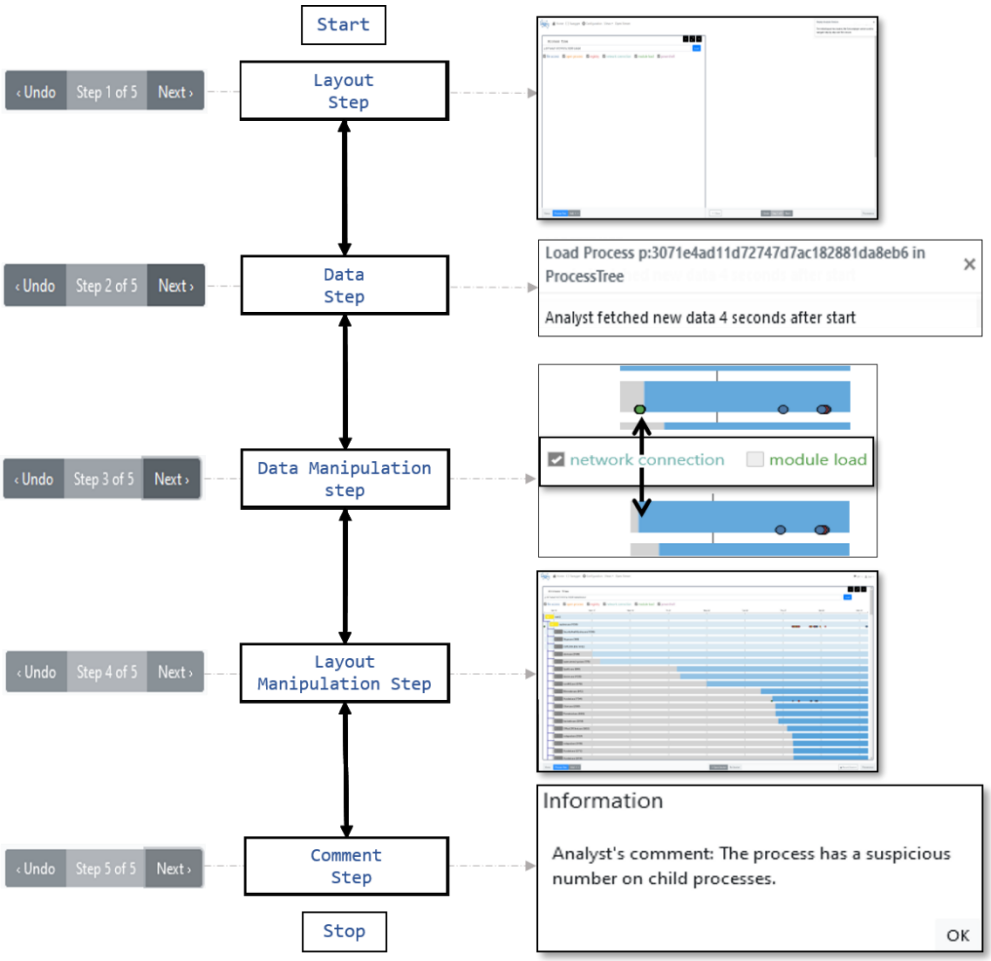

Figure 16: The execution of all command types in one session.

Figure 16 shows a simplified analysis session experienced by an analyst in the *Live Session Player*. With the navigation element shown between the steps, the analyst goes across the session. First, the session player opens the session’s views, followed by the related data request. If the session contains filter changes, the checkbox of module_load is unchecked, which leads to the visualisation being manipulated in the same manner as in the session recording. The layout manipulation step maximises the visualisation to the full-width, followed by the comment step shown in a simple read-only user modal.

5.3.5 Graphical Representation of Analysis Sessions

Analysts can use the *Session Viewer* to view the graphical representation of an analysis session. In comparison to the initial Deliverable D4.8, we changed the representation from the graph library G6 to bpmn.js[3]. In the SAPPAN dashboard, we use BPMN

[3] https://bpmn.io/toolkit/bpmn-js/

for incident response playbooks shared over the SAPPAN sharing platform and the analysis session. This brings the advantage that analysts need to familiarise themselves with only one representation.

BPMN is a graphical notation to describe business processes in a process model and was developed for analysts, technical developers and business managers. For this reason, we opted for BPMN as our graphical representation of analysis sessions, as they should be easily communicated to colleagues and other stakeholders without demanding advanced visualisation capabilities. Maintained and published by the Object Management Group (OMG) and ratified as ISO19510, BPMN in version 2.0.2 can represent complex process semantics that include human and machine-based actions in one diagram. It is based on a flowcharting technique like activity diagrams from Unified Modeling Language and defines four element categories *Flow objects*, *Connecting objects*, *Swim lanes* and *Artifacts* to model information.

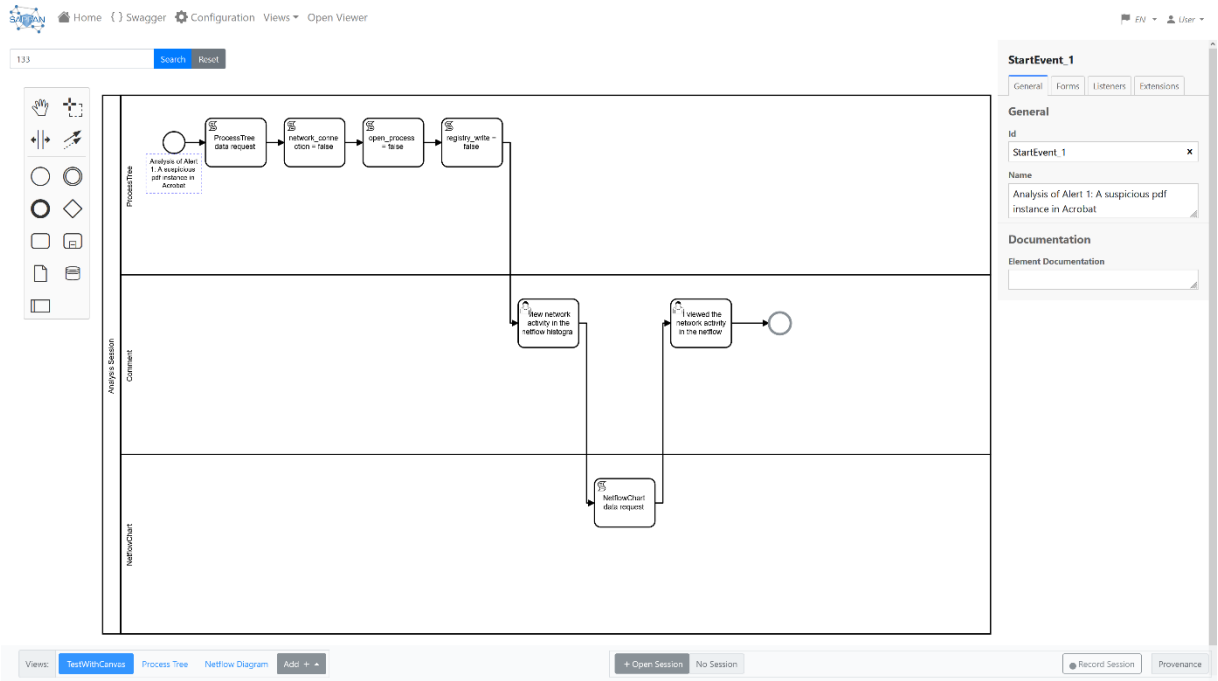

Figure 17: A simple analysis session graph.

Figure 17 shows an opened analysis session in BPMN. The input field in the upper left corner allows the analyst to search for a specific session or reset the view. An analyst can add more elements with the toolbox on the left or modify the current display. Both functions are not necessary for viewing an analysis session. In the upper right corner, a selected element can be viewed or edited.
The outer frame represents a BPMN pool and is labelled *Analysis Session*. Inside this frame are the three lanes *ProcessTree*, *NetflowChart*, and *Comment*. The lanes give the analyst a first overview of different views used in the recording. Each lane outlines associated steps to visually separate similar activities in different views. An analysis always starts in the first used view and is represented by a circle. The connecting arrows show the analysis flow that always connects two elements in the case of an analysis session. If an arrow is bent, an analyst has switched to another view or the comment function. The zoom and drag function of the viewer allows analysts to view even larger analysis sessions. The integration of tooltips makes it possible to retrieve details

on-demand that are not included in the BPMN to avoid visual cluttering. Besides tooltips displayed by hovering, analysts can interact with *Script Tasks* and *User Tasks* by double-clicking on them. *User Tasks* trigger a model for the analyst's comment, suitable for more comprehensive insights, and *Script Tasks* trigger a confirmation dialogue. Analysts can then decide whether they want to execute this step in the Live Session Player. The Live Session Player opens and jumps directly to the selected step if the user confirms the dialogue. This links the overview of an analysis session and the detailed action during a session in the dashboard. The last circle indicates the analysis result, which differs in colour and thickness compared to the start circle. If the circle is red, it indicates that the analysis ended with the assessment that it was an actual incident.

6 User Feedback

Analytical provenance will be more thoroughly evaluated as part of the SAPPAN dashboard evaluation in D6.1 which is part of the Demonstrator in WP6 and will address several SOC experts in a demo and survey. As the SAPPAN dashboard is still under development, the evaluation is not part of this deliverable. For an initial user feedback, we conducted a feedback session for this prototype. Our focus was on the usability of the recording function and the use of the graphical representations for comprehending interaction steps. In particular, we were interested if the participant could record a session and reconstruct the session by the use of the *Session Viewer* and the *Live Session Player*. We gathered user feedback in an online review session via video chat with one participant.

Setup
We prepared a computer as an experimental setup that runs a dashboard instance locally connected via the backend to the servers of the provenance database and external data sources. A video chat with WebEx is started from this computer, which the participant can join.

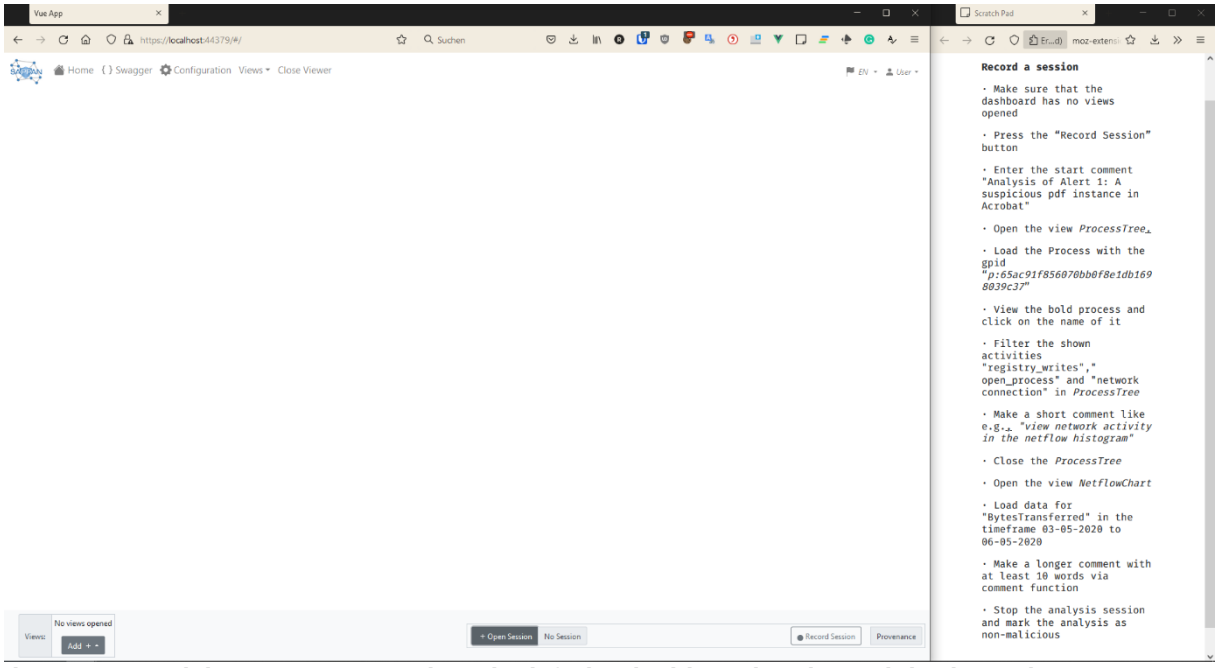

Figure 18: Participants can control on the left the dashboard and see right the tasks

The dashboard is opened in the browser next to the task description so that a participant can use it as guidance without changing windows, as shown in Figure 18. We designed the tasks so that no visualisation or security expertise was required. We initially control the experiment computer to introduce the tasks before we hand over the keyboard and mouse control so that the participant can familiarise themselves with visualisation views and the control.
When the participant has no more open questions, the setup is ready to start the experiment. This setup makes it possible to make this feedback session online without installing any software on the participant's machine. However, it requires a computer with a mouse and keyboard and an internet connection for the remote-controlled session.

Procedure
We split the session into two parts. The first part was to conduct feedback for the recording of provenance data and the second was to interpret and comprehend a session with the session graph and the session live viewer.

Part 1: Record a session

- Make sure that the dashboard has no views opened.
- Press the “Record Session” button.
- Enter the start comment "Analysis of Alert 1: A suspicious pdf instance in Acrobat".
- Open the view *ProcessTree.*
- Load the Process with the gpid “*p:65ac91f856070bb0f8e1db1698039c37*”.
- View the bold process and click on the name of it.
- Filter the shown activities "registry_writes"," open_process" and "network connection" in *ProcessTree.*
- Make a short comment like e.g., *"view network activity in the netflow histogram".*
- Close the *ProcessTree.*
- Open the view *NetflowChart.*
- Load data for "BytesTransferred" in the timeframe 03-05-2020 to 06-05-2020.
- Make a longer comment with at least 10 words via comment function.
- Stop the analysis session and mark the analysis as non-malicious.

The feedback was conducted in two ways, first we observed the participants activities and second asked the participant after the first part about the recording process. That resulted in the following insights:

Observation of the participant:
The participant started recording and where able to load and view the ProcessTree. The questions to filter shown activities confused the participant because it was unclear whether to hide or show the listed activities. The participant pressed first on “Load” button in the NetFlowChart before selecting the correct timeframe. That led to a request to the backend because the date pickers had some default values.

Direct feedback from the participant:
The participant was able start the analysis session but missed a clear indication that a recording is ongoing. Commenting worked, but the participant mentioned it was initially difficult to view the “Comment” button to open the dialogue.

A session is a short break between the recording and viewing where questions will be asked about recording. The conversation is there for two reasons: firstly, to prepare the participant for the new task and secondly, to make the interactions less memorable. The participant should then use the graphic representation to retrace his interactions.

Part 2: View a session

- Make sure that the dashboard has no views opened.
- Click in the Menu on Open Viewer.
- Enter the session id "133" and press load.
- View the components used in the session graph.
- Hover over tasks to view the tooltips.
- Double click on a comment.
- Select a script task and double-click on it to change to the *Live Session Player.*
- Use the navigation element to go through the analysis session.
- Close the session.
- Click on open session and select the session via the analysis session list.
- Close the session again.

Observation of the participant:
The participant fetched the session and initially had issues to selected the correct tool for arrangement. The participant could view the recorded interactions and used the tooltips for further details. The BPMN term “script task” was not explained and the chosen BPMN symbol for these tasks was not understood as a script task to compensate for the missing explanation. We had to explaining it to the participant. In regard of usability the participant struggled with double clicking the task, because clicking also enables changing the task description. The participant tried it out several times before the dialogue to switch to the *Live Session Player* appeared. The change to the *Live Session Player* occurred an error. The participant reloaded the page and opened the session via the “Open Session” button to start at the first step. But then could use the navigation elements to go over the sequence of steps.

Direct feedback from the participant:
The participant could identify the analysis start and comprehend which interactions lead to which graphical representations in the *Session Viewer.* The representation enabled the participant to view changes to other components within a session and a which view where used. The participant mentioned that it was difficult to click for changing to the *Live Session Player* and figure out what a script task was. The use of the navigation in the *Live Session Player* was no problem but it sometimes was difficult to see what step was executed. For example, the filter changes like changing the status of a checkbox in the ProcessTree has been lost in the process. Additional the participant mentioned that the toolbar on the left was confusing because there was no task related to it.

This feedback pointed out that the prototype of analytical provenance can be improved until the evaluation in WP6. This refers especially to the elements that caused confusions of the participant.

7 Conclusion

In this deliverable, a prototype for analytical provenance in the SAPPAN dashboard has been presented. First, we explain the use of analytical provenance regarding the SAPPAN dashboard and introduce the requirements and users. As the core part of this deliverable, the prototype shows the implementation of user interaction tracking within the user interface and the visual components to collect provenance information. On the basis of collected interaction data, two representations have been created to enable analysts: First, to overview recorded analysis sessions in a graph-based BPMN layout and second, to comprehend a session step-by-step in the dashboard's analysis area. Analysts can use this representation to track their interactions, compare it with incident response playbooks of WP4 or discuss it with colleagues to optimise a SOC's incident response process.

8 References

[1] Dennis P. Groth and Kristy Streefkerk. 2006. Provenance and annotation for visual exploration systems. *IEEE transactions on visualisation and computer graphics* 12, 6, 1500–1510. DOI: https://doi.org/10.1109/TVCG.2006.101.

[2] Jeffrey Heer and Ben Shneiderman. 2012. Interactive dynamics for visual analysis. *Commun. ACM* 55, 4, 45–54. DOI: https://doi.org/10.1145/2133806.2133821.

[3] Phong H. Nguyen, Kai Xu, Ashley Wheat, B. L. W. Wong, Simon Attfield, and Bob Fields. 2016. SensePath: Understanding the Sensemaking Process Through Analytic Provenance. *IEEE transactions on visualisation and computer graphics* 22, 1, 41–50. DOI: https://doi.org/10.1109/TVCG.2015.2467611.

[4] Chris North, Remco Chang, Alex Endert, Wenwen Dou, Richard May, Bill Pike, and Glenn Fink. 2011. Analytic provenance. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA '11*. ACM Press, New York, New York, USA, 33. DOI: https://doi.org/10.1145/1979742.1979570.

[5] Eric D. Ragan, Alex Endert, Jibonananda Sanyal, and Jian Chen. 2016. Characterising Provenance in Visualization and Data Analysis: An Organisational Framework of Provenance Types and Purposes. *IEEE transactions on visualisation and computer graphics* 22, 1, 31–40. DOI: https://doi.org/10.1109/TVCG.2015.2467551.

[6] Kai Xu, Simon Attfield, T. J. Jankun-Kelly, Ashley Wheat, Phong H. Nguyen, and Nallini Selvaraj. 2015. Analytic provenance for sensemaking: a research agenda. *IEEE computer graphics and applications* 35, 3, 56–64. DOI: https://doi.org/10.1109/MCG.2015.50.