Advanced Cyber-Threat Intelligence, Detection, and Mitigation Platform for a Trusted Internet of Things
Grant Agreement: 786698

D6.3 - CYBER-TRUST Network Tools

Work Package 6: Advanced cyber-attack detection and mitigation

Document Dissemination Level

P	Public	X
CO	Confidential, only for members of the Consortium (including the Commission Services)	

Document Due Date: 31/01/2020
Document Submission Date: 31/01/2020

Co-funded by the Horizon 2020 Framework Programme of the European Union

Document Information

Deliverable number:	**D6.3**
Deliverable title:	CYBER-TRUST network tools
Deliverable version:	V1.2
Work Package number:	WP6
Work Package title:	Advanced cyber-attack detection and mitigation
Due Date of delivery:	31/01/2020
Actual date of delivery:	31/01/2020
Dissemination level:	Public
Editor(s):	Stefano Cuomo (Mathema) Simone Naldini (Mathema)
Contributor(s):	Gueltoum bendiab, Stavros Shiaeles (CSCAN) Nicholas Kolokotronis, Konstantinos Ntemos (UOP)
Reviewer(s):	Evangelos Sfakianakis (OTE) George Pikramenos (UOP)
Project name:	Advanced Cyber-Threat Intelligence, Detection, and Mitigation Platform for a Trusted Internet of Things
Project Acronym	Cyber-Trust
Project starting date:	1/5/2018
Project duration:	36 months
Rights:	Cyber-Trust Consortium

Version History

Version	**Date**	**Beneficiary**	**Description**
V0.1	02/12/2019	Mathema, UOP	Table of contents and distribution of work
V0.2	25/01/2020	CSCAN	All the WP5 proactive components' have been described
V0.3	20/01/2020	Mathema, CSCAN	Consolidation of document
V1.0	28/01/2020	Mathema	Version ready for review
V1.2 Final	31/01/202	Mathema	Final Version

Acronyms

ACRONYM	EXPLANATION
A	Actor
AI	Artificial Intelligence
AMPQ	Advanced Message Queuing Protocol
APIs	Application Programming Interface
AS	Autonomous System
ASN	Autonomous System Number
BTC	Bitcoin
CEF	Common Event Format
CIDR	Classes Inter-Domain Routing
CPE	Common Platform Enumeration
CT	Cyber-Trust
CTI	Cyber-Threat Intelligence
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerabilities Scoring System
CWE	Common Weakness Enumeration
D	Deliverable
DB	Database
DLT	Distributed Ledger Technology
DNS	Domain Name System
DPIA	Data Protection Impact Assessment
EQL	Event Query Language
EU	European Union
eVDB	Enriched Vulnerability Database
FR	Functional Requirement
GUI	Graphic User Interface
ID	Identification
IDS	Intrusion Detection System
iRC	iIRS Client
iRE	iIRS Decision-making Engine
iRG	iIRS Attack Graph Generator
iIRS	Intelligent Intrusion Response
IoC	Indicator of Compromise
IoT	Internet of Things
IP	Internet Protocol
ISP	Internet Service Provider
JSON	JavaScript Object Notation
LEA	Law Enforcement Agency
LCPD	Local Conditional Probability Distribution
M	Month
MAC Address	Media Access Control Address
MISP	Malware Information Sharing Platform
NFR	Non-Functional Requirement
NIDS	Network Intrusion Detection System
NVD	National Vulnerability Database
OCR	Optical Character Recognition
OS	Operative System
PHP	Hypertext Preprocessor

REST	Representational State Transfer
RGB	Red Green Blue
SGA	Smart Gateway Agent
SHO	Smart Home Owner
SQL	Structured Query Language
SSH	Secure SHell
SSL	Secure Socket Layers
STIX	Structured Threat Information Expression
T	Task
TLS	Trust Level Score
TMS	Trust Management Service
UI	User Interface
URL	Uniform Resource Locator
VERIS	Vocabulary for Event Recording and Incident Sharing
VLAN	Virtual Local Area Network
VM	Virtual Machine
WP	Work Package
XML	Extensible Markup Language

Executive summary

This report is a contractual deliverable within the Horizon 2020 Project Cyber-Trust: Advanced Cyber-Threat Intelligence, Detection, and Mitigation Platform for a Trusted Internet of Things.

The report provides a detailed description of the results of the task T6.3 “Cyber-Trust Network Tools” related to the implementation of the tools of work package and according to the platform architecture as described in D4.4.

In particular a description of tools targeting at (critical) network infrastructures, with a focus on botnet detection and mitigation, is given. The tools are aimed to minimise the damage through anomaly detection, deep packet inspection, and protocol analysis techniques.

The report gives technical information about the following components, Network repository, Cyber defence Service, Smart Gateway Module and Intelligent Intrusion Response System. A chapter describes a visual representation of data already integrated in the Cyber-Trust User Interfaces.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

1.1 Purpose of the document

This document is a technical description of the network tools for the detection and remediation of advanced device attacks. The report provides a detailed description of the results of the task T6.3 “Cyber-Trust Network Tools” related to the implementation of the tools of work package and according to the platform architecture as described in D4.4.
In particular, a description of tools targeting at (critical) network infrastructures, with a focus on botnet detection and mitigation, is given. The tools are aimed to minimise the damage through anomaly detection, deep packet inspection, and protocol analysis techniques.

1.2 Relations to other activities in the project

The document is strongly linked to task 5.1, and to deliverable 5.3 (CYBER-TRUST proactive technology tools) which describes the techniques and tools developed to combat threats on the network.

1.3 Structure of the document

The document is structured in order to describe the components related to the tools for network attack and mitigation used in the Cyber-Trust platform.
In particular:

1. Network repository
2. Cyber defence Service
3. Smart Gateway Module
4. Intelligent Intrusion Response System

For each of these components, information will be given about:

- A general overview
- Functionality Coverage
- Application Architecture
- Application programming interfaces
- Technology stack
- Physical Architecture
- User Interface (where relevant)

2. Network Repository

2.1 Overview / objectives

The network architecture and assets repository (A16) is a set of tools that are used to collect, maintain and store information on a network's architecture including the topology and the security defenses that are deployed at the network level, relevant device profile information, assets and their values, etc. This component serves three main objectives within the ISP platform including the network monitoring and scanning, network forensic evidence collection and data repositories, and risk characterization. The network monitoring is done by capturing the outgoing/incoming traffic, ensuring thatvulnerability assessment is performed, mapping hosts' network Interfaces and providing network Routing table and VLANs. While the network forensic evidence gathering is automatically performed under specific conditions. For example, with the identification of an attack.

The data collection is restricted to only relevant network metadata that can be used as digital forensic evidence in the court of law. This information is stored in the "**ForensicEvidence**" database (off-chain), while the related hash values, timestamps and information regarding the owner of the data will be stored in the DLT (on-chain). Further, the Network architecture and assets repository component provides necessary information to the trust management service in order to calculate the trust level. Figure 1 gives a view of the input and output data in the network architecture and assets repository component.

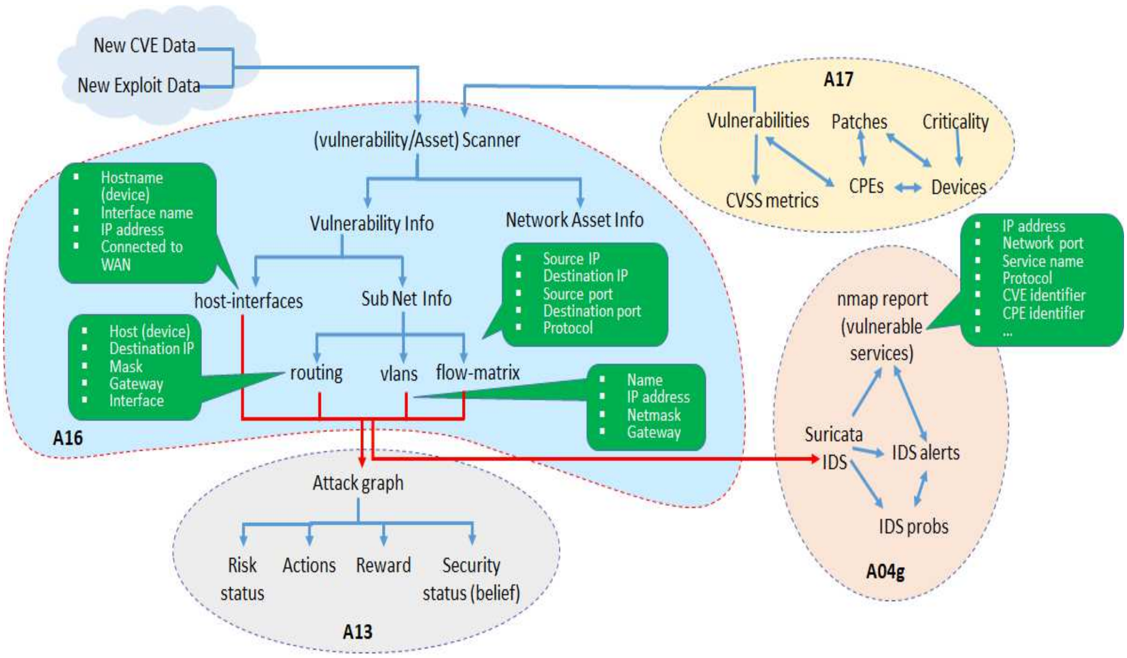

Figure 2-1: Data view of the network architecture and assets repository

2.2 Functionality coverage

The functional coverage of the Network architecture and assets repository includes the functional and non-functional requirements of the component, which are collected from the end-user questionnaires, the state-of-the-art review of relative technological advanced tools. Other requirements were derived from the operational requirements of this component in order to be connected with other API of apps, protocols tools, etc. These requirements fed the architectural requirements and structured some architectural specifications of the platform.

2.2.1 Related requirements

The main functional requirements of the Cyber Trust platform related to the Network architecture and assets repository are presented in Table 2-1, while the non-functional requirements are demonstrated in Table 2-2.

Table 2-1: Functional requirements related to the Network architecture and assets repository

ID. FR Requirement	Description	Related use cases
FR9	Based on the asset value of each device, users will perform mitigation actions both automatically and manually (hybrid). This functional requirement is related to the functional requirement FR65 of D2.4.	UCG-18-01
FR24	User will perform mitigation actions based on the asset value of each device. The mitigation action could be performed automatically or manually (hybrid). This functional requirement is related to the functional requirement FR65 of D2.4.	UCG-18-01
FR26	The user should have the capability to define the time period after which Cyber-Trust will perform mitigation actions automatically.	UCG-18-01, UCG-18-03, UCG-18-06
FR28	In case of an automatic Mitigation Action, the end-user through Cyber-Trust platform would have the capability to visualise the additional information about the mitigation actions taken through the visualisation portal component (A01).	UCG-04-03
FR54	Cyber-Trust will create a network map of the respective infrastructure.	System
FR55	The end user will be able to characterize each asset on the network and the respective value.	UCG-04-02, UCG-04-03

Table 2-2: Non-Functional requirements related to the Network architecture and assets repository.

ID. Non-FR Requirement	Description	Related use cases
NFR2	Respect of EU legal framework in the automatic or manual collection of data from the IoT devices that may contain forensic evidence that can be used for analysis and in the court of law. The collection should be done in specific conditions, such as abnormal behavior, low score of devices, etc.	UCG-14-01
NFR4	The Cyber-Trust platform will be capable of sending data (that might contain forensic evidences) exported from a device via a file. (This functionality will be available for LEAs). Based on this functionality, the Cyber-Trust platform will be able to generate the Hash value of the file to be sent.	UCG-11-01, UCG-12-02
NFR5	Based on the functionality described in FR49 (from D2.4): The file must be encrypted	UCG-14-04
NFR6	Based on the functionality described in FR49 (from D2.4): The transmission of the data must be secure in order to ensure the integrity.	UCG-14-04
NFR32	Continuous monitoring and scanning of the device's critical OS files. Information such as a communication protocol, open ports, running services, installed firmware etc., constitute correlation parameters for the detection of possible vulnerabilities specific to each device.	UCG-09-01

2.2.2 Related use cases

The use case Scenarios (Presented in D2.3) related to the network architecture and assets repository are presented in Table 2-3, while, the relationships between each requirement and those use cases are demonstrated in Table 2-1 and Table 2-2.

Table 2-3: Use cases related to the Network architecture and assets repository.

Use case reference ID	Description

UCG-04-02	**Characterize asset's importance**: The user prioritizes the attributes of the devices and their services according to her preferences.
UCG-04-03	**Define mitigation actions' impact:** The security officer quantifies the impact that the various mitigation actions on the availability of the network resources to trusted devices. This information, along with the smart home user's preferences is used to define the utility function which is required by the iIRS.
UCG-09-01	**Monitor device critical OS files / vulnerabilities:** The critical OS files/directories are continuously monitored. They will be scanned for open ports and running processes and obtained information is synced with the central backend database. In case any attempt is made in modifying the state of the device, backend services are triggered to check the status of vulnerability.
UCG-11-01	**Gather device forensic evidence:** The process of gathering evidence, especially in IoT environments differs based on the device, it's storage capabilities and software. Therefore, the collection and storage of forensic evidences (e.g. device log files, timestamps etc.) will be depicted from the cyber-trust registered devices.
UCG-13-02	**Compute device trust level**: The trust module collects all needed information and recomputes the trust level of the device
UCG-14-01	**Update device critical OS files / vulnerabilities:** In case a legitimate update is performed on the OS, firmware or any device critical files, key device parameters are recalculated and updated to the central database. Then the process of detecting vulnerabilities is also performed.
UCG-14-04	**Manage available patch databases**: For each type of registered device, the Patch database contains information related to the latest security fixes of the firmware as well as the relevant binaries. Hash information is also securely stored to ensure the integrity of the vanilla patch versions.
UCG-16-02	**Discover Network**: The exploitation of the Cyber-Trust device profiles conjoined with location information to allow for support to visualization capabilities, either via dynamic (flow) or static (GID) graphs.
UCG-18-01	**Apply Mitigation Policy on Device:** The decision taken at the network level is applied at the device level.
UCG-18-03	**Apply network security defense rule:** Network security combines multiple rules and layers of defenses at the edge and in the network. These include access control, application security, Intrusion prevention system, firewall and many more**.**
UCG-18-06	**Define applicable mitigation actions**: The Security officer consults the cyber-attack graphical security model, which contains the system's security conditions and the available exploits, as well as their relations, and defines the mitigation actions which are at the iIRS disposal.

2.3 Technology update

An extended search was performed to identify whether new versions or implementations had emerged since the last review of the relevant areas. Since no interesting progress was reported during this period, the current tools used by the network architecture and the asset repository remain those indicated in D6.1.

2.4 Application architecture

The following diagram (Figure 2) illustrates the conceptual view of the network architecture and assets repository component. Its architecture is designed to allow enquiring information on a network's architecture (including the topology and the security defenses deployed therein), assets and their values, and vulnerability assessment. As mentioned before, this component ensures the monitoring and scanning at the network level as well as at the device level, which may involve the collection of relevant information in the "**ForensicEvidence**" database, in case any attempt to modify the state of the device or network. While the hash values of the collected information, time stamps regarding the data and information regarding the owner of the data will be stored in the DLT (on-chain).

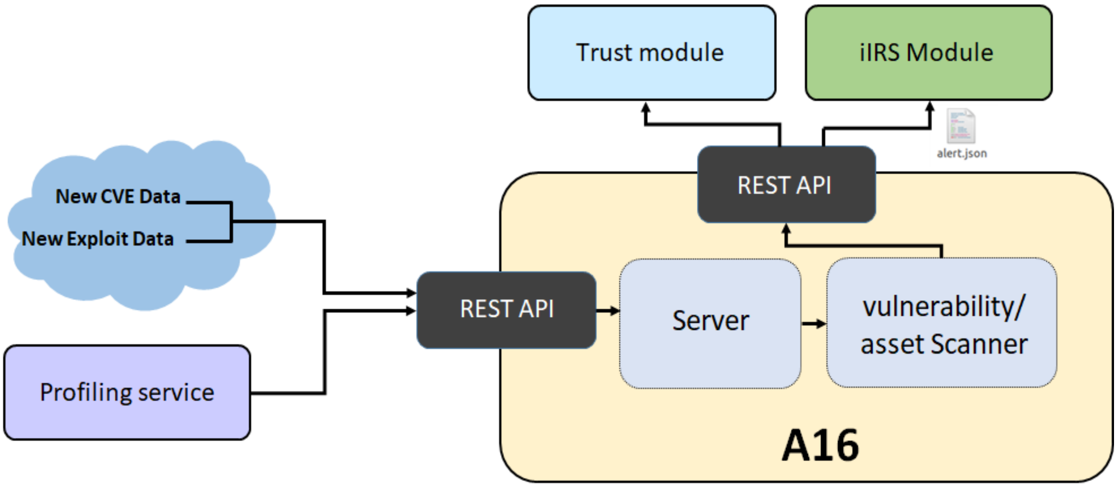

Figure 2-2: High level view of the network and assets repository architecture.

2.5 Application programming interfaces

The network architecture and assets repository component is written mostly in Linux utilities and Python with the interface provided by Django-REST-Framework and direct JSON outputs in a folder. It consists of three main parts:

1. The core which is responsible for actual capturing
2. The REST interface which calls the same REST APIs from the convenient Web interface.
3. The direct output is given to a predefined folder in a form of JSON provided by the REST APIs.

The JSON output folder has four main JSON files; “list_flow_matrix.json”, “list_hosts_interface.json”, “list_nmap_vulnerabilities.json” and “list_routing_tables.json”. It also has extra JSON, “alert.json”, which reports changes in any other JSON file in the folder. The description of those JSON outputs is provided in Table 2-4. All the APIs of the network architecture and assets repository have two common input parameters; the IP address or subnet and the “cached_or_live” parameter, which can either have the “cached” or “live” values.

- live – The output will be computed on demand and the internal cache would be updated.
- cached – It will return cached output. So, for some long-running computation, this can be quicker

Table 2-4:Network architecture and assets repository APIs

API	Description	Example response (JSON file)
List-flow-matrix	This API provides information about the network traffic, where each object in the list corresponds to flow that happened. It has the following output parameters: 1. **Source**: refers to the source of address of this flow. 2. **Destination:** refers to the destination address of this flow. 3. **Source_port:** refers to the source port for this connection.	[{ "source": "10.0.10.105", "destination": "178.248.236.150", "source_port": "33932", "destination_port": "443", "protocol": "TCP" }, { "source": "178.248.236.150", "destination": "10.0.10.105", "source_port": "443", "destination_port": "33932", "protocol": "TCP" }, { "source": "10.0.10.1",

	4. **Destination_port**: refers to the destination port for this connection. 5. **Protocol:** refers to the Transport layer protocol used (e.g. TCP, UDP).	"destination": "10.0.10.105", "source_port": "80", "destination_port": "38248", "protocol": "TCP" }]

List-hosts-interfaces	This API provides information about the hosts and their interface. It has the following output parameters:
	1. **Hostname:** refers to the hostname of this interface.
	2. **Interface_name**: refers to the name of this specific interface.
	3. **IP_address**: refers to the host IP address.
	4. **connected_to_wan** : this parameter shows if this interface is connected to the Internet or not.
		[
		{
		"hostname": "host-00:0c:29:c5:f1:ce",
		"interface_name": "fa0",
		"ip_address": "10.0.10.105",
		"connected_to_wan": true
		}
		]

List-routing-tables	This API fetches all the routing tables in the network. It has the following output parameters:
	1. **Hostname:** refers to hostname where this routing table resides.
	2. **Destination**: refers to the destination of this routing entry.
	3. Mask: refers to the mask of this routing entry.
	4. Gateway: refers to the gateway of this routing entry.
	5. **Interface**: refers to the interface of this routing entry.
		[
		{
		"hostname": "pfsense",
		"destination": "default",
		"mask": "",
		"gateway": "10.0.20.1",
		"interface": "em1"
		},
		{
		"hostname": "pfsense",
		"destination": "10.0.10.0/24",
		"mask": "",
		"gateway": "link#1",
		"interface": "em0"
		},
		{
		"hostname": "pfsense",
		"destination": "10.0.10.1",
		"mask": "",
		"gateway": "link#1",
		"interface": "lo0"
		}
		]

List-vlans	This API provides the list of all virtual LANs on the current subnet. It has the following output parameters: 1. **Name:** refers to the name of this VLAN.	[{ "name": "VLAN00", "address": "10.0.10.0", "netmask": "24", "gateway": "link#1" },

	2. **Address**: refers to the IP Address of CIDR block.
	3. **Netmask:** refers to the Net mask for this CIDR block.
	4. **Gateway: refers to the** default gateway for this VLAN
		{
		"name": "VLAN01",
		"address": "10.0.20.0",
		"netmask": "24",
		"gateway": "link#2"
		}
		]

List-nmap-vulners	This API fetch the latest exploit from Vulners for given IP address. The output format for this API can be found at: NMAP-Python library and Vulners. **1- Python-nmap** (https://github.com/vulnersCom/nmap-vulners): python-nmap is a python library which helps in using nmap port scanner. It allows to easily manipulate nmap scan results and will be a perfect tool for systems administrators who want to automatize scanning task and reports. It also supports nmap script outputs. **2- Nmap-Vulners** (https://github.com/vulnersCom/nmap-vulners): It uses some well-known service to provide information about vulnerabilities.	

Alerts	This API reports changes in any of the previous JSON files. It has two main parameters:
	1. **Date**: refers to the date when the changes is made.
	2. **API**: The API affected by the change.
		[
		{
		"date": 2019-07-09 20:10:48 .232480,
		"API": list-routing-tables
		},
		{
		"date": 2019-05-11 05:06:43 .081209,
		"API": list-hosts-interfaces
		}
		]

2.6 Technology Stack

As illustrated in Figure 3, the network architecture and assets repository component is written mostly in Linux utilities and Python 3.7 with the interface provided by Django-REST-Framework and RESTful APIs.

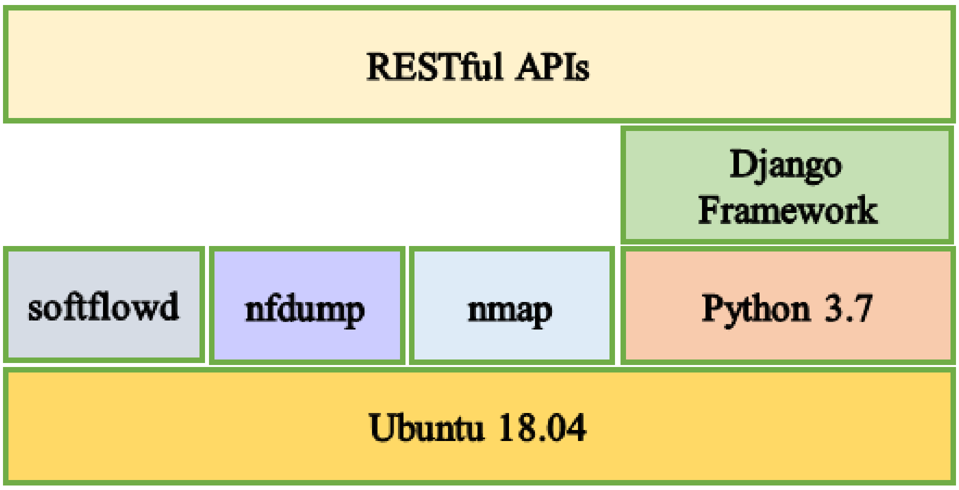

Figure 2-3: Technology stack of the network and assets repository architecture.

The following table provides a detailed description of the tools used in the development of the network and assets repository architecture.

Table 2-5: Technology stack of the network architecture and assets repository.

Tool	Version	Description
Django-REST-Framework	Any	Django is a Python-based free and open-source web framework, which follows the model-template-view architectural pattern. It is used for providing visual representation of the network architecture and assets repository outputs.
SOFTFLOWD	Any	SOFTFLOWD[1] is a flow-based network traffic analyser capable of Cisco NetFlow data export. It is used to semi-statefully tracks traffic flows recorded by listening on the network interface or by reading a packet capture file. These flows may be reported via NetFlow™ to a collecting host or summarized within SOFTFLOWD itself.
NFDUMP	version: 1.6.13	The NFDUMP[2] tool is used to collect and process NetFlow data on the command line. They are part of the NfSen project including nfcapd, NFDUMP, NFPROFILE, NFCLEAN.pl and FT2NFDUMP.
Python	Python 3, version 3.6. or greater.	The network architecture and assets repository component is written mostly in Python 3. The following are the dependences for the developed component based on Ubuntu 18.04 environment, all of which are open source packages/libraries: ares0.5, certif2019.3.9, chardet3.0.4, colorama0.4.1, coreapi2.3.3, coreschema0.0.4, Django2.1.7, django-log-file-viewer0.9, django-logtailer1.0.1, django-request-logging0.6.7, django-rest-swagger2.2.0, djangorestframework3.9.2, djangorestframework-stubs0.4.2, idna2.8, itypes1.1.0, Jinja22.10, MarkupSafe1.1.1, numpy1.16.2, oauthlib3.0.1, openapi-codec1.3.2, pandas0.24.2, pyfiglet0.8. post1, PySocks1.6.8, python-dateutil2.8.0, python-nmap0.6.1, pytz2018.9, requests2.21.0, requests-oauthlib1.2.0, simplejson3.16.0, six1.12.0, termcolor1.1.0, tqdm4.31.1, tweepy3.7.0, twitter1.18.0, twython3.7.0, uritemplate3.0.0, urllib31.24.1, vulners1.4.6.
nmap	Version 7 or greater.	The nmap[3] tool is used for network discovery and security auditing. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics.
RESTful APIs	Any	The network architecture and assets repository output four main JSON files.

2.7 Physical architecture

In terms of physical architecture, the following deployment options are used:

- The A16 (network and assets repository architecture) component is deployed as two VMs.

[1] https://www.mindrot.org/projects/softflowd/
[2] http://nfdump.sourceforge.net/
[3] https://nmap.org/

 - The first VM is used for running the open-source software "pfsense[4]" as a router based on FreeBSD. pfSense is a popular, state-of-the-art, easy-to-configure open source firewall, VPN, and router solution. With over 1 million active installations across the world.
 - Due to the libraries and resources limitations, the network and assets repository architecture [A16] core component is deployed in a second VM that is running ubuntu 18.04.2 TLS.
- The network and assets repository architecture is deployed as a Python application with the interface provided by Django-REST-Framework and RESTful APIs.
- The network and assets repository architecture use the SSH (Secure Shell) to enable secure access to the pfsense VM and retrieve the required information.

2.8 User Interface

The network assets repository architecture has its own UI, independent of Cyber-Trust's platform. This interface is provided by Django-REST-Framework and accessible through the ROOT_URL (on same machine http://127.0.0.1) (Figure 4). All the REST APIs are accessible from this interface by selecting the type of the API to be called (Figure 5).

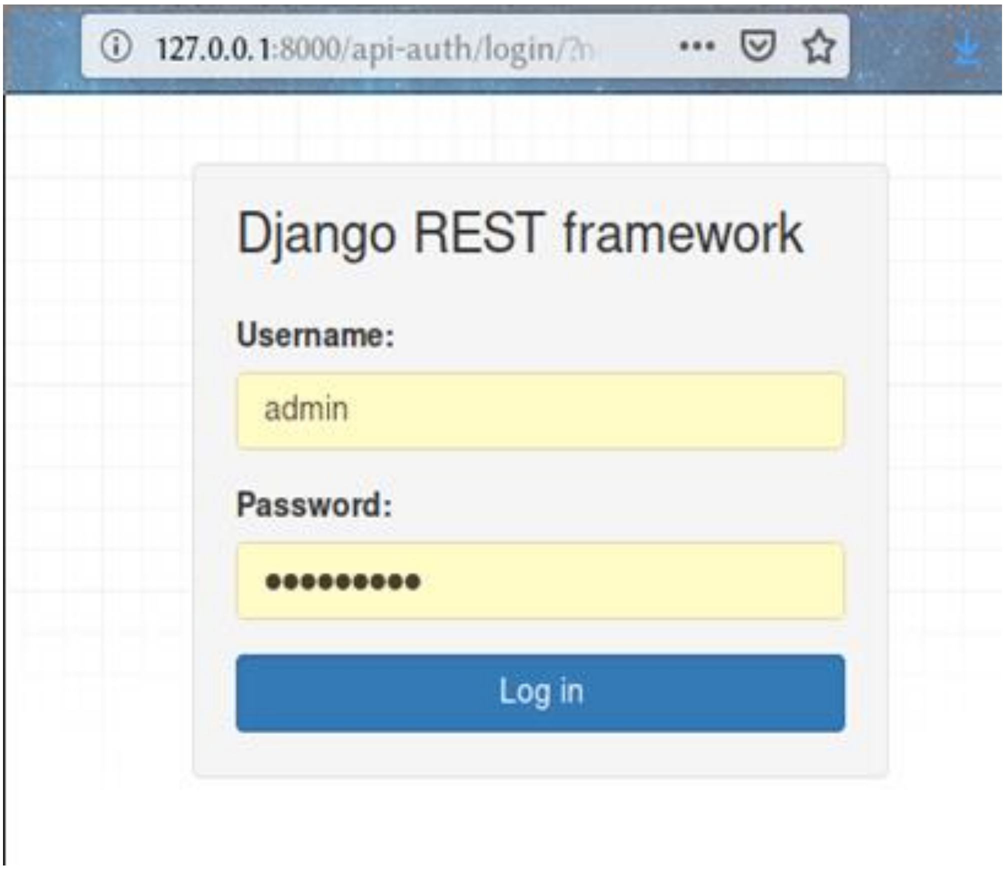

Figure 2-4: Login interface to the network assets repository architecture.

[4] https://docs.netgate.com/pfsense/en/latest/

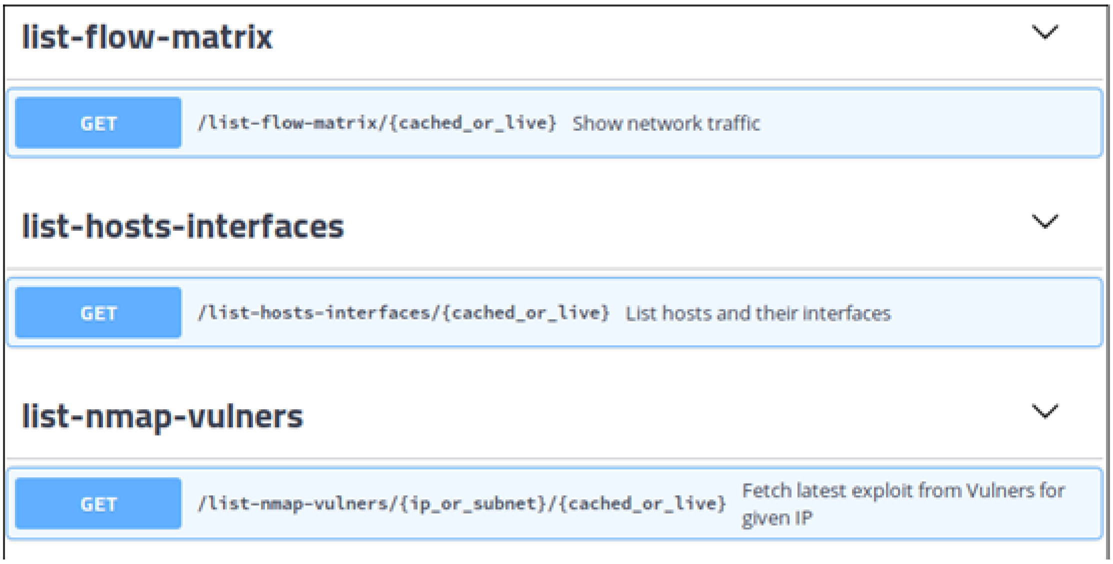

Figure 2-5: the network assets repository architecture REST APIs interface.

3. Cyber-Defense Service

3.1 Overview / objectives

The Cyber-Defense service (A04) main objective is the detection and mitigation of potential Cyber-attacks on the networks level as well as at the device levels. The device monitoring is done at the gateway which is running the Network Intrusion Detection System (NIDS). At this level, a signature-based technique is used to check for anomalies on a predefined database of signatures from known malicious threats. Similarly, at the network level, the intrusion detection system is used to monitor the incoming and outgoing network traffic. The IDS compare the incoming traffic (i.e. packets signatures) against the predefined malware signatures to identify possible attacks. The signature-based technique is very accurate at detecting known attacks, but largely ineffective in detecting unknown and new versions of attacks for which there exist no signatures. Therefore, an AI anomaly detection module is integrated to the NIDS, at the Cyber-Defense service level, as a second layer in the detection and mitigation process. Figure 3-1 presents the activity diagram of the detection and mitigation process at the Cyber-Defense service level.

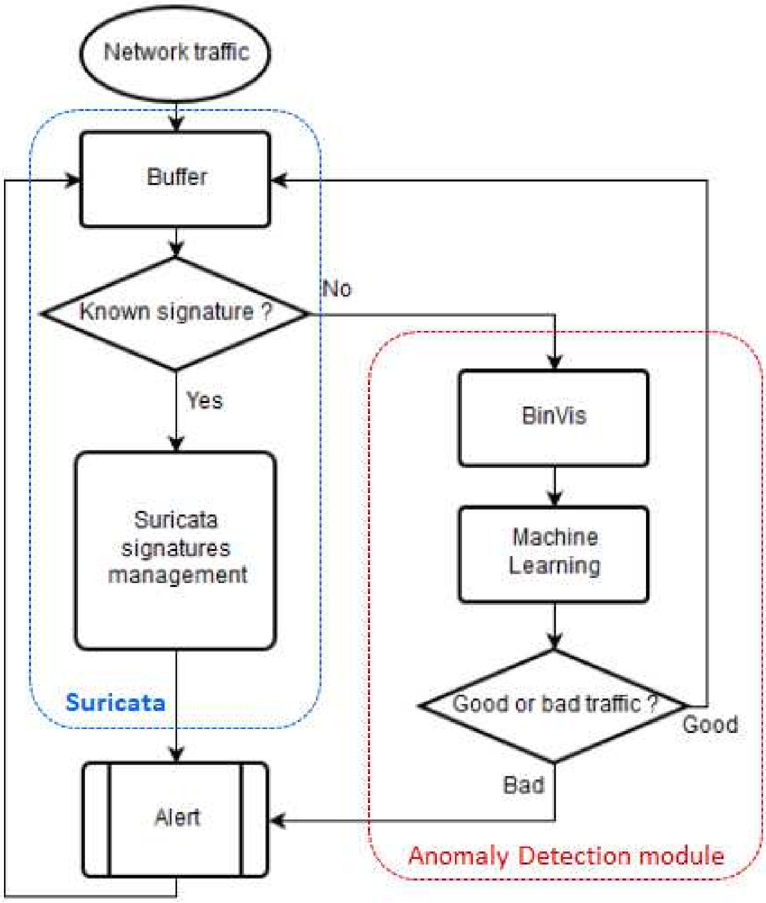

Figure 3-1: Activity diagram of the detection and mitigation process at the Cyber-Defense service level.

3.2 Functionality coverage

The functional coverage of the Cyber-Defense service includes the functional and non-functional requirements of the component, which are collected from the end-user questionnaires, the state-of-the-art review of relative technological advanced tools. Other requirements were derived from the operational requirements of this component in order to be connected with other ??? of apps protocols tools, etc. These requirements fed the architectural requirements and structured some architectural specifications of the platform.

3.2.1 Related requirements

The main functional requirements of the Cyber Trust platform related to the Cyber-Defense service are presented in Table 3-1, while the non-functional requirements are demonstrated in Table 3-2.

Table 3-1: Functional requirements related to the Cyber-Defense service.

ID. FR Requirement	Description	Related use cases
FR29	In accordance with the relevant legal framework, information regarding the firmware of the device will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-01
FR30	In accordance with the relevant legal framework, critical software files of the device will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-01
FR31	In accordance with the relevant legal framework, information regarding relevant configurations of the device will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-01
FR32	In accordance with the relevant legal framework, Audit logs of the device will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-01
FR33	In accordance with the relevant legal framework, critical OS files of the device will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law	UCG-11-01
FR34	In accordance with the relevant legal framework, information depicting if the latest patches have been installed on the device will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-01
FR35	In accordance with the relevant legal framework, network log files will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law	UCG-11-02
FR36	In accordance with the relevant legal framework, typical volumes of packet transfer of the network will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law	UCG-11-02
FR37	In accordance with the relevant legal framework, typical protocols of the network will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law	UCG-11-02
FR38	In accordance with the relevant legal framework, suspicious connections and services of the network will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-02
FR39	In accordance with the relevant legal framework, traffic analysis of the network will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law	UCG-11-02
FR56	Cyber-Trust will automatically mitigate abnormal behaviour based on the network map, the characterization of the assets, the impact of the attack as well as the impact of the mitigation actions. If the mitigation action has severe impact on certain dimensions of assets that score high value Cyber-Trust will propose possible actions, but it will not implement it automatically.	UCG-04-03, UCG-06-07
FR64	Once a new patch is stored in the respective repository an alert/notification will be send, through the UI, to the end user of the respective device.	UCG-07-01

FR65	The platform will provide functionality so as to enable automatic update for devices when new patch/firmware is out.	UCG-07-01
FR79	The platform will be capable to restoring a device to a healthy state (i.e. A device that has been previously compromised is detected to have its health restored) if one or more vulnerabilities have been mitigated for this device.	UCG-17-01
FR85	For each connected device the Connection rates of the device should be provided.	UCG-11-01
FR86	For each connected device the MAC address of the device should be provided.	UCG-11-01

Table 3-2: Non-Functional requirements related to the Cyber-Defense service.

ID. Non-FR Requirement	Description	Related use cases
NFR7	Users will take the final decisions about appropriate actions based on the recommendation of Cyber-Trust Platform.	UCG-04-03, UCG-06-07, UCG-18-06
NFR8	Regarding the collection, storing and processing of material that may contain forensic evidences: Data must be kept safe to avoid Integrity Violations and be backed up for any unfortunate eventualities. An alternative failover plan will be offered by Cyber-Trust in case of not having an applicable mitigation action.	UCG-18-01, UCG-18-06
NFR11	A fully configured failover plan will be proposed by Cyber-Trust to the users with a set of predefined options.	UCG-18-01, UCG-18-06
NFR12	In case of important alert, the end-user will have the capability to update the system/device manually.	UCG-14-01
NFR14	The Cyber-Trust platform must be secure to protect collected data.	UCG-04-02, UCG-04-03
NFR31	Packet information regarding source and destination IP address, source and destination ports, flags, header length and checksum will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-08-01
NFR32	Continuous monitoring of the device's critical OS files to check for anomalies	UCG-09-01

3.2.2 Related use cases

The use case Scenarios (Presented in D2.3) related to the Cyber-Defense service are presented in Table 3-3, while, the relationships between each requirement and those use case scenarios are demonstrated in Table 3-1 and Table 3-2.

Table 3-3: Use cases related to the Cyber-Defense service.

Use case reference ID	Description
UCG-04-02	**Characterize asset's importance**: The user prioritizes the attributes of the devices and their services according to her preferences.
UCG-04-03	**Define mitigation actions' impact:** The security officer quantifies the impact that the various mitigation actions on the availability of the network resources to trusted devices. This information, along with the smart home user's preferences is used to define the utility function which is required by the iIRS.

UCG-06-07	**Communicate iIRS actions to the security officer**: The iIRS after computing the optimal defense action taken by the cyber defense service, it informs the Security officer.
UCG-07-01	**Check device patching status:** Intelligence regarding the latest versions of firmware is stored in the Cyber-Trust backend system. Periodically, the installed firmware and software on monitored devices [A03] is checked and when outdated the end user is notified.
UCG-08-01	**Monitor device at the gateway (network traffic filtering)**: The getaway is running network intrusion detection systems (NIDS) to check for signatures and anomalies based on signatures.
UCG-09-01	**Monitor device critical OS files / vulnerabilities:** The critical OS files/directories are continuously monitored. They will be scanned for open ports and running processes and obtained information is synced with the central backend database. In case any attempt is made in modifying the state of the device, backend services are triggered to check the status of vulnerability.
UCG-11-01	**Gather device forensic evidence:** The process of gathering evidence, especially in IoT environment differs based on the device, it's storage capabilities and software. Therefore, the collection and storage of forensic evidences (e.g. device log files, timestamps etc.) will be depicted from the cyber-trust registered devices.
UCG-11-02	**Gather network forensic evidence**: The process (automatic) and conditions (e.g. with the identification of an attack) under which the Cyber-Trust will start collecting relevant network data in order to be used as digital forensic evidences in the court of law as well as the collection mechanisms/techniques (e.g. DPI).
UCG-14-01	**Update device critical OS files / vulnerabilities:** In case a legitimate update is performed on the OS, firmware or any device critical files, key device parameters are recalculated and updated to the central database. Then the process of detecting vulnerabilities is also performed.
UCG-17-01	**Remediate Device:** Restoring a device to a healthy state. Remediation takes place once the detection of an attack is confirmed meaning that either abnormal device behaviour is detected. The remediation process involved the isolation of affected files and their recovery to a previous healthy state or the notification of the end user about advised actions.
UCG-18-01	**Apply Mitigation Policy on Device:** The decision taken at the network level is applied at the device level.
UCG-18-06	**Define applicable mitigation actions**: The Security officer consults the cyber-attack graphical security model, which contains the system's security conditions and the available exploits, as well as their relations, and defines the mitigation actions which are at the iIRS disposal.

3.3 Technology update

The current technology used by the Cyber-Defense service remain the same as indicated in D6.1. No interesting progress was reported during this period in the area of detection and mitigation techniques.

3.4 Application architecture

The following diagram illustrates the architecture of the Cyber-Defense Service. The architecture follows a loosely coupled approach allowing for scalability and extensibility where necessary. As shown in Figure 7, the Cyber-Defense service receives data as input from three other components in the platform including the Network architecture and assets Repository (A16), the Smart Gateway iIRS app (A13) and the smart gateway agent (A04g).

- Form the Network architecture and assets Repository service, it receives information on the network architecture and assets including the flow-matrix (source IP, destination IP, source port, destination port, protocol, etc.), routing (host-name, destination IP, mask, gateway, interface, etc.), VLANs (name, IP address, netmask, gateway, etc.) and information about the device (Hostname, interface name, IP address, connected to wan, etc.).
- From the Smart Gateway iIRS app (A13) it receives as input the trust metric and CT signatures

- A04 has also access to the real-time security alerts generated by A04g

As output, A04 generates real-time security alerts that will be used by the Smart Gateway iIRS app for selecting the response actions in real-time to mitigate and block the progression of a cyber-attack on the network by refusing communication requests, shutting down running services, etc. This component does not have direct interactions with end-users.

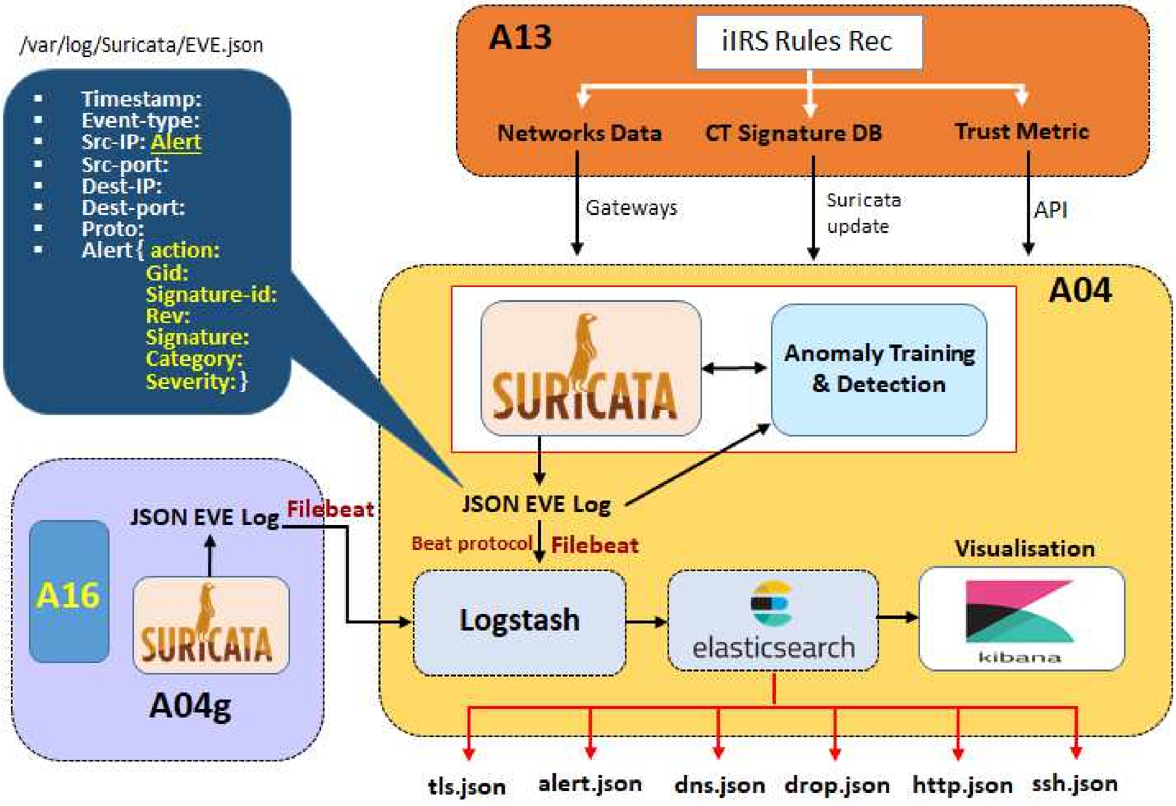

Figure 3-2: High-level design of the Cyber-Defense service.

3.4.1 Suricata IDS

Suricata[5] is a high-performance Network IDS, IPS and Network Security Monitoring engine. It was developed in 2010 by the Open Information Security Foundation (OISF)[6] in an attempt to meet the requirements of modern infrastructures. Suricata Is used by the Cyber-defense service as a signature-based Network IDS that inspects the network traffic using a powerful and extensive rules and signature language, and a powerful Lua[7] scripting support for detection of complex threats. It implements a complete signature language to match on known threats, policy violations and malicious behaviour. Moreover, it can log HTTP requests, log and store TLS certificates, extract files from flows and store them to disk. The full pcap capture support allows easy analysis [1, 2]. All this makes Suricata a powerful engine for the detection and mitigation of potential Cyber-attacks [2].

This NIDS introduced multi-threading to help speed up the network traffic analysis and overcome the computational limitations of the single-threaded architecture [2]. It also provides intrusion prevention (NIPS) and network security monitoring. Unlike Snort, Suricata provides offline analysis of PCAP files by using a PCAP recorder [3]. It also provides excellent deep packet inspection and pattern matching which makes it more efficient for threat and attack detection [4]. Many studies assume that Suricata is a powerful adversary to Snort and thus they are often compared with each other [1, 2].

[5] https://suricata-ids.org/
[6] https://oisf.net/
[7] https://www.lua.org/

3.4.2 Anomaly Detection module

As illustrated in Figure 6, the anomaly detection module receives traffic that was analyzed by the signature-based module (Suricata) and considered as normal traffic for further analysis. This module can dynamically adapt to distinguish between legitimate and malware traffic, especially unknown and new malware. The alerts http events, DNS events, TLS events and file info are generated through a json file. Its main modules are the Binary visualisation with Binvis module and the machine learning module.

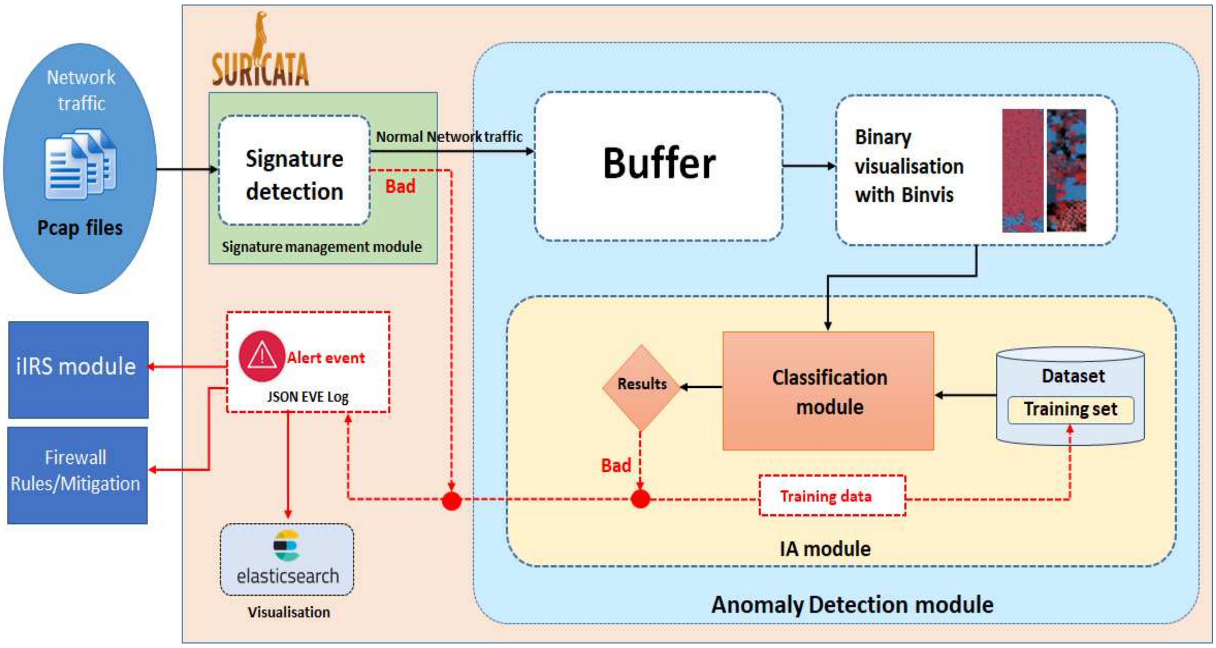

Figure 3-3: High level design of the anomaly detection module

1. **Binary visualisation with Binvis module**: This module uses the Binvis[8] tool to store the pcap files in a binary format and then convert them to RGB images. The RGB values mapping is done by comparing each byte value in the pcap file to their equivalent in the ASCII table, according to the predefined colour scheme [5]. Binvis divided the different ASCII bytes into four groups of colour, where printable ASCII bytes are assigned a blue colour, control bytes are assigned a green colour and extended ASCII bytes are assigned a red colour. Black (0x00) and white (0xFF) colour respectively represent null and (non-breaking) spaces [5]. After that, the clustering algorithm Hilbert space-filling curve [6] is used to find the coordinates of each byte colour in the output RGB image. The size of the output RGB image is 784 (1024*256) bytes. Figure 7 shows Binvis images for normal and malware pcap files, which are created using the Hilbert space-filling curve clustering algorithm.

[8] https://github.com/cortesi/scurve/blob/master/binvis

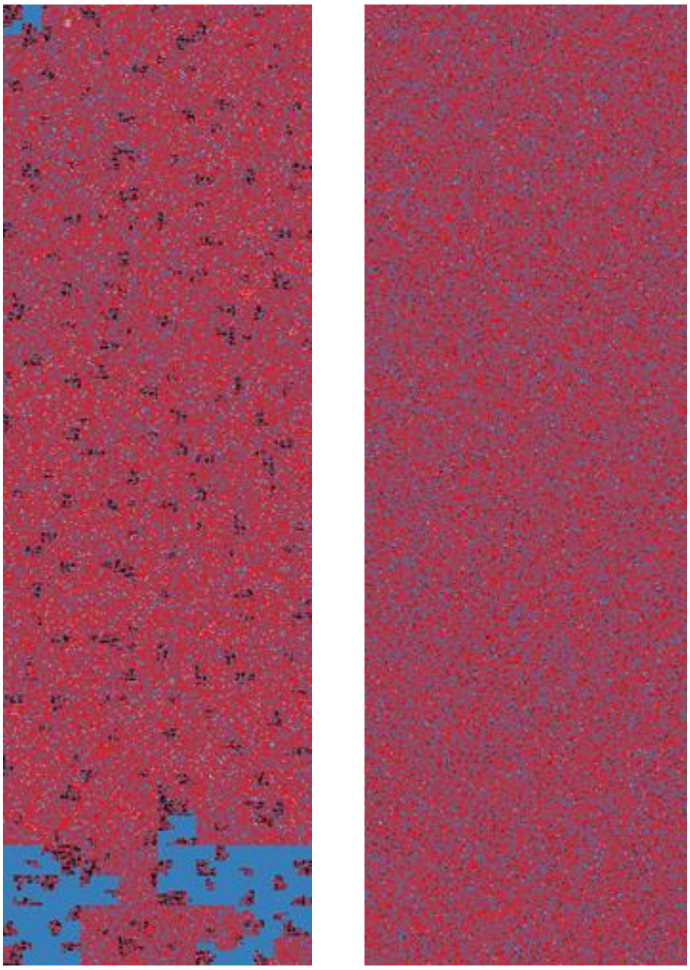

Figure 3-4: Output Binvis images for (1) malware pcap files of Lucky Ransomware traffic

2. **Machine learning module**: This module analyses the produced Binvis images against its in-depth training dataset to perform the classification of the incoming traffic as normal or malware. Detected malware traffic will be used to continuously train the classifier in order to enhance their detection accuracy. Several learning algorithms were used to tests and evaluate the success of the detection method and determine its accuracy. Table 3-4 presents the overall results of the tests carried out on a dataset consists of 600 pcap files of normal and malware traffic that are collected from different network traffic sources.

Table 3-4: Tests results of the anomaly detection module

Machine learning algorithms	Accuracy	Recall	F1-Score
MobileNet	91.32%	91.03%	91.35%
Self-Organizing Incremental Neural Networks (SOINN)	91.70%	/	/
Residual Neural Network (ResNet50)	91.18%	/	/
AdaBoost	85.00%	85.00%	85.00%
K-Nearest Neighbours (KNN)	61.00%	54.00%	50.00%
C-Supported Vector Classification (SVC)	71.00%	70.00%	60.00%
Gaussian Process Classifier (GPC)	26.00%	51.00%	34.00%
Decision Tree	72.00%	72.00%	72.00%
Random Forest	77.00%	75.00%	75.00%
Multi-Layer Perceptron	24.00%	49.00%	32.00%
Gaussian Naive Bayes (GaussianNB)	83.00%	81.00%	81.00%

The Cyber-Defense service also involves the decision making at defending the Cyber-Trust system and limiting the impact of the imposed threat (e.g. hindering the impostors in accessing the victim devices in the network); the detection and mitigation service depending on the findings of the analysis determines the most appropriate mitigation policy to be taken at the device level.

3.4.3 Deep Packet Inspection

The cyber defense service uses Deep Packet inspection for evidence collection. With the identification of an attack, network forensic evidence gathering is automatically done under specific conditions, where data collection is restricted to only relevant network metadata that can be used as digital forensic evidence in the court of law. As shown in Figure 8, Raw data is stored in the "**ForensicEvidence**" database (off-chain), while the related has values, timestamps and information regarding the owner of the data will be stored in the DLT (on-chain).

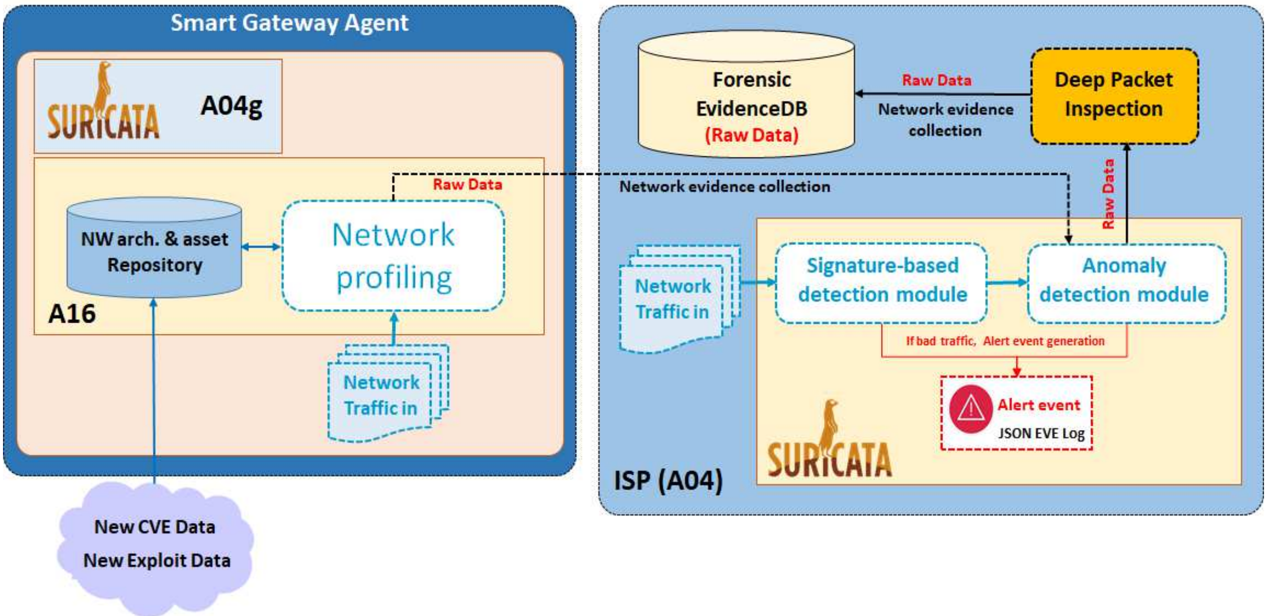

Figure 3-5: Network evidence collection with Deep Packet Inspection

3.5 Application programming interfaces

Suricata IDS output alerts, http events, DNS events, TLS events and file info into one "**eve.json**" file, which will be later processed by other tools like Logstash. EVE can output to multiple methods such as "syslog", "unix_dgram", "unix_stream" and "redis". It is also possible to have multiple 'EVE' instances. For example, alerts and drops could go into one instance of the eve.json file, while http, DNS and TLS go into another instance of this file. Moreover, each log can be handled completely separately. All the JSON log types generated by Suricata share a common structure, which has a field "**event_type**" to indicate the log type and more additional fields if Suricata is processing a pcap file.

```
{
        "timestamp":"2009-11-24T21:27:09.534255",
        "event_type":"TYPE",

        ...tuple...,

       "TYPE":
              {
                 ... type specific content ...
              }
```



```
    }
```

The following table describes the main output types that could be generated by Suricata in the eve.json file.

Table 3-5: Output types generated by Suricata in the “eve.json” file

JSON log type	Description	Example of the TYPE section

Alerts	Alerts are event records for rule matches. If the incoming traffic signature matches with any one of the existing signatures (rules), it is considered as malicious and an alert JSON log will be generated in the eve.json file.
	The action parameter is set to “allowed” unless a rule used the “drop” action and Suricata is in IPS mode, or when the rule used the “reject” action.
		"alert":
		{
		"action": "allowed",
		"gid": 1,
		"signature_id": 1,
		"rev": 1,
		"app_proto": "http",
		"signature": "HTTP body talking about corruption"
		"severity": 3,
		"source": {
		"ip": "192.168.43.32",
		"port": 36292
		},
		"target": {
		"ip": "179.60.192.3",
		"port": 80
		},

http events	It mainly provides information about http events in the network, which may include the hostname this HTTP event is attributed to, the URL at the hostname that was accessed, the user-agent of the software that was used and the type of data returned (ex: application/x-gzip, “cookie”).	"http":
		{
		"http_port": 1337,
		"hostname": "www.digip.org",
		"url" :"\/jansson\/releases\/jansson-2.6.tar.gz",
		"http_user_agent": "<User-Agent>",
		"http_content_type": "application\/x-gzip"
		}

DNS events	It mainly provides information about a DNS event, which may include the type of the DNS message (answer or query), DNS logging version in use, DNS answer flag, in hexadecimal (ex: 8180), the Resource Record Name (ex: a domain name), Resource Record Type (ex: A, AAAA, NS, PTR), Resource Data (ex. IP that domain name resolves to) and the Time-To-Live for this resource record.	Example of a DNS query for the IPv4 address of “twitter.com” (resource record type ‘A’):
		
		"dns":
		{
		"type": "query",
		"id": 16000,
		"rrname": "twitter.com",
		"rrtype":"A"
		}

TLS events	It provides information about TLS events in the network that may include the subject and issuer fields from the TLS certificate, TLS session was resumed via a session ID or not, the serial number and fingerprint of the TLS certificate, SSL/TLS version used, etc.	"tls": { “subject”: “C=US, ST=Californai, L=Mountain View, O=Google Inc, CN=*.google.com", "issuer": "C=US, O=Google Inc, CN=Google Internet Authority G2” "session_resumed": true, "serial": "0C:00:99:B7:D7:54:C9:F6:77

		: 26: 31:7E:BA:EA:7C:1C", "fingerprint": "8f:51:12:06:a0:cc:4e:cd:e8:a3:8b :38:f8:87:59:e5:af:95:ca:cd", "version": "TLS 1.2" }

3.6 Technology Stack

The following table provides a detailed description of the tools used in the development of the Cyber-defense service.

Table 3-6: Technology stack of the Cyber-Defense service.

Tool	Version	Description
Elastic search RESTful API	Any	The Elastic search[9] is used for storing and sharing all log files provided by Suricata IDSs at the Smart Gateway Agent (SGA) level and the ISP level. It provides scalable search, has near real-time search, and supports multitenancy.
Logstash	Any	Logstash [10]is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favorite "stash." It is used by the Cyber-defense service as a data processing pipeline that retrieves data from a multitude of sources simultaneously (Suricata EVE log files), transforms it, and then sends it to Elastic search.
Python	Version 6.3 or greater	The anomaly detection module is written in Python. This module deals with unknown attacks, for which there is no signatures.
Lua scripting language	Any	Suricata IDS support rules (or signatures) written in the embeddable scripting language Lua. This language is mainly used to read, adjust and create new rules (signatures).
Kibana	Any	The visualisation tool Kibana[11] is used with Elastic search to provide a 2D visualisation of the results in the Suricata log file.
Filebeat	Any	Filebeat is used to forward and centralise the collection of the eve log files generated by Suricata IDSs at the Smart Gateway Agent (SGA) level in the ISP level.
EveBox	Any	Evebox is a web frontend that is used to display the Suricata alerts after being processed by Elastic search.

3.7 Physical architecture

In terms of physical architecture, the Cyber defense service is deployed as a single VM, running the Suricata IDS and the anomaly detection module based on Debian GNU/Linux 10.2. Suricata is installed and configured as Intrusion Detection System (IDS) with a specific set of rules (or signatures). Elasticsearch, Logstash, Kibana, EveBox and Filebeat are installed in the same VM and configured to handle the EVE JSON logs produced by Suricata.

The integration of the anomaly detection module with Suricata is achieved by passing 10MB PCAP files retained within SELKS '/data/nsm/' directory through the anomaly detection module, detection is then performed on the Binvis output images and when the network traffic is determined to be 'malware' the stored network information for that block is outputted to eve.json using the documented json format in **(https://suricata.readthedocs.io/en/suricata-5.0.0/output/eve/eve-json-format.html).** Malware detected

[9] https://www.elastic.co/
[10] https://www.elastic.co/guide/en/logstash/current/getting-started-with-logstash.html
[11] https://www.elastic.co/guide/en/kibana/current/introduction.html

images are then outputted to the anomaly detection module's log file in '/var/log/malwaresquid/' for further processing and analysis, the alert data is then read by EveBox and treated the same way as Suricata's alerts.

3.8 User Interface

In addition to the visualisation interface provided by the Cyber-Trust's platform, the cyber defense service outputs could be also visualized by using Kibana with Elastic search, independent of Cyber-Trust's platform. With this interface, users and officers have the possibility to get details about all the events (stats, flow, DNS, fileinfo, http, tls, dhcp, alert, etc.) generated by the cyber the Cyber-Trust's platform as well as general statistics about the alerts detected along with timeline graphs of generated events over time (e.g. alerts count, Alerts categories, Alerts details, Alerts, locations, Alerts top 10 signatures, Alerts top 10 source & destination ports, Alerts top source & destination IPs, alerts by information type (SSH alert, SMTP alert, TLS alert, SMB alert, DNS alerts, other Alerts), etc. more details will be provided in deliverable D6.4. Figures 11 and 12 illustrates Kibana dashboards, which show information from the Suricata IDS EVE log file.

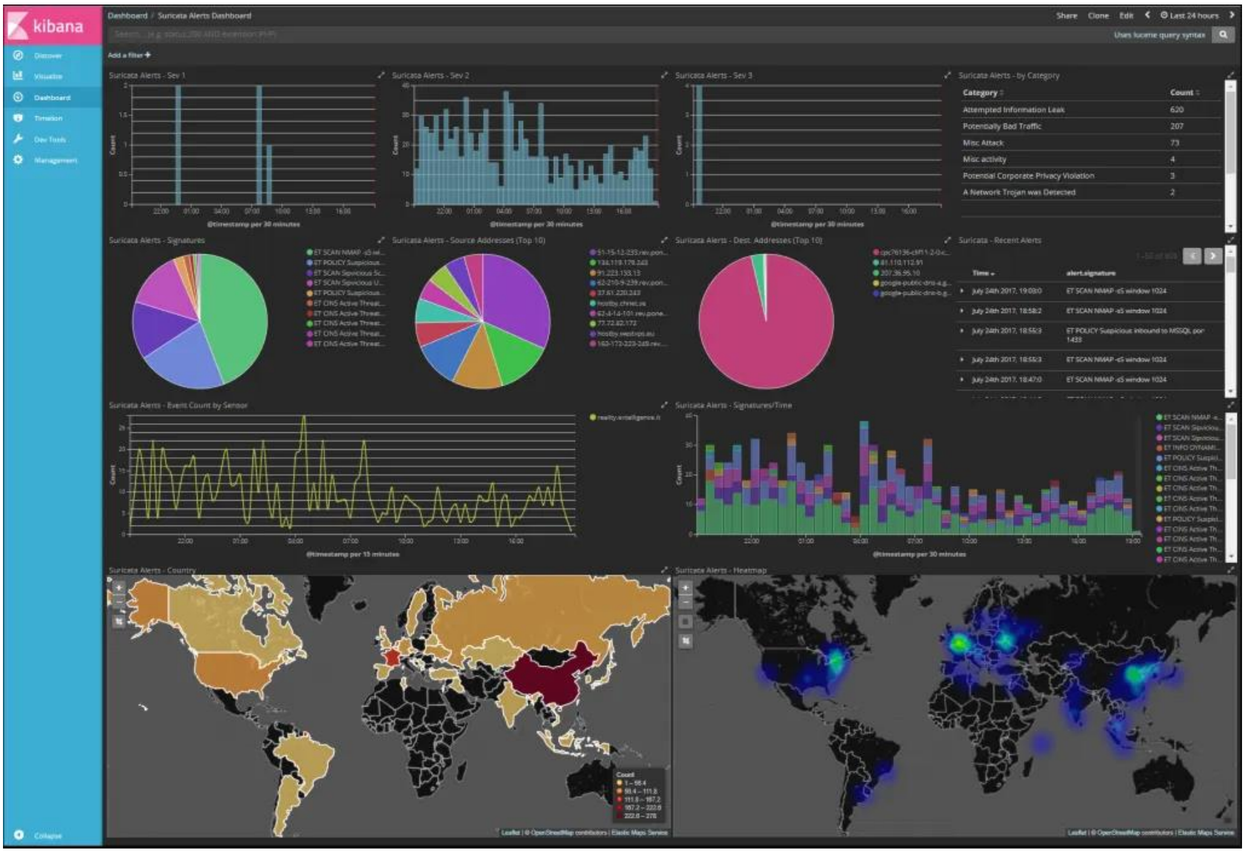

Figure 3-6: Visualisation of information from the Suricata IDS EVE log file through Kibana dashboard.

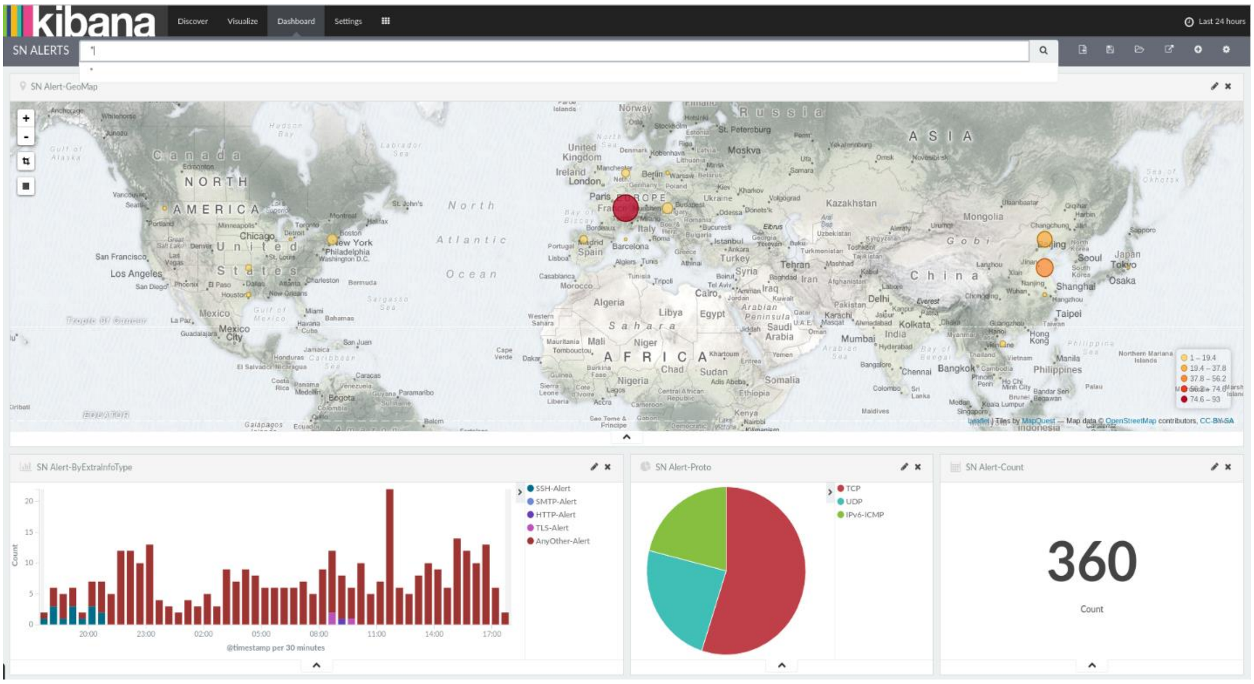

Figure 3-7:Kibana Alert dashboard.

4. Smart Gateway Module

4.1 Overview / objectives

The Smart Gateway Agent (SGA) main objectives are the monitoring of network traffic, and the computation and application of mitigation policies and remediation actions. SGA ensures the following functionalities:

- Continuous Monitoring at the gateway (network traffic filtering) to check for signatures and anomalies based on the signature and report of the attack is issued to the responsible security officer, capture and classify network packets (DPI) and traffic profiling for each device.
- Discovery & Intelligence of ongoing attacks on the network level.
- Ensure mitigation and communication by defining mitigation policies and parameters, computing mitigation policies and attack surface, computing and delivering of trust and risk assessments. It communicates the received intrusion detection alerts to the user.
- Enforce Network Mitigation by applying then mitigation and remediation actions as necessary when an attack is detected.
- Store of trust information in Database to be later retrieved and processed or returned as a response to queries. The SGA also includes the component A13 that realizes the intelligent intrusion response system(iIRS), which is presented in the next section.

4.2 Functionality coverage

The functional coverage of the Smart Gateway Agent (SGA) includes the functional and non-functional requirements of the components A03g, A04g, A05g, A08g and A11, which are collected from the end-user questionnaires, the state-of-the-art review of relative technological advanced tools. Other requirements were derived from the operational requirements of this component in order to be connected with other of apps protocols tools, etc. These requirements fed the architectural requirements and structured some architectural specifications of the platform.

4.2.1 Related requirements

The main functional requirements of the Cyber Trust platform related to the Smart Gateway Agent (SGA) are presented in Table 4-1, while the non-functional requirements are demonstrated in Table 4-2.

Table 4-1: Functional requirements related to the Smart Gateway Agent (SGA).

ID. FR Requirement	Description	Related use cases
FR1	For each connected device the Type of Device must be provided	UCG-04-01
FR2	For each connected device the Vulnerabilities of the device must be provided	UCG-04-01
FR3	For each connected device the Open Ports of the device must be provided	UCG-04-01
FR9	Every device connected to the Cyber-Trust platform has visual representation of the Trust level (scoring) before the identification of abnormal behavior (e.g. cyber-attack)	UCG-05-07, UCG-05-05
FR10	Every device connected to the Cyber-Trust platform has visual representation of the Trust level (scoring) during abnormal behaviour (e.g. cyber-attack).	UCG-05-07, UCG-05-05
FR11	Every device connected to the Cyber-Trust platform has visual representation of the Trust level (scoring) after the mitigation of any abnormal behavior (e.g. cyber-attack).	UCG-05-07, UCG-05-05
FR19	In case of vulnerabilities detected on a device, the Cyber-Trust platform will inform users by alert messages.	UCG-06-01, UCG-06-02, UCG-16-03, UCG-18-04

FR20	In case of vulnerabilities detected on the device, the Cyber-Trust platform will inform users, by alert icons.	UCG-06-01, UCG-06-02, UCG-16-03, UCG-18-04
FR21	The user will be informed for the importance of the alert based on the overall Score of the device (it will be derived based on the abnormal behaviour, detected vulnerabilities etc.)	UCG-06-01, UCG-06-02, UCG-13-01, UCG-16-03
FR29	Information regarding the firmware of the device will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-01
FR30	Critical software files of the device will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-01
FR31	Information regarding relevant configurations of the device will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-01
FR32	Audit logs of the device will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-01
FR33	Critical OS files of the device will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-01
FR34	Information depicting if the latest patches have been installed of the device will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-01
FR35	Network log files will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-02
FR36	Typical volumes of packet transfer of the network will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-02
FR37	Typical protocols of the network will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law	UCG-11-02
FR38	Suspicious connections and services of the network will be collected, stored and analysed as forensic evidence that can be used for analysis and in the court of law.	UCG-11-02
FR39	Traffic analysis of the network will be collected, stored and analyses as forensic evidence that can be used for analysis and in the court of law.	UCG-11-02
FR43	Deep Packet Inspection: MAC address of source packet	UCG-08-02
FR44	Deep Packet Inspection: MAC address of destination packet	UCG-08-02
FR45	Deep Packet Inspection: IP of source address	UCG-08-02
FR46	Deep Packet Inspection: Information derived from capturing and analyzing the payload	UCG-08-02
FR47	Deep Packet Inspection: Destination port of the packet	UCG-08-02
FR48	Deep Packet Inspection: The packets will be characterized and classified into various categories such as benign, anomaly, suspected	UCG-08-02
FR56	Cyber-Trust will automatically mitigate abnormal behaviour based on the network map, the characterization of the assets, the impact of the attack as well as the impact of the mitigation actions. If the mitigation action has severe impact on certain dimensions of assets that score high value Cyber-Trust will propose possible actions, but it will not implement it automatically.	UCG-04-03, UCG-06-07
FR61	Statistics regarding the network traffic will be visualised in the monitoring service (2D).	UCG-06-03

FR85	For each connected device the connection rates of the device should be provided.	UCG-11-01
FR86	For each connected device the MAC address of the device should be provided.	UCG-11-01
FR91	Deep Packet Inspection: Number of hops from source to destination (TTL – Time To Live mechanism).	UCG-08-02

Table 4-2: Non-Functional requirements related to the Smart Gateway Agent (SGA).

ID. Non-FR Requirement	Description	Related use cases
NFR31	Packet information regarding source and destination IP address, source and destination ports, flags, header length and checksum will be collected.	UCG-08-01
NFR32	Continuous monitoring of the device's critical OS files.	UCG-09-01
NFR40	iIRS will use the alerts raised by the IDS in order to update the belief it possesses over the system security state.	UCG-15-04
NFR41	The monitoring service will be comparing the hash information from the device's critical files and the values provided by the device information management system, continuously.	UCG-16-01
NFR48	Regarding the collection, storing and processing of material that may contain forensic evidences: ISO 27001 controls (A.5 Security policy, A.6 Organization of information security, A.7 Asset management, A.8 Human resources security, A.9 Physical and environmental security, A.10 Communications and operations management, A.11 Access control, A.12 Information systems acquisition, development and maintenance, A.13 Information security incident management, A.14 Business continuity management, A.15 Compliance,).	

4.2.2 Related use cases

The use case Scenarios (Presented in D2.3) related to the smart Gateway Agent are presented in Table 4-3, while, the relationships between each requirement and those use case scenarios are demonstrated in Table 4-1 and Table 4-2.

Table 4-3: Use cases related to the Smart Gateway Agent (SGA).

Use case reference ID	Description
UCG-04-01	**Private IoT Device Profile generation:** Implementation of one-way cryptographic hash functions to pseudonymise data and secure multi-party communications for anonymous data distribution.
UCG-04-02	**Characterize asset's importance**: The user prioritizes the attributes of the devices and their services according to her preferences.
UCG-04-03	**Define mitigation actions' impact:** The security officer quantifies the impact that the various mitigation actions have on the availability of the network resources to trusted devices (e.g. by refusing communication requests, shutting down running services, etc.). This information, along with the smart home user's preferences is used to define the utility function which is required by the iIRS.
UCG-05-05	**Visualize device vulnerability levels**: The level of vulnerability of the devices targeted is presented in 2D OMCP.
UCG-05-07	**Visualize device trust level: T**he level of trust of the devices targeted is presented in 2D OMCP

UCG-06-01	**Raise alert for security officer:** Raise an alert to intelligent UI user when for example, there is a mitigation policy to be applied as a response to a threat originated from IoT device.
UCG-06-02	**Raise alert for device owner**: Raise an alert to user when for example, there is a threat is detected in one of the owners' s IoT devices.
UCG-06-03	**Establish baseline traffic statistics**: Traffic statistics of the network will be displayed on the 2D Monitoring and Control Panel (MCP) in order to allow ISP operator in being aware about the situation
UCG-06-07	**Communicate iIRS actions to the security officer**: The iIRS after computing the optimal defense action (Cyber Defense service), it informs the Security officer.
UCG-08-01	**Monitor device at gateway (network traffic filtering)**: The getaway is running network intrusion detection systems (NIDS) to check for signatures and anomalies based on signature.
UCG-08-02	**Capture and classify network packets (DPI)**: The system automatically captures and classify the packets based on their contents; this can be achieved using Deep Packet Inspection (DPI) approach. The DPI will characterize the packets into various categories such as benign, anomaly, suspected.
UCG-09-01	**Monitor device critical OS files / vulnerabilities:** The critical OS files/directories are continuously monitored. They will be scanned for open ports and running processes and obtained information is synced with the central backend database. In case any attempt is made in modifying the state of the device, backend services are triggered to check the status of vulnerability.
UCG-11-01	**Gather device forensic evidence:** The process of gathering evidence, especially in IoT environment differs based on the device, it's storage capabilities and software. Therefore, the collection and storage of forensic evidences (e.g. device log files, timestamps etc.) will be depicted from the cyber-trust registered devices.
UCG-11-02	**Gather network forensic evidence**: The process (automatic) and conditions (e.g. with the identification of an attack) under which the Cyber-Trust will start collecting relevant network data [A03] in order to be used as digital forensic evidences in in the court of law as well as the collection mechanisms/techniques (e.g. DPI).
UCG-15-04	**Compute a belief on current security status**: The iIRS uses the alerts provided by the intrusion detection system to update the belief it has about the system security state. The belief is a probability distribution over the possible security states, which are comprised of the system security conditions. These conditions denote system attributes (e.g. active services (and the associated vulnerabilities), network connectivity, trust relationships between hosts, and attacker privileges on hosts) and represent attacker's capabilities.
UCG-16-01	**Determine device firmware and software through remote detection:** A backend service runs between the device and central device profile database to identify device's firmware and software thereby building and updating the central database with all firmware. As a result, the system can be able to generate metrics such as vulnerability and trust.
UCG-16-03	**Receive intrusion detection system(s) alerts**: In case of an attack discovery, or a false alarm event, the intrusion detection system is activated and generates alerts and informs the iIRS . The iIRS evaluates the generated alerts in order to infer the true system security state, by considering the possible mis detections and false alarms.
UCG-18-04	**Notify of device compromise**: Describe the procedure the system will use in case of a comptonization of a device.

4.3 Technology update

The current technology used by the Smart Gateway Agent remain the same as indicated in D6.1. No interesting progress was reported during this period.

4.4 Application architecture

The following diagram illustrates a high-level architecture of the smart Gateway Agent module. The architecture follows a loosely coupled approach allowing for scalability and extensibility where necessary. As shown in Figure 9, A04g receives data as input from other components in the platform including the Network architecture and assets Repository (A16), the Smart Gateway iIRS app (A13). A04g has also access to the real-time security alerts generated by A04g.

As output, A04g generates real-time security alerts that will be used by A13 for selecting the response actions in real-time to mitigate and block the progression of a cyber-attack on the network by refusing communication requests, shutting down running services, etc.

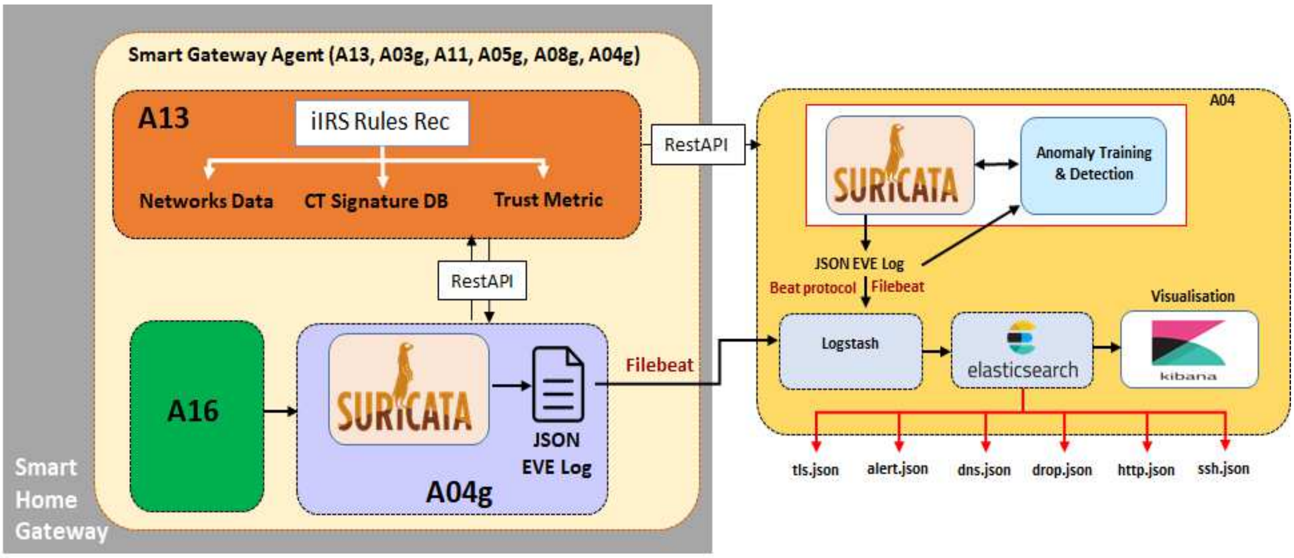

Figure 4-1: High-level Architecture of the Smart Gateway Agent

4.5 Application programming interfaces

The Smart Gateway Agent exposes a set of REST APIs for direct invocation by other Cyber-Trust modules. It additionally employs a loose coupling communication pattern, through the exchange of messages via the message bus; the respective messages consumed through the message bus will be elaborated on in the context of WP8. Table 4-4 shows the output generated by A04g component inside the smart gateway agent. Other APIs exposed by other components (A03g, A05g and A08g) are described in D5.3.

Table 4-4: Output generated by A0g

API	Description	Example response (JSON file)
A04g JSON EVE log	Suricata IDS in A04g output alerts, http events, DNS events, TLS events and file info into one "**eve.json**" file, which will be later processed by other tools like Logstash as described in section **3.5**.	

4.6 Technology Stack

The following table provides a detailed description of the tools used in the development of the Cyber-defense service.

Table 4-5: Technology stack of the Cyber-Defense service.

Tool	Version	Description
Suricata IDS	Any	A light version of Suricata Is used in the Smart Gateway Module as a signature-based Network IDS that inspects the smart home network traffic using a specific set of rules /signatures, and a powerful Lua scripting

		support for detection of complex threats. It implements a complete signature language to match on known threats, policy violations and malicious behaviour. Moreover, it can log HTTP requests, log and store TLS certificates, extract files from flows and store them to disk. The full pcap capture support allows easy analysis. All this makes Suricata a powerful engine for the detection and mitigation of potential Cyber-attacks.
Elastic search RESTful API	Any	The Elastic search is used for storing and sharing all log files provided by Suricata IDSs at the Smart Gateway Agent (SGA) level and the ISP level. It provides scalable search, has near real-time search, and supports multitenancy.
Logstash	Any	Logstash is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favorite "stash." It is used by the Cyber-defense service as a data processing pipeline that retrieves data from a multitude of sources simultaneously (Suricata EVE log files), transforms it, and then sends it to Elastic search.
Python	Version 6.3 or greater	In the development of the SGA, we used the following Python 3.6 libraries; SQL Alchemy, Flask, Numpy, Scipy, Sympy, Cython, Sklearn, Networkx, Pandas.
Lua scripting language	Any	Suricata IDS support rules (or signatures) written in the embeddable scripting language Lua. This language is mainly used to read, adjust and create new rules (signatures).
Filebeat	Any	Filebeat is used to forward and centralise the collection of the eve log files generated by Suricata IDSs at the Smart Gateway Agent (SGA) level in the ISP level.

4.7 Physical architecture

In terms of physical architecture, A single VM is used for running a light version of Suricata IDS (i.e. A04g component) with a specific set of rules (or signatures) that is related to the smart home devices. The VM is running ubuntu 18.04.2 LTS. This VM is used as an IDS for the pfsense router that is deployed in another VM due to libraries and resource limitations. A04g use the SSH (Secure Shell) to enable secure access to the pfsense VM and retrieve the required information.

5. Data representation integrated in the Cyber-Trust Interface

The modules responsible for the development of threat detection and mitigation tools provide the Cyber-Trust platform with a lot of important data, characterized by different formats and qualities, as well as at a level of information that they are able to transmit to the user.
The graphic management in the UI of this type of data requires the use and design of ad hoc forms of representation, in order not to make the decoding of this data too complex and not very immediate.

The choice of the types of graphic representation mainly concerns the type of data (e.g. qualitative or quantitative data, time series, punctual data, geographic data). For each typology the possible representation scenarios have been identified, always following the principle of optimizing the readability and usability of the data.

Furthermore, for the correct decoding of information and its use by users, it was necessary to make it possible to apply filters on these: the applicable filters include the selection of a time window (by setting the start and end date and time of selection). Figure 14 shows a data representation already integrated in the Cyber-Trust UI.

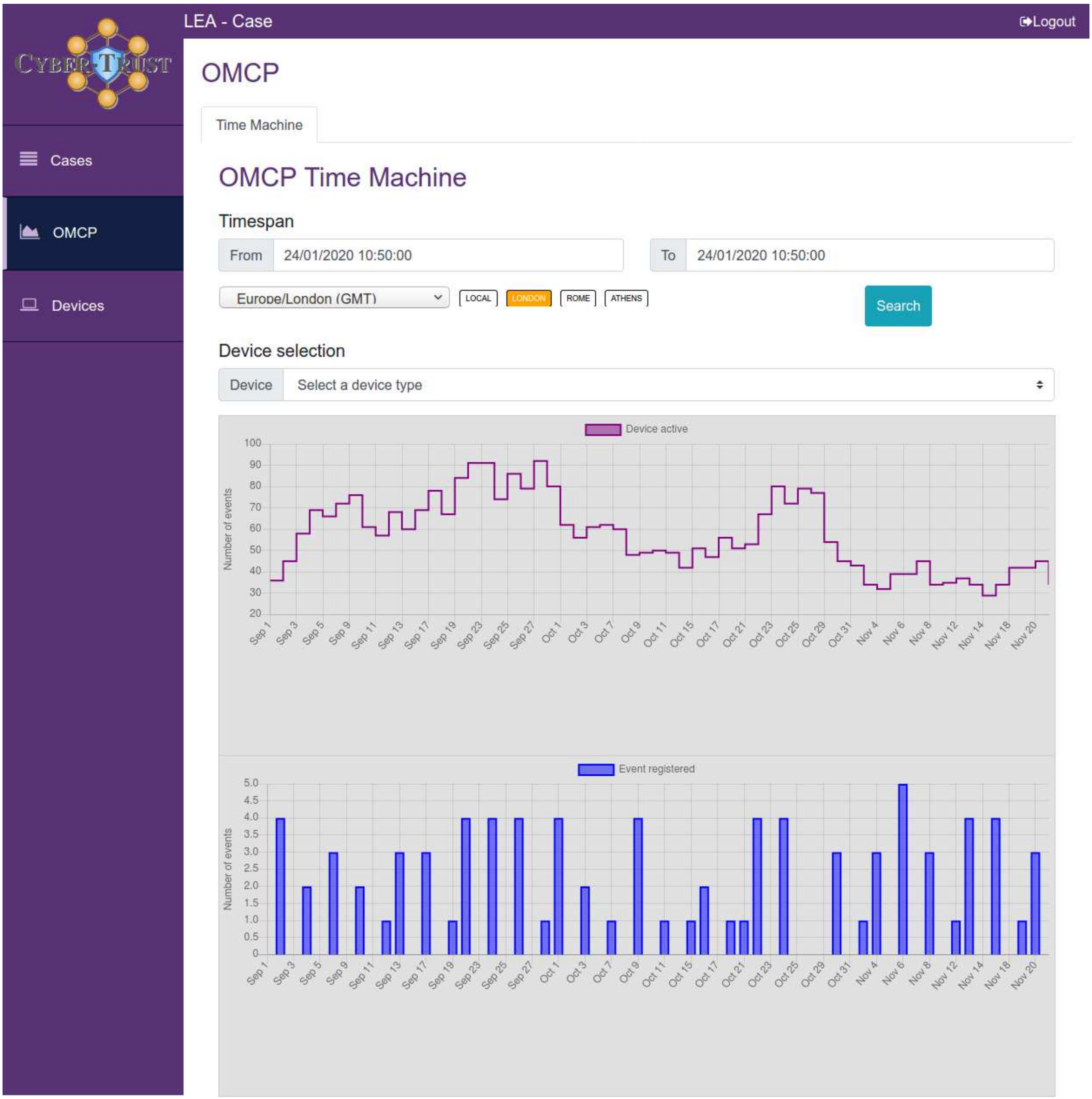

Figure 5-1: Data representation integrated in Cyber-Trust UI

6. Intelligent intrusion response system

6.1 Overview / objectives

In this chapter, we describe the operation and objectives of the *Intelligent Intrusion Response (iIRS, A13)* system. The objective of the iIRS is the real-time computation of mitigation actions that could be employed, with or without user interaction, against sophisticated network attacks. Our design of iIRS is based upon the model proposed in [KN1].

In doing so, the iIRS interacts with other system modules. More specifically, the iIRS receives as input the attack graphical security model, which represents the interconnection between exploits and the security attributes of both network devices and their provided services (the capabilities an attacker has and might acquire), to build the *state space*. To aid in optimization of the defense actions and to maximize the user's satisfaction, various attributes about the network devices and their provided services may be taken as input by the iIRS. Then, the iIRS computes the *response actions*, which are given as input to the IDS.

Here, we focus on the interaction between the iIRS and the *Intrusion Detection System (IDS, A04g)*. For more details regarding the interaction of iIRS with other system modules, the interested reader can refer to *Deliverable D5.3.*

6.2 Cooperation with IDS and Network Repository

The interactions between iIRS and IDS are of critical importance and are shown schematically in Fig. 10 The state space represents security conditions. Since we want to study the problem in a realistic setting, where erroneous alerts about the true system state can occur, we consider *partial observability* in the problem study, following the work presented in [KN1].

More specifically, the iIRS takes as input *security alerts* by the IDS, which do not necessarily represent the true underlying state, to update its belief about the security state. However, security alerts are subject to *false alarms* and *miss-detections*. This fact is taken into account by the iIRS, which utilizes the respective *probabilities of miss-detection and* false alarm in order to update the *belief vector* about the current system state. Then, the iIRS solves the *Partially Observable Markov Decision Process (POMDP)* to provide the optimal response actions to the IDS. These actions are in form of *firewall rules*.

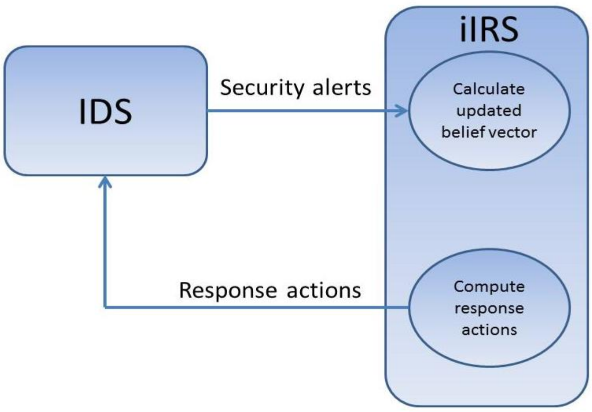

Figure 6-1: High-level illustration of cooperation between the iIRS and IDS

7. Conclusion

This document shows the current status of the Proactive Technology tools to be integrated into an operational environment. The integrated prototype will be piloted and tested in Task T8.3 and needed adaptations, further to the evaluation of the test will be performed.

8. References

1. Brumen, Boštjan, and Jernej Legvart. "Performance analysis of two open source intrusion detection systems." *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2016.
2. Isa, Fuad Mat, et al. "Comprehensive Performance Assessment on Open Source Intrusion Detection System." *Proceedings of the Third International Conference on Computing, Mathematics and Statistics (iCMS2017)*. Springer, Singapore, 2019.
3. Albin, Eugene, and Neil C. Rowe. "A realistic experimental comparison of the Suricata and Snort intrusion-detection systems." *2012 26th International Conference on Advanced Information Networking and Applications Workshops*. IEEE, 2012.
4. Svoboda, Jakub. "Network Traffic Analysis with Deep Packet Inspection Method." *Fac. Informatics Masaryk Univ., no. Master's Thesis* (2014).
5. Robert Shire, Stavros Shiaeles, Keltoum Bendiab, Bogdan Ghita, Nicholas Kolokotronis. "**Malware Squid: A Novel IoT Malware Traffic Analysis Framework Using Convolutional Neural Network and Binary Visualisation**". In: Internet of Things, Smart Spaces, and Next Generation Networks and Systems. Springer, Cham, 2019. p. 65-76. DOI https://doi.org/10.1007/978-3-030-30859-9_6.
6. Sagan, Hans. "Hilbert's space-filling curve." *Space-filling curves*. Springer, New York, NY, 1994. 9-30.
7. E. Miehling, M. Rasouli, and D. Teneketzis, "A POMDP approach to the dynamic defense of large-scale cyber networks," *IEEE Transactions on Information Forensics and Security*, vol. *13, no.* 10, pp. 2490-2505, 2018.

