Sharing and Automation for
Privacy Preserving Attack Neutralization

(H2020 833418)

D6.3 SAPPAN demonstrator (M33)

Published by the SAPPAN Consortium

Dissemination Level: Public

H2020-SU-ICT-2018-2020 – Cybersecurity

Document control page

Document file: D6.3
Document version: 1.0
Document owner: Martin Zadnik (CESNET)

Work package: WP6
Task: T6.2 SAPPAN demonstrator
Deliverable type: Demonstrator
Delivery month: M33
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	Martin Zadnik (CESNET)	2021-12-16	Outline of the document
0.2	All partners	2022-01-04	Filling out description of components
0.3	All partners	2022-01-18	Fine-tuned demonstration story
0.4	Martin Zadnik (CESNET)	2022-01-21	Draft version
0.5	Martin Zadnik (CESNET)	2022-10-27	Internal review version
0.6	Martin Zadnik (CESNET)	2022-10-28	Integrating reviewers' comments
1.0	Martin Zadnik (CESNET)	2022-01-29	Final version for submission

Internal review history:

Reviewed by	Date	Summary of comments
Gabriela Aumayr (HPE)	2022-01-27	Technical, story and grammar
Milan Čermák (MU)	2022-01-27	Technical, story and grammar

Executive Summary

The goal of Task T6.2 is to collect and prepare individual components developed within the work packages WP3, WP4 and WP5 for deployment. As a part of this task, we prepare a demonstrator of the SAPPAN results. In this deliverable D6.3, we document the results, their basic functionality, interfaces, availability. We provide a schema of the demonstrator composed of the results and explain how the results fit together. Subsequently, we build an artificial story around the demonstrator to present the results in an easy to understand way to the parties external to the project such as stakeholders.

Table of Contents

1 Introduction 5

2 Description of demonstrator 6

3 Demonstration story 19

3.1 Artificial story of a malware incident (NIST perspective) 19

3.2 Perspective of SOC manager 21

4 Summary and plans 31

5 References 32

1 Introduction

The SAPPAN project has developed multiple cybersecurity results. One of the goal of the Task 6.2 is to prepare these results to be presented to various communities, such as End User Committee (EUC), potential users/stake-holders (e.g. Security Operation Center members, Managed Service Providers, as well as network and computer administrators) and the project reviewers.

There are several aspects that render the integration and demonstration challenging:

- The set of developed results is diverse. The diversity of the results is caused by the proposed project objectives which range from scalability data processing, federated threat detection, privacy-preserving CTI sharing, modeling incident response and recovery information, semi- and automated response up to advanced visualization.
- The set of the results covers several phases of the incident response process as defined by NIST [1] - preparation, detection, assessment, and handling. Therefore the results cannot be shown at once.
- Some of the results work at the local level (within an organization) while some work across organizations and require multiple organizations to cooperate.
- Some of the results are developed as an open-source while other are closed source and their interface vendor specific.

To address these issues, we introduce a schema of the demonstrator. The schema groups together the results based on a data flow and depicts the communication flow between the results. Subsequently, we decided to present the demonstrator using a story about handling of a particular incident. The flow of the story follows the NIST incident response process and we believe it is a fitting perspective that can explain and convey relevant information to the users that are external to the project consortium.

We provide a schema and the description of the demonstrator including its components in Chapter 2. We describe the story including individual demonstrations in Chapter 3. We conclude with summary and our future plan Chapter 4.

2 Description of demonstrator

The Figure 1. displays the schema of the SAPPAN demonstrator. The results to be demonstrated are depicted as a solid-line box. With a dashed-line we marked specific components that are helpers in the demonstrator – *End-point agents*, *FSC RDR Portal* and *MISP*. The *End-point agents* were improved within the project to support processing scalability but the agents will not be demonstrated, however they will produce input data for the *Anomaly detection*. Another specific component is the *MISP* sharing platform [2]. This platform is a third-party tool developed by CIRCL.LU and we use it in the SAPPAN architecture to share information across organizations. To this end, we have prepared two dedicated MISP objects – an object to carry a cybersecurity playbook [3] and an object to carry a machine learning model [4].

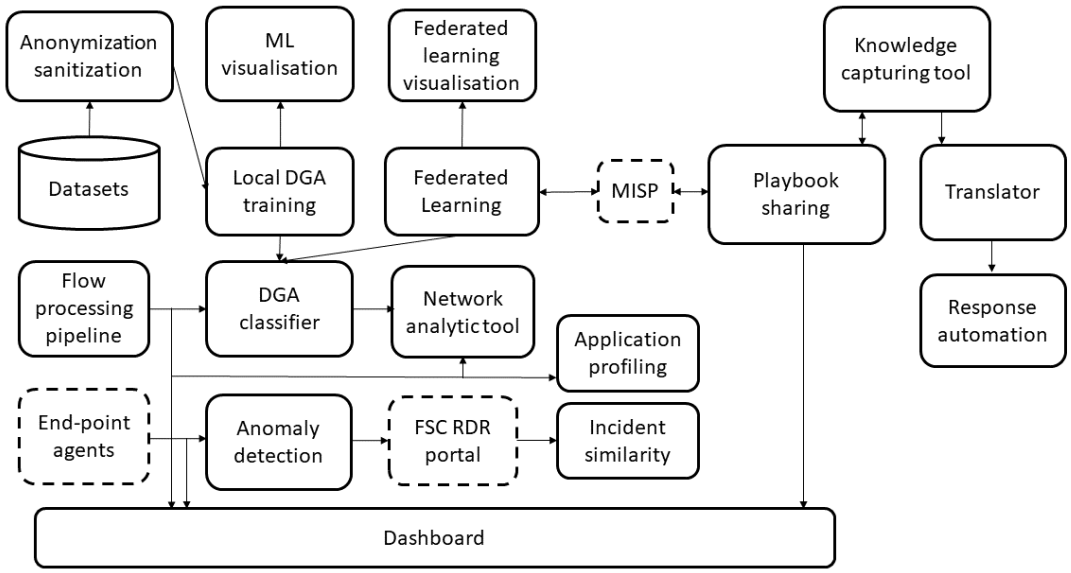

Figure 1: Schema of the SAPPAN demonstrator

The description of the results (from now on components) starts from the left side of the figure. The *End-point agents* generate data about events at the end-points. The end-point data are consumed by the *Anomaly detection* component. The output of *Anomaly detection* is assessed by an analyst in *RDR FSC portal* to establish if an incident has happened and what actions should be taken further. The analyst is supported by the *Incident similarity* component. The *Incident similarity* component clusters and looks up similar incidents and how they were handled and provides such information to the analyst. Additionally, the analyst can use the *Dashboard* component to observe the end-point data. In parallel, to the end-point data, there is a branch of components that work with network monitoring data.

The network monitoring data are processed by the *Flow processing pipeline* first. The *Flow processing pipeline* consists of multiple components capable of processing a large volume of network monitoring data in a scalable way. Its output is either visual, displayed to an analyst, or machine-readable in the form of a stream or a query/answer interface. The *Flow processing* provides data to the *Application profiling* which infers contextual data for the analyst. The *Network analytic tool* enables interactive and intuitive inspection of the flow data.

The pipeline also produces a feed to the *DGA classifier*. The *DGA classifier* was trained offline by the *Local DGA training* or *Federated learning* with the support of the machine learning *(ML) visualisation* or the *Federated learning visualisation* components. The ML is dependent on the annotated dataset which was anonymized by the *Anonymization/sanitization* component. The *MISP* component is used to convey trained DGA classification models, not only during the federated learning but also as a new form of privacy-preserving cyber threat intelligence, i.e. the model becomes the new Indicator of a Compromise (fingerprint of bad behaviour).

On the right side of *MISP*, there is a set of components focused on response and recovery. The *Playbook sharing* component facilitates the sharing of the cybersecurity playbooks. It can consume the output of the *Knowledge capturing tool* and vice versa. The *Knowledge capturing tool* provides a graphical interface to an incident handler to create or modify playbooks. Its output can be translated by the *Translator* component into an Apache Airflow workflow which is interpretable by a machine. But before the workflow is deployed it must be customized by the handler to the particular specifics of a given organization and serves to correctly guide the response in the *Response automation* component (two distinct examples of response automation were prepared). A more detailed description of the components, their interfaces as well as availability is depicted in the following table.

Result	Function	Input and output	Links to supportive materials and availability
Flow processing pipeline	The fast and scalable data processing pipeline was implemented as proof of concept based on the methodology presented in WP3 . HPE further developed this proof of concept to be production ready and applied it to a wide range of network data feeds: traffic from webproxy, firewall etc.	Input: Data feeds that can be processed with the pipeline can be in various formats: Syslog, JSON, and in various amounts: from 100 events per second to 50K events per second. Output: Processed and aggregated results are available both in Kafka as short-term storage and in a database for long-term storage. The data format uses AVRO schema (short-term storage) or relational database schemas (long-term storage) to represent the processed, aggregated, enriched data. Metrics and logging to monitor the behaviour of the processing pipeline are available in a dashboard.	Deliverables: D3.3 Available in SAPPAN git: https://gitlab.fit.fraunhofer.de/sappan/hpe
Local DGA training	This allows to train a deep learning model for DGA classification based on NX domains. The input consists of labeled training data (malicious and benign NX domains) and the result is a classifier (binary- or multiclass) for DGA classification.	Input: Training data consisting of NX domains and labels (malicious or benign),. In case of multiclass, the data also is labeled with the DGA family. Output: A tensorflow model in HDF5 format.	Publications: https://dl.acm.org/doi/10.1145/3407023.3407030 https://dl.acm.org/doi/10.1145/3407023.3409190 Deliverables: D3.4 Available in SAPPAN git: https://gitlab.fit.fraunhofer.de/sappan/sappan_dga_poc

Result	Function	Input and output	Links to supportive materials and availability
DGA classifier	The classifier is a machine learning model that was trained on labeled data. The model takes NX domains as input and outputs a confidence score (binary-class) or the DGA family that the domain most likely corresponds to (multiclass).	Input: One or more NX domains that need to be classified. Output: A confidence score (binary-class) or the most probably DGA family (multiclass) of each input domain.	Publications: https://dl.acm.org/doi/10.1145/3407023.3407030 https://dl.acm.org/doi/10.1145/3407023.3409190 Deliverables: D3.4 Available in SAPPAN git: https://gitlab.fit.fraunhofer.de/sappan/sappan_dga_poc
Application profiling	The tool detects applications (or OS) running on hosts based on network traffic. The tool is rule-based, which means that rules need to be added for each applications. However, the rule-generation process is automated.	Input: Set of rules for all applications (or OS) that should be detected. Network traffic (e.g. in form of a pcap or stream), Output: For each IP address in the data the list of applications (or OS) that were detected. The data is unstructured either in text form or presented in a GUI.	Deliverables: D3.4 Available in SAPPAN git: https://gitlab.fit.fraunhofer.de/sappan/application-profiling
Analytic tool	A demonstration of a novel approach to exploratory network data analysis based on associations stored in a	Input:	Deliverable D3.5 Algorithms for analysis of cybersecurity data

Result	Function	Input and output	Links to supportive materials and availability
	graph database. The tool allows the analyst to load PCAP (full packet capture) or IP flow data and analyze them as graph data using an interactive visual interface.	Alert data that can be partially processed automatically to find identifiers (source/destination IP address, domain name, HTTP URI, etc.) and partially must be inspected manually to derive the context of the data. PCAP (full packet capture) or IP flow data to be analyzed. Output: Visual interface allowing an analyst to browse network traffic data and examine the alert and all related events. Output data are in the form of unstructured observations (other infected machines or details about the alert) resulting from manual analysis of the alert in network traffic data.	Publication: http://dx.doi.org/10.5220/0010581807850790 Available in SAPPAN git: https://gitlab.fit.fraunhofer.de/sappan/granef
Knowledge capturing tool	A demonstration of a user interface to facilitate the capturing, viewing and editing of incident management knowledge. With this tool, an analyst can capture the response and recovery steps into structured and standardized machine-readable playbooks. The outputs are in SAPPAN and CACAO format. It is based on Semantic MediaWiki and provides a BPMN graphical representation of the playbooks.	Input: Playbook steps from analyst. JSON file (SAPPAN playbook). Output: Playbook in SAPPAN format (JSON) and a graphical representation in BPMN. CACAO playbook (JSON).	Deliverable D4.2 and D4.3 Available in SAPPAN git: https://gitlab.fit.fraunhofer.de/sappan/dockersmw

Result	Function	Input and output	Links to supportive materials and availability
Playbook sharing	The tool performs writing of playbook into MISP in order to be shared within the MISP community: connects to MISP, loads the playbook from a file, sends the resulting playbook into MISP. as well as reading a playbook from MISP: connects to MISP, filters the events according to the rule on a playbook parameters, reads and stores the playbook into a file.	Input: JSON File Native format is CACAO playbook as defined at: https://docs.oasis-open.org/cacao/security-playbooks/v1.0/cs02/security-playbooks-v1.0-cs02.html But other JSON based playbook format can be used as well but metadata must be filled in manually. Output: PyMISP interface to connect to MISP. Format: MISP Security playbook object containing playbook metadata and the playbook itself. Specification of the object is available in: https://www.misp-project.org/objects.html#_security_playbook	Deliverable D5.8 Sharing response handling information - final version Publication: https://arxiv.org/pdf/2110.10540.pdf Available in SAPPAN git: https://gitlab.fit.fraunhofer.de/sappan/sappan/-/tree/master/sappan-misp/playbook_sharing
Translator	The tool translates a CACAO playbook into Apache Airflow's DAG format: Checks validity of CACAO playbook Creates tasks and callable functions for each workflow step Sets up task execution order Automatically selects proper operator types	Input: JSON File Format is CACAO playbook as defined at: https://docs.oasis-open.org/cacao/security-playbooks/v1.0/cs02/security-playbooks-v1.0-cs02.html Output:	Deliverable D4.7 Algorithm to automate recommended response and recovery actions without human operators, final version Available in SAPPAN git: https://gitlab.fit.fraunhofer.de/sappan/sappan-t4.4/-/tree/DarrelDerDagDolmetscher/DarrelDerDagDolmetscher

Result	Function	Input and output	Links to supportive materials and availability
	Copies metadata from playbook (Author, labels, title) Prepares step functions with information from playbook as comments (description, manual commands) or as prebuilt code (http, bash, ssl)	Python file (.py) Format: Apache Airflow DAG as specified in their documentation: https://airflow.apache.org/docs/apache-airflow/stable/tutorial.html	
Response automation	This is an integration of TheHive and Cortex with Apache Airflow to implement a workflow that automates response and recovery actions through calling APIs.	Input: JSON At least the following fields: Source/ip: The Source-IPv4 Address of the affected asset Domain_hostname: The destination domain being accessed Enrichment/domain_hostname/dga/score: The DGA score of domain_hostnameipfix:destinationIPv4Address. Output: Various API calls to TheHive	D4.7 Algorithm to automate recommended response and recovery actions without human operators, final version#DGAShowcasePrototype DL proprietary code repository
Response automation	This tool runs a workflow on a suspicious file. It is an integration of a custom evaluator, an open-source CTI aggregator Intel Owl and an orchestrator Apache Airflow.	Input: Web application GUI file uploader Input data format: a file of arbitrary format, uploadable through browser file upload interface Output:	References to mentioned tools https://airflow.apache.org/ https://intelowlproject.github.io/ https://min.io/

Result	Function	Input and output	Links to supportive materials and availability
	The tool utilizes Intel Owl to collect known CTI about the file as well as related IPs and domains. Collected CTI is analyzed, the tool decides, whether the file is malicious or not. Related domains and IPs that were found malicious are black-listed. Finally, the tool provides the user with the analysis results in a form of MISP event and PDF report.	Output interface: a link to a downloadable report served by the web application GUI + MISP UI Output format: PDF file + MISP Event object	Available in SAPPAN git: https://gitlab.fit.fraunhofer.de/sappan/malware-evaluator
Anonymization/sanitization	A command line tool, which can sanitize training data sets and individual samples via similarity-perserving encodings. It provides different encodings styles and allows to perform decoding attempts on individual encodings to evaluate the choice of parameters.	Input for data set encoding: CSV (one sample per row, label delimiter can be given as a parameter) and encoding parameters. Input for encoding individual samples or parameter evaluation. Format: string containing the individual sample, encoding parameters. In addition to encoding generic string data, the tool also provides a specialized encoding for URLs as input data. Output for data set encoding: sanitized data set in the same format as the input Labels are preserved	Deliverables D3.6 and D3.7 Cybersecurity Data Abstractions (initial and final version, respectively) Available in the SAPPAN GitLab: https://gitlab.fit.fraunhofer.de/sappan/sappan-t3.4/-/tree/master/urlgen

Result	Function	Input and output	Links to supportive materials and availability
		Output for encoding individual samples: the respective encoding. Output for the parameter evaluation: statistics of the performed decoding attempt.	
Dashboard - analysis	A dashboard that displays endpoint and network data. It contains visualisations that an analyst can arrange flexible to inspect the behaviour of processes in an endpoints process tree or view the data.	Input: It consumes RDR endpoint sensor data and IP-Netflow captures stored in an ElasticSearch instance. Furthermore, it can use the SAPPAN playbooks to show a playbook in BPMN. Output: The dashboard can trigger a remediation workflow and can exchange notifications with other dashboards over SignalR The analysts can consume the visual representation.	D6.1 SAPPAN dashboard (Initial version) Available in the SAPPAN GitLab: https://gitlab.fit.fraunhofer.de/sappan/sappan-dashboard
Dashboard - provenance	A provenance capture tool is integrated in the dashboard and its visualisations to record visualisations and the dashboards layout status and create a visual representations of it.	Input: It consumes the interaction events within the dashboard's frontend and saves them in a command pattern Output: Visual representation of the interactions sequence.	D4.9 Demonstrator for tracking provenance in visual analyses, final version Available in the SAPPAN GitLab: https://gitlab.fit.fraunhofer.de/sappan/sappan-dashboard
Federated learning	This allows to train a deep learning model for DGA classification based	Input:	Publications:

Result	Function	Input and output	Links to supportive materials and availability
	on NX domains. Similar to the local DGA training, the input consists of labeled training data (malicious and benign NX domains) and the result is a classifier (binary- or multiclass) for DGA classification. However, the classifier is not trained on a single dataset, but on multiple datasets from different parties in a federated fashion.	Training data consisting of NX domains and labels (malicious or benign). In case of multi-class, the data also is labeled with the DGA family. The training data consists of multiple datasets from different parties. Output: A tensorflow model in HDF5 format.	https://dl.acm.org/doi/10.1145/3474374.3486915 Deliverables: D5.1 D5.2 D5.3 D5.4 D5.5 D5.6 Available in SAPPAN git: https://gitlab.fit.fraunhofer.de/sappan/sappan-dga-misp
ML classifier visualization	Visualization tool for analysis of deep learning models for DGA classification Gives an overview of the related dataset and model performance on this dataset Allows for analysis of dynamic subsets using Dimensionality reduction	Input interface: file. Input format: CSV containing domains, class labels and model outputs (prediction, predictions score, ground truth label, ground truth score). Output interface: browser. Outpu format: visualizations (Website).	Deliverables: D3.8 Demonstrator of Visual Support for Designing Detection Models (Initial version) D3.9 Demonstrator of Visual Support for Designing Detection Models (Final version) Publication:

Result	Function	Input and output	Links to supportive materials and availability
	Clustering Decision Tree as a local surrogate explainer		https://sappan-project.eu/wp-content/uploads/2020/11/fbecker-interpretable-vizsec-20201.pdf Available in SAPPAN GitLab: https://gitlab.fit.fraunhofer.de/sappan/dga-visualization
Federated learning visualization	Visualization tools for analysis of federated learning scenarios Gives an overview of performance during training (for each party) Shows similarity of weights per layer between parties Uses clustering and dimensionality reduction to identify clusters (based on prediction scores) of interesting data in a given dataset	Input Interface: application GUI. Input Format: JSON containing meta-data, NPY/BIN files containing weights - CSV containing domains, CSV containing prediction scores. Output Interface: C++ Desktop Application. Output Format: Visualizations (Desktop Application).	Deliverable: D5.10 Demonstrator for visualisation support for distributed and federated learning Available in SAPPAN GitLab: https://gitlab.fit.fraunhofer.de/sappan/federated-learning-vis
Anomaly detection	A set of models for detecting anomalous endpoint events of specific types (e.g., process launch, process and thread access). A method for presenting connections among detected anomalies as a graph for higher detection reliability.	Input format: FSC JSON object representing end-point events, possibly enriched with analysis results (such as anomaly scores). Output: FSC JSON object representing end-point events enriched with analysis results (such as anomaly scores). FSC proprietary format for representing graphs of detected anomalies.	Deliverables: D3.4 D5.3 D5.4 Proprietary, FSC RDR codebase

Result	Function	Input and output	Links to supportive materials and availability
Incident similarity	Methods for (i) computing similarity between security incidents; (ii) finding the incidents most similar to a given one; (iii) clustering incidents observed in the monitored endpoints.	Input: (i) a pair of incidents; (ii) a given incident and the repository of the stored ones; (iii) the repository of the stored incidents and the clustering hyper parameters. Output: (i) incident similarity value; (ii) the set of the most similar incidents and their meta-data (e.g., whether they were false or true positives); (iii) incident cluster representation.	Deliverables: D4.4 D4.5 Proprietary, FSC RDR codebase

The components were developed following continuous integration principles. We used gitlab environment as a means of collaboration for code development. To simplify the deployment of the components docker images were prepared together with Ansible scripts.

In case of productional deployment, we must consider that the organizations have already deployed various cybersecurity tools and devices. The goal of the demonstrator is to present the capabilities and not to replace the existing toolset. Therefore, we envision that the organizations will select only the relevant parts of the demonstrator or just even a specific component and deploy it in their infrastructure in parallel or on top of the existing tools to increase their cybersecurity capabilities.

3 Demonstration story

The SAPPAN demonstrator integrates a versatile set of components jointly developed throughout the project. We prepared an artificial story to introduce the demonstrator in an easy-to-understand way. The demo story tells two perspectives: one from the perspective of the NIST response cycle and the second from the perspective of a cybersecurity manager.

3.1 Artificial story of a malware incident (NIST perspective)

The first signs of an attack were detected by the FSC Rapid Detection and Response (RDR) service. The attack was run professionally and the attacker succeeded in staying "below the radar" until the lateral movement stage, when alerts were produced by the RDR anomaly detection models. A graph showing connections among the detected anomalous events (and with some other events) was built, and after inspecting it an incident handler (a security analyst handling the case) decided to form a security incident (1).

The handler then used the FSC incident similarity and clustering mechanism to look for earlier incidents similar to the one under investigation. Several similar incidents were found, some of those were confirmed true positives (attacks) and some were unresolved, which increased the incident handler's confidence that the case is an attack and has to be investigated further (2).

It was discovered that the attack started with an accountant downloading an infected file. When that file is opened, the malware exploits a vulnerability and infects the machine. The malware uses domain generation algorithm to generate domain names of its C2 server. The malware tries to resolve multiple DGA names which causes multiple unsuccessful resolutions (NX DNS records) in the network traffic. This traffic was observed via the HPE flow processing pipeline (3) and analyzed by RWTH DGA classifier. The detection results were reported to the incident handler via the Dashboard (4).

The handler proceeded with the incident handling in the Dashboard by inspecting the process tree, flow chart of the machine. Based on the graphical findings, the incident handler decided to start the response workflow regarding DGA remediation (5). The incident handler fires DGA remediation response workflow from the Dashboard. The DGA remediation response workflow includes branching upon asset criticality. In the story. As the asset is deemed critical the workflow asks incident handler to analyse incident manually. As a part of the remediation (6), the Analytic MU tool is used to perform the manual analysis of the packet capture. The manual analysis using MU Analytic tool further confirms the malware and that another machine could be infected.

The confirmation of the malware is fed back manually to the DGA remediation workflow which then blocks the domain access. The handler initiates manual analysis of the machine. During the manual analysis the incident handler fires analysis of the email attachment by MU response workflow (7). The MU workflow confirms the attachment contains DGA malware. As an alternative to host data, the incident handler can use network-based application profiling to get additional context of the infected machine and find out what applications are running on it (7.5).

Now we roll back in time by a week prior to the incident to the preparation phase of the handling cycle. The preparation allows to implement and deploy detection and response measures.

Namely, the team prepared and deployed DGA detection using a DGA classifier trained on their local data (8). The team investigated the performance of the classifier using advanced SAPPAN visualisation techniques (9). Later on they decided to improve the classifier by participating in the federated training (10)(11). They also decided to share it with other organization who can benefit from the trained classifier which serves as a novel Indicator of Compromise.

The team also looked for possibilities of automated response to ensure faster response to an incident. As the organization participates in a MISP sharing community in which also playbooks are shared, the team used the SAPPAN sharing playbook tool to filter all playbooks with labels malware and DGA (12). They downloaded these playbooks from MISP. The DGA playbook from DL is edited in Knowledge capturing tool to add an additional step in the playbook to reflect organization specific process (13). Subsequently, the playbook was translated into Airflow workflow by the Translator (14). Then, the SOC team used the translated workflow, manually customized it to organization-specific environment and toolset and prepared it for being used during incident handling.

3.2 Perspective of SOC manager

The story can be viewed also from the perspective of a SOC manager who returns to the office after a long absence (e.g. due to COVID-19). The manager knows that before her absence there were multiple issues. The manager meets with members of the SOC and discusses with them the issues:

- She outlines the issue to remind the person of the issue.
- She asks a particular question that fits the issue.
- She receives an answer, ideally, including a demonstration.

The issues and questions cover the SAPPAN results that we want to demonstrate. This perspective follows the use case of malware using DGA, and follows the NIST perspective. The number in brackets at each demo reference back to the NIST perspective story.

(1)

Issue: New malware infected our machine and we need to discover it as soon as possible.

Question: How to detect suspicious behavior at the end-point?

Answer: Combination of multiple anomaly detection models.

Demo: Anomaly detection.

Starting point	**Keypoints**	**End point**
Security monitoring in endpoints by FSC RDR.	Several anomaly detection models for endpoint events of relevant types (e.g., process launch, process access, thread access). Representation of anomalous (and related) events in a graph form for supporting SOC personnel.	Security alerts, event graphs.

(2)

I: The handler needs to figure out how to react to an incident (e.g. needs to rule out false positive).

Q: How to capture and utilize information obtained through previous incident handling?

A: Incident similarity clustering.

D: Incident similarity.

Starting point	Keypoints	End point
New incident data and the repository of earlier incidents.	Incident similarity and clustering mechanisms.	A set of incidents similar to the one under investigation.

(3)

I: Large amounts of flow data need to be processed fast (almost real-time) and in a scalable way.

Q: How do we store and query all the data?

A: Scalable processing of IP flow and log data.

D: Flow processing pipeline.

Starting point	Keypoints	End point
Continuous data flow processing from multiple end-points.	Fast and scalable processing pipeline runs continuously to process streams of networking data. Data can be stored almost raw in short term storage or aggregated in long term storage. Various data retention periods are defined for individual data feeds.	Visualization of flow data and DGA detection

(4)

I: A host visited DGA domain.
Q: Which process is responsible at the end-point
A: Correlation of end-point data with network data or application profiling,

D: Dashboard.

Starting point	Keypoints	End point
Continuous data flow processing pipeline is integrated with the DGA classifier and run periodically (e.g. every 5 minutes). Suspicious end-points that egressed to maliciously classified domains are further analysed.	Overview analysis of the end-point‘s process hierarchies, process behaviour and network behaviour.	Manual overview analysis about end-point and netflow data is done, an analyst decides to send the alert to the remediation tool.

(5)

I: We need to automate response to DGA detection to take measures as soon as possible.
Q: Can we automate it and how?
A: Let‘s use DGA remediation workflow

D: Response automation.

Starting point	Keypoints	End point
RESTAPI of Remediation Playbook sending json data to end-point	Since a high-importance asset is affected, the playbook will pause and wait from input from the analyst. The TheHive case will already provide some threat intelligence (virustotal) on IP and domain to analyst. The analyst determines whether the communication should be blocked or not based on its experience and knowledge and then blocks or allows the communication in the workflow.	The analyst has decided to block the traffic and the case in TheHive is marked as closed and "True Positive".

(6)

I: We need additional context to observe the malware outbrake.

Q: How to correlate findings of manual analysis with flow records and CTI source.

A: Let's use graph analysis to investigate our data.

D: Network analytic tool.

Starting point	Keypoints	End point
Request for manual analysis of the incident using monitored IP flow data. Load of IP flow data from selected time range and relevant CTI.	Graph-based analysis approach allowing to simply follow connection and extracted data. Mutual analysis of IP flow data and CTI.	Malware infection verification (based on communication with C&C and hash from CTI). Verification that no other host have been infected (based on relevant IoC from CTI).

(7)

I: It is too much labour to obtain additional data for analyzing content of the infected machine.

Q: How to automate this process.

A: Similar to response workflow, let's use file analysis workflow.

D: Response automation.

Starting point	**Keypoints**	**End point**
Request for manual analysis of the malware file and possible mitigation. We have the suspected file.	Malware analysis based on the CTI. Automatic mitigation and victim search. Automatic CTI sharing via MISP and report.	Malware was analyzed, all possible related observables were found. Mitigation was run. Relate observables of type IP matched against netflow data to find any communicating hosts in the network.

(7.5)

I: No end-point data is available for the machine but we need more context.

Q: How to classify the host further?

A: Use Application profiling tool to find the applications running on the infected host

D: Application profiling.

Starting point	**Keypoints**	**End point**
Context is needed for infected host (e.g., is it a client or server?). Recent network data (DNS) of the host is available or can be captured.	DONUT tool can use DNS traffic for analysis of running applications on the host.	List of applications running on the host (profiles of applications of interest are needed.

(8)

I: Malware is using artificial domains to figure out new IP address of its C2 server.

Q: How do we detect artificial domain names, so that we can detect compromised machines?

A: Let's train our DGA ML classifier locally.

D: Local DGA training.

Starting point	**Keypoints**	**End point**
We need a DGA classifier and have a dataset available which includes NX domains labeled as malicous or benign. The DGA classifier should be trained locally based on a local dataset.	The classifier is trained locally. This allows to use a local dataset potentially including private data, because only the trained model needs to be shared afterwards.	The classifier was trained based on a local dataset and can be shared such that other parties can use it for DGA detection.

(8.5)

I: We want to train a classifier for the same purpose and want to share it with parties we do not fully trust.

Q: Can we train a similar classifier which does not leak private information?

A: We sanitize the data first.

D: Sanitization command line tool.

Starting point	**Keypoints**	**End point**
Sets of malicious and benign samples are available. The benign samples should not be reconstructable by any party receiving the trained classifer.	The benign training data is privacy critical, as it reveals information about browsing behavior in the source network (e.g., typos of domains, humans wanted to access). Once a classifier is shared, control over it is lost. The classifier should hence also be protected against future advancements in the area of membership inference. The sanitization tool can sanitize given data in a similarity-preserving way (crucial property for machine learning). Protection is provided by collisions in the encoding space, and (if parameterized) noise. The tool is called via command line and provides various parameters. The tool can also be used to sanitize data, which should be directly shared. It also provides an implementation of an attack, which allows the operator to test, whether the chosen parameters provide a suitable protection for sharing of sanitized data.	A classifier trained on sanitized data.

(9)

I: We do not understand how DGA classifier works, what if it will detect legitimate domains.

Q: Can we figure out how DGA ML classifier works?

A: By visualization of samples.

D: ML visualisation.

Starting point	**Keypoints**	**End point**
Model has been developed, then: Developer wants to verify strengths/weaknesses of the classifier. Developer wants to communicate these strengths/weaknesses. (new instance should be analyzed, e.g. from an alert where the operator is not sure whether its actually malicious).	Visual analysis of the model. Feature-based overview. Formulating interesting data subset. "Deeper" analysis.	Formulation of strengths and weaknesses of the particular model.

(10)
I: DGA has false positives.

Q: Can we make it better?

A: Let's use SAPPAN federated learning.

D: Federated learning.

Starting point	**Keypoints**	**End point**
We already have a classifier that was trained locally using a private dataset However, the classifier produces false positives. Other parties also have labeled datasets for training available but they don't want to share the data directly. Hence, we train a model using federated learning, which uses multiple training datasets without the need of sharing them, to improve the performance of the classifier.	Each party that has a training dataset trains a local model and shares the weight updates (compared to a shared baseline) via MISP. All weight updates are aggregated into a global model without the need of sharing private datasets.	A global classifier that was trained on data from multiple parties with the goal of increased accuracy.

(11)
Q: Does our new classifier perform better?

A: collaborative learning evaluation

D: Federated learning visualization.

Starting point	Keypoints	End point
Developer wants to gain understanding of federated training process for the DGA use case. Developer wants to verify strengths/weaknesses of the classifier. Developer wants to communicate results.	Visual analysis: Investigate general performance development during training. Compare weight trajectories during training. Investigate models for specific dataset Where do differences lie and how do they affect the federated classifier (best & worst & average case).	Understanding of federated learning for the DGA use case (best & worst & average case).

(12)
I: Organizations do not know how to respond or their response is not adequate, consistent, etc.
Q: How can share our response?
A: Sharing of playbooks.

D: Playbook sharing.

Starting point	Keypoints	End point
Sharing among organizations using CACAO.	MISP data model for CACAO. Automated metadata extraction. Filtering upon metadata.	Playbook available in JSON. To be customized for organization.

(13)

I: We need to be able to customize and create playbooks.
Q: Is there a graphical way to create playbooks and store it in a machine readable format?
A: Let's edit DGA remediation workflow to customize it to our infrastructure.

D: Knowledge capturing tool.

Starting point	Keypoints	End point
Steps to create a machine-readable playbook for DGA detection.	Playbooks need to be stored and edited in a structured way. Playbooks need to be machine-readable. The tool should facilitate the process of creation and editing of a playbook.	A SAPPAN format playbook is created (JSON). A BPMN graphical representation of a playbook is created.

(14)

I: We have a lot of DGA detection, it takes a lot of manual labour to execute the steps of the playbook.
Q: Can we automate it and how?
A: Let's create DGA workflow in Airflow.

D: Translator.

Starting point	Keypoints	End point
High-level machine-readable CACAO playbook for DGA detection. Goal: Automation of the workflow described by the playbook.	The input playbook describes a workflow, but on a more abstract level than required for automation. For automation, Apache Airflow is used. Implementing the workflow manually requires a lot of effort The first step (translation of playbook to Airflow workflow structure) consists of repetitive tasks. The translation tool can retrieve the playbook structure of a given playbook and translate it to an Airflow workflow structure. This automatically creates the necessary generic methods and the interconnection between the methods. Afterwards, the methods need to be properly implemented. This step is infrastructure-specific, and the given playbook does not provide sufficient information to allow for automation of this step.	The Airflow workflow structure has been created automatically.

We prepared video demonstrations that cover the components as they appear in the story. We decided to use the video demonstration mainly due to three reasons. We want to show a high number of results, and this needs guidance for a first-time user. Also, some of the tools offer only a command-line interface. Lastly, the video is able to convey the story itself to a user better than other types of demonstrations.

4 Summary and plans

The demonstrator consists of various components to introduce major results developed within the SAPPAN project. The components differ from the perspective where they appear in the NIST response cycle. The description of the demonstrator follows this cycle and describes functionality, interface and availability of the components including their related documentation. Besides the demonstrator, we prepared a demo story to present the demonstrator to an external audience.

Our short-term plan is to organize EUC meeting during February 2022 and to introduce the demonstrator to the EUC. We will use the prepared video presentation as the meeting will be held via videoconference. Our plan is to collect feedback on the results which will be documented in the Deliverable 6.4.

Next, we also plan a stake-holder meeting where the prepared video presentation will help to introduce the developed results. Some of the results are developed with an intention to make them publicly available. Therefore we will also use the prepared videos to explain these results to the users. To this end, we will review the videos not to reveal confidential information of any consortium partner.

5 References

[1] Cichonski, Paul & Millar, Tom & Grance, Tim & Scarfone, Karen. (2012). NIST Special Publication 800-61 Revision 2, Computer Security Incident Handling Guide. http://dx.doi.org/10.6028/NIST.SP.800-61r2

[2] MISP - Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing, available online: https://www.misp-project.org

[3] Playbook sharing object for MISP, available online: https://github.com/MISP/misp-objects/pull/324#issue-1009464958

[4] Machine learning object for MISP, available online: https://gitlab.fit.fraunhofer.de/sappan/sappan/-/tree/master/sappan-misp/collaborative-ml/data_models/classification-model