SPHINX AI Honeypot v1

WP4 – SPHINX Toolkits

Version: 1.00

SPHINX

A Universal Cyber Security Toolkit for Health-Care Industry

Disclaimer

Copyright message

Document information

Grant Agreement Number	826183	Acronym	SPHINX

Full Title	A Universal Cyber Security Toolkit for Health-Care Industry			
Topic	SU-TDS-02-2018 Toolkit for assessing and reducing cyber risks in hospitals and care centres to protect privacy/data/infrastructures			
Funding scheme	RIA - Research and Innovation action			
Start Date	1st January 2019	**Duration**	36 months	
Project URL	http://sphinx-project.eu/			
EU Project Officer	Reza RAZAVI (CNECT/H/03)			
Project Coordinator	Dimitris Askounis, National Technical University of Athens - NTUA			
Deliverable	D4.4 – SPHINX AI Honeypot v1			
Work Package	WP4 – SPHINX Toolkits			
Date of Delivery	**Contractual**	M20	**Actual**	M20
Nature	R - Report	**Dissemination Level**	P - Public	
Lead Beneficiary	FINT			
Responsible Author	Dimitris Apostolakis	**Email**	dapostolakis@f-in.eu	
		Phone		
Reviewer(s):	SIMAVI, PDMFC			
Keywords	Honeypots, Artificial Intelligence, HW/SW co-design, Heterogeneous Computing, VMs			

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Document History

Version	Issue Date	Stage	Changes	Contributor
0.10	17/06/2020	Draft	ToC	Dimitris Apostolakis (FINT), Anargyros Sideris (FINT)
0.20	14/08/2020	Draft	1st consolidated version, ready for internal review	Dimitris Apostolakis (FINT), Anargyros Sideris (FINT)
0.30	25/08/2020	Draft	Internal Review 1	Radu Popescu (SIMAVI)
0.40	28/08/2020	Draft	Internal Review 2	Stylianos Karagiannis (PDMFC)
0.50	31/08/2020	Pre - Final	2nd consolidated version, addressing reviewers' comments	Dimitris Apostolakis (FINT), Anargyros Sideris (FINT)
0.60	31/08/2020	Pre - Final	Quality Control	George Doukas (NTUA), Michael Kontoulis (NTUA)
1.00	31/08/2020	Final	Final	Christos Ntanos (NTUA)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Executive Summary

Deliverable D4.4 reports on the implementation status of the SPHINX AI Honeypots. In this context, the document first provides a general overview of the Honeypots concept. Further to that it presents in detail the several types of Honeypots envisaged in the SPHINX ecosystem along with their internal structure and interfaces. In addition, a performance evaluation of the implemented API, through which other SPHINX entities are able to access the Honeypot generated data, is presented. It is noted that this deliverable reports on the results of the first iteration (M13-M20) of the associated task (T4.4). The next iteration's (M24-M32) outcomes concerning the final implemented system will be included in the final version of this deliverable D4.10 (M32).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Table of Figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Table of Tables

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Table of Tables

HP - Honeypot

DB - Database

API - Application Programming Interface

IDS - Intrusion Detection System

MLID - Machine Learning Intrusion Detection

UML - Unified Modelling Language

HTTP - Hyper Text Transfer Protocol

URL - Uniform Resource Locator

REST - Representational State Transfer

ARM - Acorn RISC Machine

HW - Hardware

SW - Software

MPSoC - Multiprocessor System on Chip

PS - Processing System

PL - Programmable Logic

FPGA - Field-Programmable Gate Array

VM - Virtual Machine

OS - Operating System

CPU - Central Processing Unit

CapEX - Capital Expenses

OpEX - Operating Expenses

AI - Artificial Intelligence

IT - Information Technology

UI - User Interface

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

1 Introduction

1.1 Purpose & Scope

The purpose of this document is to present the work results regarding the design and implementation of the first SPHINX AI Honeypot prototype. The scope of the current deliverable focusses on discussing:

- The various types and categories of Honeypots, summarising their advantages and disadvantages.
- The design and implementation of a low to middle interaction level Honeypot system able to generate the data needed from the MLID (Machine learning Intrusion Detection) component of SPHINX
- The design and implementation of the interfaces providing access to the Honeypot's generated data.
- Design the Honeypot's safe layer aiming to prevent any adversary from tampering with the logs and overall generated data of the Honeypot (in an attempt of hiding its presence and malicious activities).
- Package the Honeypot system in two flavours. A software (virtualized) one and a hardware one.

1.2 Structure of the deliverable

This document is structured as follows: section 2 presents a survey about Honeypots; section 3 presents the design of the AI SPHINX Honeypot and its relative interfaces. Section 4 contains a performance evaluation of the SPHINX AI Honeypot. Finally, section 5 contains the conclusions about the SPINX AI Honeypot.

1.3 Relation to other WPs & Tasks

SPHINX AI Honeypots task is directly linked to WP3 and specifically to the Machine Learning-empowered Intrusion Detection (MLID) component of SPHINX (as described in Task 3.4 – Machine Learning empowered intrusion detection using Honeypots' data). Based on data generated by the HPs, the MLID component will first be able to train the deep learning algorithms to identify incoming attacks, and then use the trained system as an early detection functioning Intrusion Detection System (IDS) that can flag and classify, in near real-time, traffic generated by the HPs and their interaction with intruders. Finally, the low-level classification outputs of the MLID component will be sent back to the HPs, which in turn will take proper actions (if needed).

Being able to directly interact with intruders and gather attack information from them, SPHINX AI Honeypots are closely related to many other tasks of the different Work Packages (WPs) developing SPHINX components (i.e. WP3, WP4 and WP5), in order to share interaction data generated by attackers with them. Importantly, this work also involves close interaction with the WP6 tasks dealing with integration and deployment components.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

2 Honeypots survey

2.1 Overview

Honeypots are computer security components used to deflect cyber-attacks and deter certain types of attacks. They try to perfectly emulate information system resources with data that seem legitimate in order to lure the attackers in them and protect the real systems of a network from them. By *information system resources* we mean every type of computer resource (workstations, servers, printers, routers and any network device). Honeypots are similar to a bait in a network. They are intentionally vulnerable and has no legitimate production value beyond the honeypot goals. Thus, a poorly configured web server, for example, that is easily and frequently exploited cannot be considered as a honeypot. If it is required by the use case, a honeynet can be made by combining two or more honeypots in the same network. We can, then, define a HoneyNet as a collection of honeypots under the control of a person or organization. A HoneyNet can have a single theme (for instance they can mimic a university network). A single theme honeynet can still run different operating systems, programs and features, in order to closely mimic our theme. If a centralized HoneyNet is combined with an analysis tool, a HoneyFarm is created. If a honeypot is used internally, it's not uncommon for the honeypot to detect and report non-malicious broadcast traffic. Broadcast traffic can be in the form of Address Resolution Protocol (ARP) packets and Windows NetBIOS broadcasts. Honeypot network segments should be designed to filter normal broadcasts away from the honeypot.

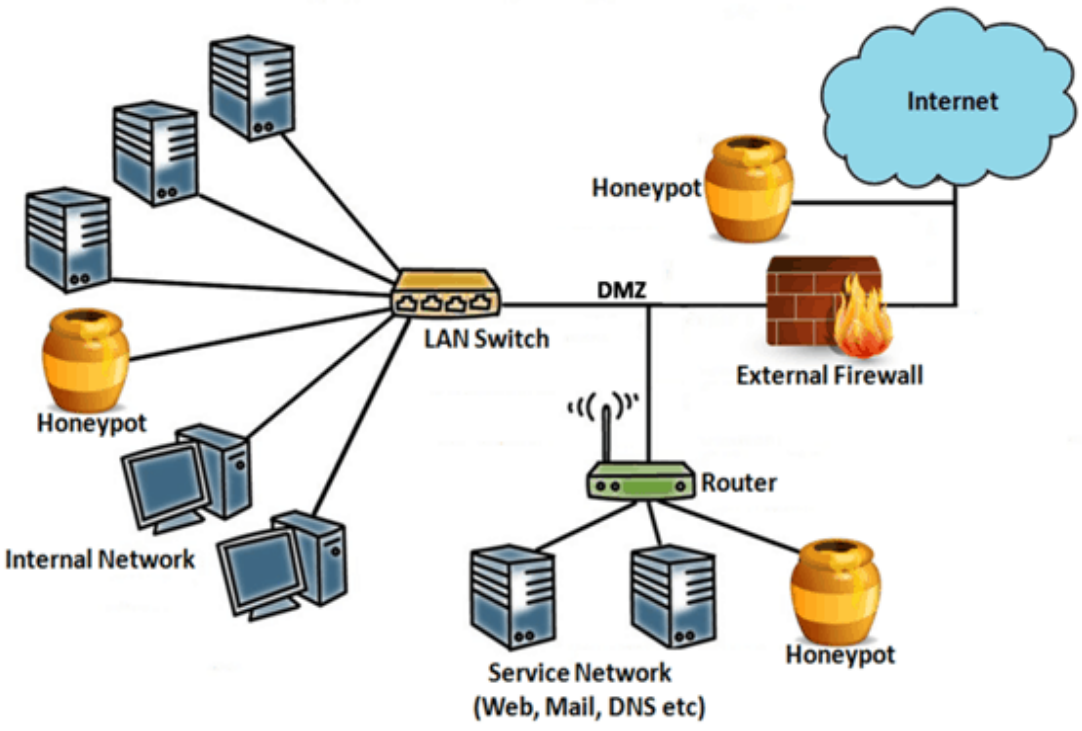


Figure 1 *Example of a Honeypot Topology [1]*

2.2 Categorization of Honeypots

There are a lot of honeypots with different functions. That is why there are a lot of categorizations.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

First of all, we have to categorize the honeypots according to their design standards to Pure Honeypots, High Interaction Honeypots and Low Interaction Honeypots. Pure Honeypots are real systems with real data that are used to lure attackers. All their activities are monitored though some hidden software on the system or by the network. Although these systems are real and lure many attackers, they cannot provide information about the attackers easily. High interaction honeypots are emulation systems that have a lot of fake services, just to waste the attacker's time. High Interaction Honeypots are usually virtual machines that can be reverted to their previous state easily and can be deployed as many times as the hardware is able to support. Low interaction honeypots are emulation of selected services that an attacker may target. Since they are selective services, this kind of honeypots require a miniscule amount of resources and they can be deployed/managed easily in any system [2].

Based on their deployment, we can classify honeypots in two categories: Production Honeypots and Research Honeypots. Production Honeypots are usually deployed in production environments, in between real services/hardware. They are used as a mouse trap, to lure attackers to them and protect the real services. Big corporations and governments use production honeypots with Deception Technology (by deploying a very large number of them in the network) in order to protect critical equipment/services. Research Honeypots on the other hand are used to learn from attackers and gather information. They are not really used for protection most of the time, but as a scientific sandbox for possible vulnerability detection. Malware honeypots for instance are part of this category because they are trying to find new malware [3].

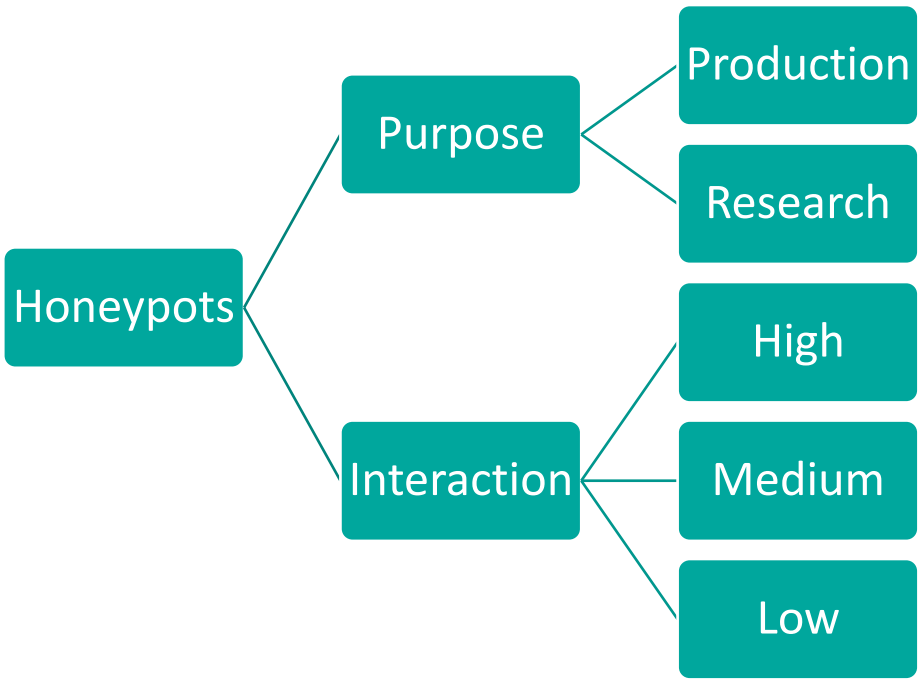


Figure 2 *Honeypot Categorization*

2.3 Advantages of Honeypots

Honeypots have some very important advantages that can help the security of a network. First of all, Honeypots can help us discover and patch potential vulnerabilities. Using a high-level Honeypot of a service will help us monitor the attacker interacting with our service. If a bug is found and/or utilised by the attacker, we will be

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

alerted and we will have the knowledge to eliminate the vulnerability. Secondly, Honeypots can help us find the attackers. Using a Honeypot, we can spot some patterns more easily, such as repeated or similar IPs, that would be difficult to detect in a real network sweep with a lot of legitimate traffic. Such findings can help us fortify our network from repetitive attacks. Another advantage of the Honeypots is that they are relatively lightweight on resources because they handle a limited amount of data traffic. That means that we can utilise old hardware for our honeypots or low-cost small board computers. As already mentioned, honeypots have no legitimate production value, and therefore, normally, should never be accessed by anyone but the honeypot administrator. So, knowing that any honeypot traffic (outside the expected administrative traffic) is probably malicious, previously unknown attacks can be found just as quickly as known attack vectors. For example, at least two major zero-day exploits were first discovered and documented by honeypots [4]. New hacking methods, while not necessarily zero-day, are discovered routinely by honeypots. There are even research tools that allow brand-new threats discovered by a honeypot to automatically generate an IDS signature [4]. Unlike a virus scanner or signature-detecting IDS, honeypots are excellent at detecting new threats. Moreover, one of the most important advantages of honeypots is the low number of false positives and false negatives they have [4] (a false positive is an error in which a security tool indicates that non-malicious activity is malicious while a false negative is the exact opposite error where the tool fails to indicate malicious activity when it exists). Security logs with many false positives are considered to contain a lot of noise. However, by using a honeypot we can prioritize our attention in real attacks and use the honeypot's logs in conjunction with other security solutions to better refine our network security. Honeypots, also, are a great tool for training technical staff, as they can see the attacks in real time without interacting with our real network. Finally, by utilizing honeypots in our network (or creating a honeynet), you can lure attackers and waste their resources and time trying to attack these honeypots [5].

2.4 Disadvantages of Honeypots

While honeypots can help us make our network more secure and they are an important security tool, they have some important disadvantages that we have to take into consideration. First of all, if the honeypot is not configured properly to behave like our real service, it can lead the attackers to just exclude the honeypot from their attacks and attack instead real services. Moreover, if the attackers have figured out the honeypot, they can create fake attacks and fake traffic, just to distract us from real attacks. Finally, a High-Level Interaction Honeypot can actually help the attacker to intrude into the real systems [5]. High-Level Honeypots, by design, are honeypots that can emulate/present a real system (identical to the real service) that has some vulnerabilities. An attacker can utilise these vulnerabilities to attack our real systems though our honeypot. Therefore, honeypots should never be used as is, without any other security tools (as a Firewall or/and IDS). In this way attacks detected by honeypots can also initiate other proactive defences. For example, if a honeypot detects malicious activity, it can update firewall rules to make sure the hackers never get access to any production assets. On the downside, any type of automated defence tool has the potential to react too quickly to false-positives and generate a self-created DoS attack [4].

2.5 Conclusion

Honeypots are a great tool to help as fortify our network. According to their level of interaction they can emulate our systems in order to lure attackers away from our real systems. That means that they can be used, in real time, to protect our network in conjunction with other security tools. Among all these types of Honeypot low-interaction Honeypot is the mostly used, since it is easy to implement and to manage. However, the most secure and efficient Honeypot type is the High-interaction Honeypot. These honeypots provide security as well

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

as generates a log about all entries in the system which is very helpful to find the intrusive activity in the system. But the honeypot must need to upgrade to new methods and attacks at some interval of time to provide security against new type to attacks. It can't be said as a solution but it is a good supplement for the security system. The ideal security system would be a Honeypot working together with an IDS, a Firewall and Antivirus. With such a security system a Production server will be fortified from almost every possible attack.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

3 AI SPHINX Honeypot

Following the introduction to the general concept of Honeypots (see the Honeypots survey in the previous section), we can move on to a detailed presentation of the SPHINX AI Honeypots specific characteristics and features. The following subsections aim to provide extensive information on the SPHINX AI Honeypots, their internal structure and interfaces.

3.1 Honeypot Design and Modules

The following figure (

Figure 3), shows a component diagram, also known as a UML component diagram, describing the organization and wiring of the physical components in the SPHINX HP system. Component diagrams are really useful and (probably) the most appropriate way to model implementation details and ensure that every aspect of the system's required function is covered by planned development.

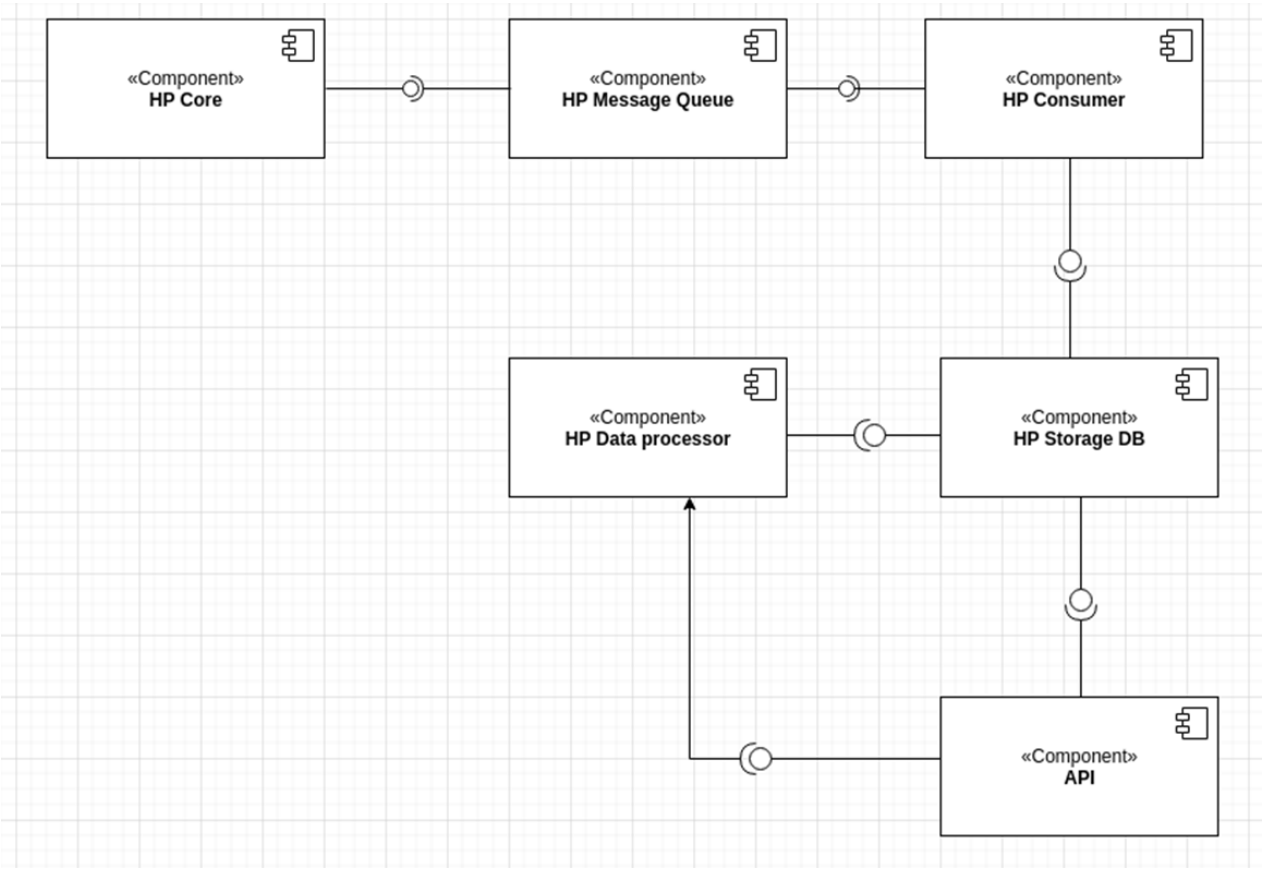


Figure 3 *SPHINX AI Honeypot Component Diagram*

As can been seen from

Figure 3, Honeypot consists of the following components:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

- HP Core
- HP Message Queue
- HP Data Consumer
- HP Storage DB
- HP API
- HP Data Processor

Table 1 presents in more detail each one of the aforementioned HP components.

HP - Component	Description
HP Core	HP Core is the main component of Honeypot. It is responsible for the simulation of the following services: • SSH • FTP • SMTP • HTTP
HP Message Queue	HP Message Queue is the component responsible for retrieving attack data recorded from HP Core. After retrieving the data, it sends them to HP Data Consumer.
HP Data Consumer	HP Data Consumer receives attack data from HP Message Queue. After that, depending on the service of the attack, it stores them in the corresponding tables in the HP Storage DB.
HP Storage DB	This component is responsible for storing attack logs data from each HP Core service.
HP API	HP API provides a REST interface to components that want to consume attack data logs for the HP Core services. It provides endpoints for the following services: • SSH • FTP • SMTP • HTTP It also includes a dedicated endpoint for retrieval of MLID data. Furthermore, upon demand, it can request MLID data from HP Data Processor.
HP Data Processor	This component is responsible for gathering attack logs data from the HP Storage DB. By processing the attack logs it can produce the data needed by the MLID component of SPHINX.

***Table 1** Description of HP components*

Towards facilitating the deployment and maintenance of the SPHINX Honeypots (i.e. their soft components) the docker framework is exploited; this also enables the SPHINX AI Honeypots to work as farms and to be dynamically (re)configured by utilising, for example, the SPHINX's situation awareness system.

All Honeypot components are implemented in Golang language [6]. Components that provide REST API endpoints use the Beego web framework [7]. Furthermore, the dockerclient Golang package [8] was used for handling the dockerised HP components and the go-sqlite3 Golang package [9] for implementing the data model in an SQLite3 database [10]. Intercommunication between the HP components is performed using RabbitMQ.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Figure 3 also illustrates the basic interactions between the Honeypot components; the HP Core sends data to the HP Message Queue via an interface. The HP storage DB receives data from the HP Consumer. It also sends data to both HP data processor and HP API. Finally, the HP API component sends data to the HP data processor.

However, interactions between the Honeypot components will be better described by the following sequence diagram. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. The main purposes of a sequence diagram are:

- Model high-level interaction between objects in a system
- Model the interaction between object instances within a collaboration that realizes a use case
- Model the interaction between objects within a collaboration that realizes an operation
- Either model generic interactions (showing all possible paths through the interaction) or specific instances of an interaction (showing just one path through the interaction)

Figure 4 shows the following actions: when someone attacks the Honeypot, the HP Core gathers the attack data and sends them to the HP message queue. After that, the HP message queue pushes these data to HP Consumer, where the HP consumer categorizes them and stored them to the HP storage DB.

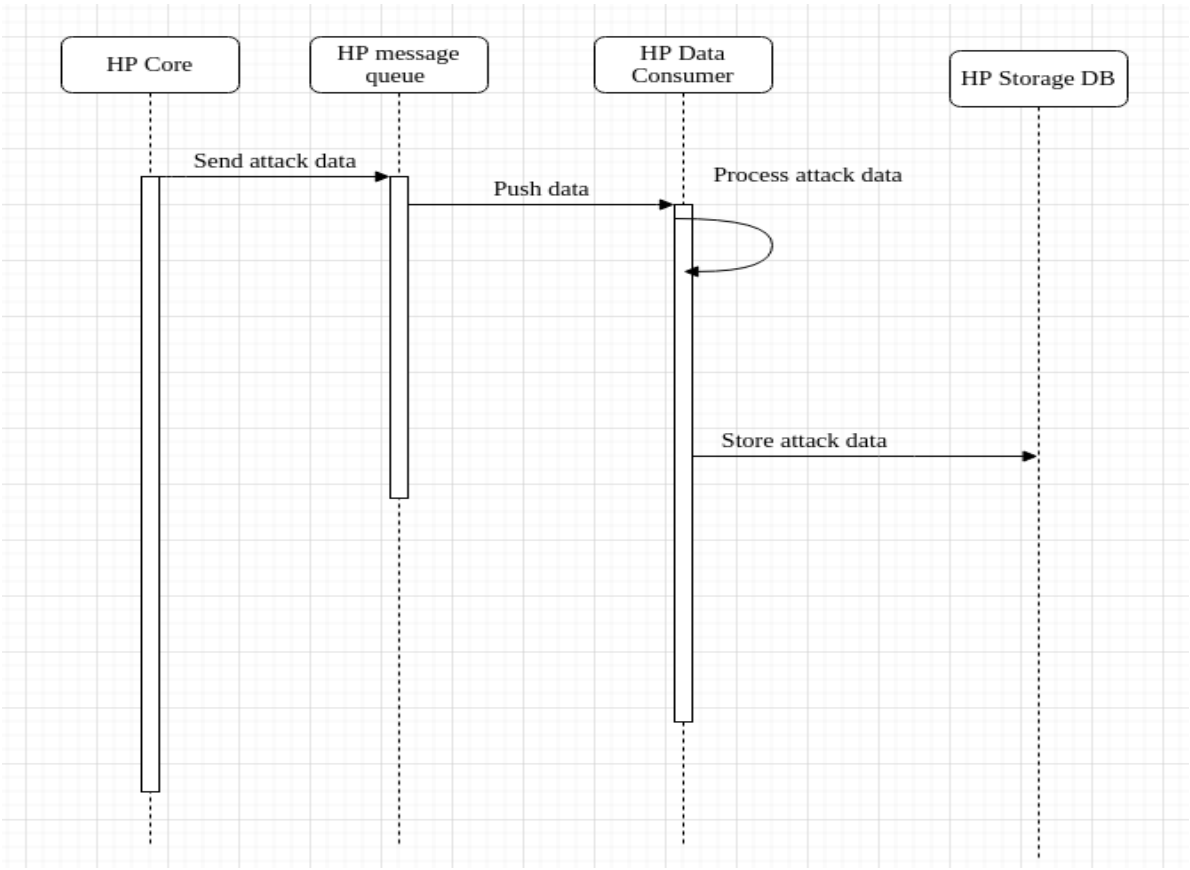


Figure 4 *HP Sequence Diagram – Storing Attack Data*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

The following diagram illustrates the actions that happen when attack data are requested from the HP API. First the HP API requests the data from the HP data processor. After that the HP data processor retrieves the data from the HP storage DB, processes them and send them back to the HP API. Finally, the HP API returns the requested data to the corresponding component.

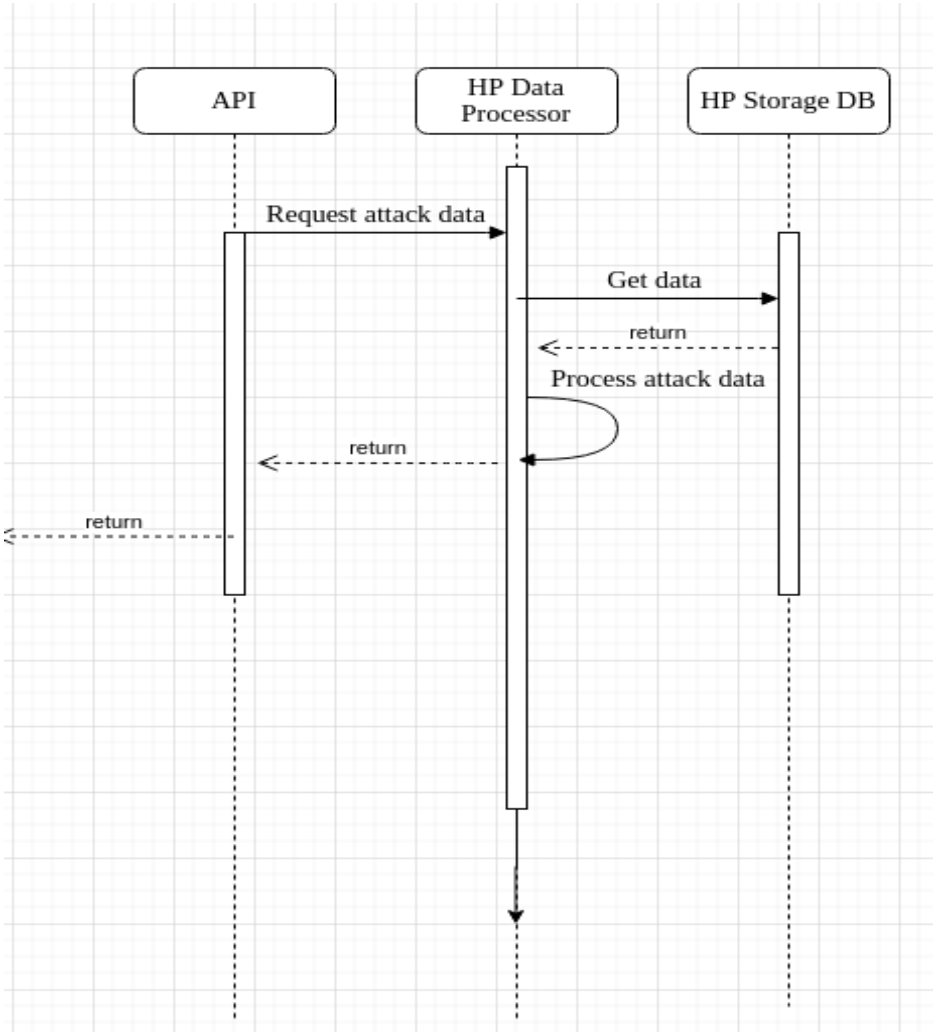


Figure 5 *HP Sequence Diagram – Retrieving Attack Data from API*

3.2 Honeypot AI

One of the most challenging and intensive tasks in the SPHINX HP development process is related with their ability to support AI algorithms designed to detect anomalies (e.g. attempt to install malware in the authority's IT infrastructure). For this purpose, particular attention and time was devoted to implement the HP Data Processor presented in the previous section. This module performs sophisticated algorithms in order to properly process the attack information gathered by HPs, and generate data in a format that AI algorithms (such as the SPHINX MLID component) can understand, manage and use to detect attacks, so that prompt and effective action can be taken, as appropriate.

The HP-generated dataset, is a variant of the most widespread IDS benchmark dataset at present, named NSL-KDD dataset [11], extended to include a few additional features necessary according to the needs expressed in

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

SPHINX so far. For the time being, our dataset consists of 46 features per record, but new ones may be added if needed. The first 44 features (Table 2) are related to the traffic input itself, while the last two are labels indicating the type of the attack and the severity of the traffic input itself and they are used only during the training phase of the AI-based systems. The following table explains the meaning of all 44 features referring to HP traffic input.

No	Feature	Description
1	Timestamp	timestamp of the last recorded action within a session
2	Source IP	IP of the attacker
3	Destination IP	IP of the victim
4	Duration	length (number of seconds) of the connection
5	Protocol type	type of the protocol, e.g. tcp, udp, etc
6	Service	network service on the destination, e.g., http, telnet, etc.
7	Flag	normal or error status of the connection
8	Source bytes	number of data bytes from source to destination
9	Destination bytes	number of data bytes from destination to source
10	Land	1 if connection is from/to the same host/port; 0 otherwise
11	Wrong fragment	number of wrong fragments
12	Urgent	number of urgent packets
13	Hot	number of hot indicators
14	Number failed logins	number of failed login attempts
15	Logged in	1 if successfully logged in; 0 otherwise
16	Num compromised	number of compromised conditions
17	Root shell	1 if root shell is obtained; 0 otherwise
18	Su attempted	1 if su root command attempted; 0 otherwise
19	Num root	number of root accesses
20	Num file creations	number of file creation operations
21	Num shells	number of shell prompts
22	Num access files	number of operations on access control files
23	Num outbound cmds	number of outbound commands in an ftp session
24	Is host login	1 if the login belongs to the hot list; 0 otherwise
25	Is guest login	1 if the login is a guest login; 0 otherwise
26	Count	number of connections to the same host as the current connection in the past two seconds
27	Srv count	number of connections to the same service as the current connection in the past two seconds
28	Serror rate	number of connections to the same host as the current connection in the past two seconds
29	Srv serror rate	% of connections that have SYN errors
30	Rerror rate	% of connections that have REJ errors
31	Srv rerror rate	% of connections that have REJ errors
32	Same srv rate	% of connections to the same service
33	Diff srv rate	% of connections to different services
34	Srv diff host rate	% of connections to different hosts
35	Dst host count	Number of connections to the same host to the destination host as the current connection in the past 2 seconds
36	Dst host srv count	Number of connections from the same service to the destination host as the current connection in the past 2 seconds

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

37	Dst host same srv rate	% of connections from the same services to the destination host
38	Dst host diff srv rate	% of connections from the different services to the destination host
39	Dst host same src port rate	% of connections from the port services to the destination host
40	Dst host srv diff host rate	% of connections from the different hosts from the same service to the destination host
41	Dst host serror rate	% of connections that have SYN errors from the same host to the destination host
42	Dst host srv serror rate	% of connections that have SYN errors from the same service to the destination host
43	Dst host rerror rate	% of connections that have REJ errors from the same host to the destination host
44	Dst host srv rerror rate	% of connections that have REJ errors from the same service to the destination host

***Table 2** List of features referring to traffic input*

3.3 Honeypot safe layer

In order to protect the Honeypot's generated logs and data from any tampering by the potential attacker, a two-layer virtualized and minified system was created (see

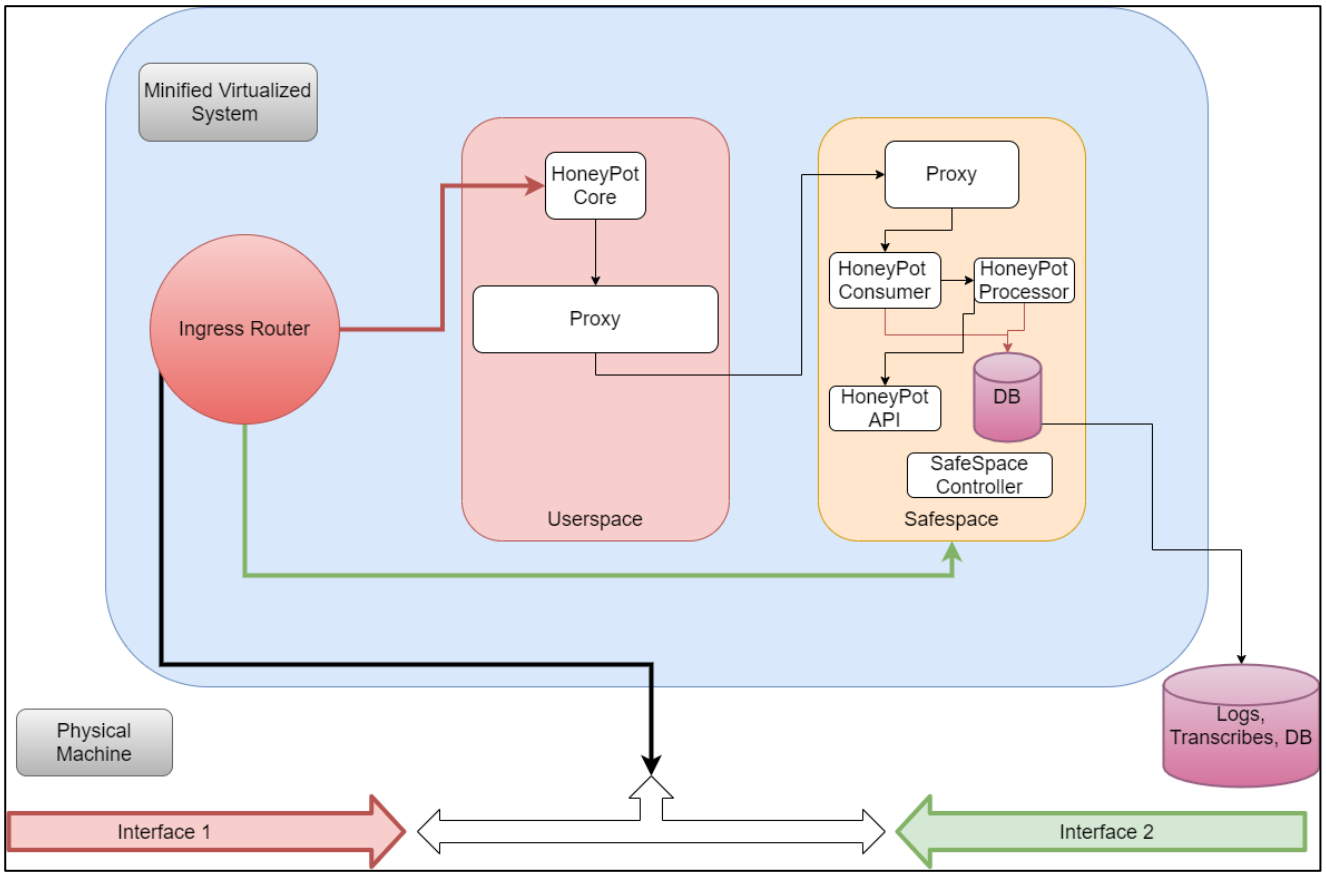


Figure 6). The first layer, named Userland, holds all the honeypot-emulated services (i.e. the basic core of the HP) and is the layer that it is visible by the attacker. The second layer is the Safe Layer where all the other HP modules reside in order to collect and manage the Honeypot generated data. The two layers are separated with

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

distinct networks and they can only communicate through two proxies, one in the UserLand and the other in the Safe Layer. The *Consumer* and *Processor* modules within the Safe Layer, take the data from the HP Core and transform them to data that we can store and analyse.

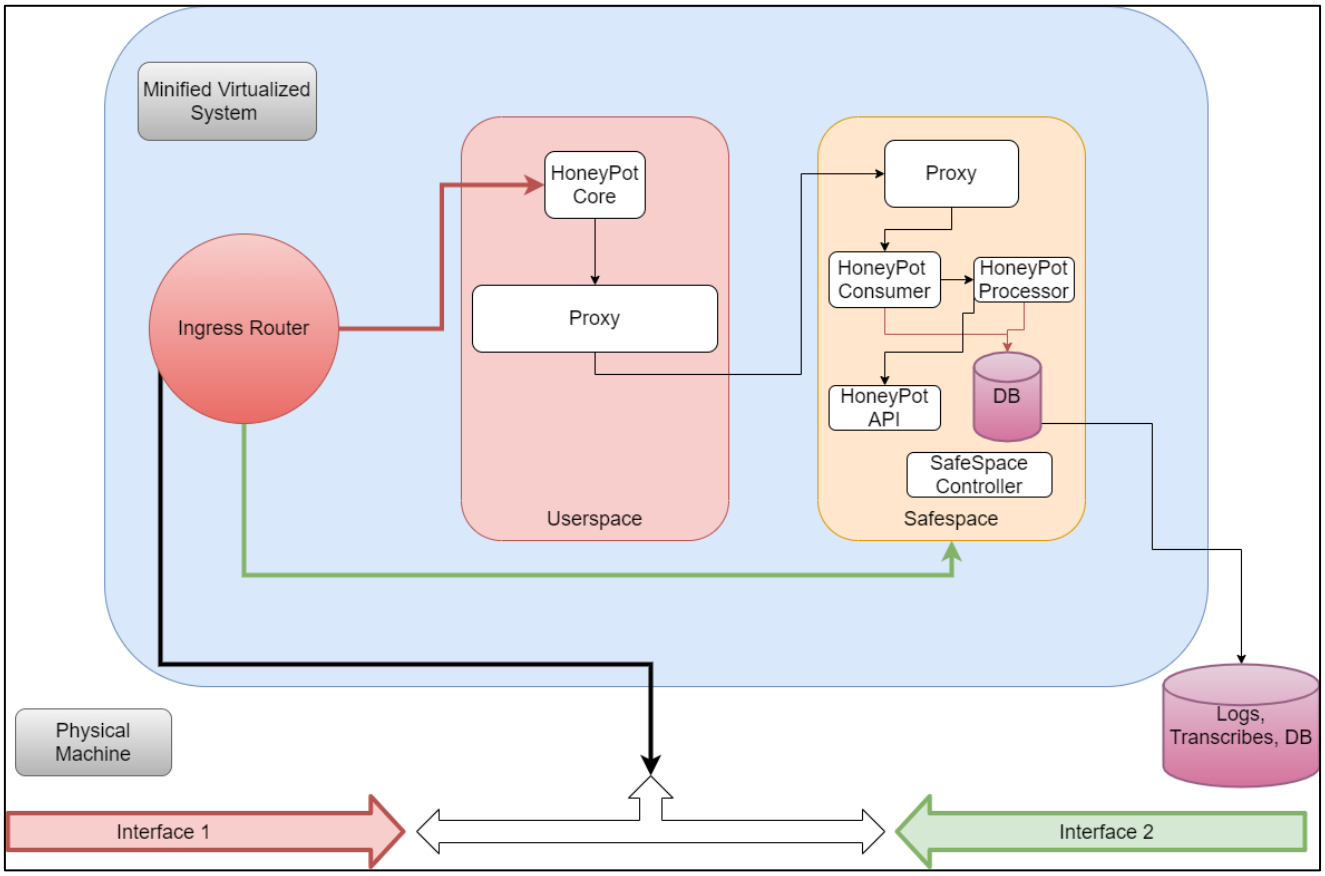


Figure 6 *The two-layer virtualized Honeypot System*

The *API* module is used for the communication of the honeypot system with other SPHINX components and to retrieve data as well. In the Database (DB) we save the analysed data as logs and vital Information. All the logs and vital information is also saved through our Safe Layer to the physical device or virtualized system. In order to physically protect the Safe Layer from attacks, a user can communicate with the Safe Layer only through a second interface. That means that from the main network interface a user can only interact with the UserLand and a second network interface (virtualized or physical) has to be used for managerial purposes.

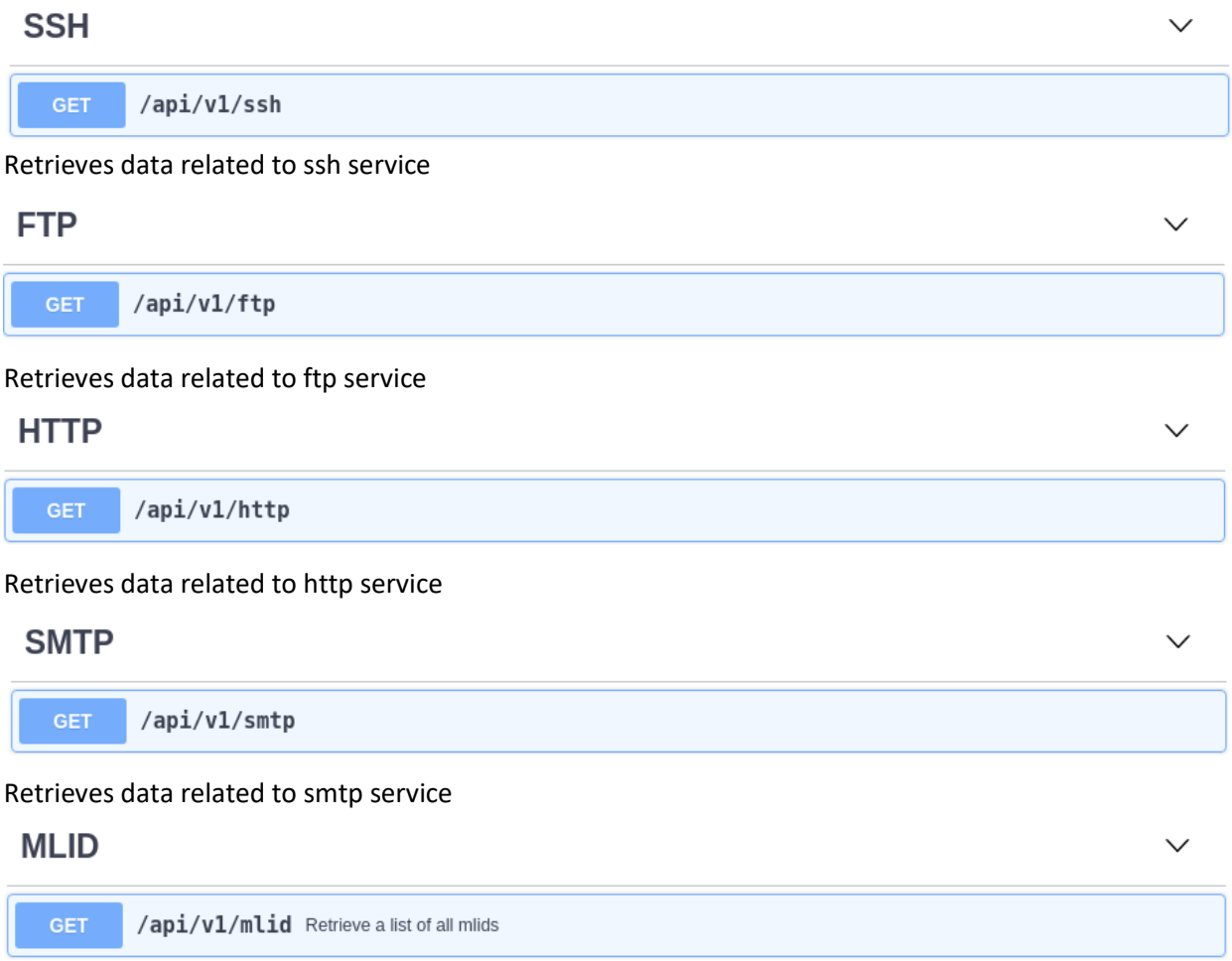
3.4 Honeypot interfaces

As already mentioned, Honeypots have to be in constant communication with many other SPHINX components, in order to share the data generated during the interactions of the attackers with them. So far, HP data are accessible by other SPHINX entities (e.g. the MLID component) via a REST API (API design can be seen in Annex II: HP API swagger file [12]) described in the following subsections.

3.4.1 HP API endpoints

HP API Component provides the following endpoints for retrieving data regarding the HP Core services

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.



3.4.2 Data models

Below you will find the data models associated with each API endpoint:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

SSH ⌄ {

date	string *example: January 8th 2020, 13:13:55.255*	The full date of the incident
destination_ip	string	The IP of the destination of the packet (the Server's IP)
destination_port	number	The PORT of the destination of the packet (the Server's PORT)
source_ip	string	The IP of the source packet (the attack IP)
source_port	number	The PORT of the source packet (the attack PORT)
ssh_password	string *example: 123456*	The password used in the attack
sessionid	string	The session id created by the ssh honeypot
ssh_username	string *example: root*	The username used in the attack
type	string *example: password-authentication*	The method of authentication used [password-authentication, ssh-certificate]

}

***Figure 7** SSH Data model*

Figure 7 shows the data model associated with the SSH API endpoint. By performing a GET request to this endpoint, we are able to retrieve all records relevant to the SSH service. Each record contains some basic network information (such as source/destination port and IP) for a specific connection. It also contains metadata specific to the SSH service (e.g. sessionid, ssh_username, ssh_password).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

HTTP ⌄ {

date	string	*example: January 8th 2020, 13:13:55.255* The full date of the incident
destination_ip	string	The IP of the destination of the packet (the Server's IP)
destination_port	number	The PORT of the destination of the packet (the Server's PORT)
source_ip	string	The IP of the source packet (the attack IP)
source_port	number	The PORT of the source packet (the attack PORT)
http_header	string	*example: http.header.accept:image/webp,image/apng,image/*,*/*;q=0.8 http.header.accept-encoding:gzip, deflate, br http.header.accept-language:en-US,en;q=0.9,el-GR;q=0.8,el;q=0.7 http.header.connection:keep-alive http.header.referer:http://127.0.0.1:8080/ http.header.sec-fetch-mode:no-cors http.header.sec-fetch-site:same-origin http.header.user-agent:Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.70 Safari/537.36* All the lines used in the HTTP header of the HTTP Request
http_host	string	*example: 127.0.0.1:80* The virtual Host, used by the Honeypot
http_method	string	*example: GET* The method used in the HTTP Request
http_proto	string	*example: HTTP/1.1* The protocol used in the HTTP Request
sessionid	string	The session id created by the HTTP honeypot
url	string	The url used in the HTTP Request
payload	string	The total answer of the HTTP Response, made by the HTTP Honeypot

}

Figure 8 *HTTP Data model*

Figure 8 shows the data model associated with the HTTP API endpoint. By performing a GET request to this endpoint, we are able to retrieve all records relevant to the HTTP service. Each record contains some basic network information (such as source/destination port and IP) for a specific connection. It also contains metadata specific to the HTTP service (e.g. sessionid ,http headers, http method, http protocol).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

FTP {

date	string *example: January 8th 2020, 13:13:55.255*	The full date of the incident
destination_ip	string	The IP of the destination of the packet (the Server's IP)
destination_port	number	The PORT of the destination of the packet (the Server's PORT)
source_ip	string	The IP of the source packet (the attack IP)
source_port	number	The PORT of the source packet (the attack PORT)
ftp_command	string *example: PASS admin*	The command used in the FTP attack
ftp_username	string	Username used for ftp authentication
ftp_password	string	Password used for ftp authentication
sessionid	string	The session id created by the FTP honeypot

}

Figure 9 *FTP Data model*

Figure 9 shows the data model associated with the FTP API endpoint. By performing a GET request to this endpoint, we are able to retrieve all records relevant to the FTP service. Each record contains some basic network information (such as source/destination port and IP) for a specific connection. It also contains metadata specific to the FTP service (e.g. sessionid , ftp_username,ftp_password).

SMTP {

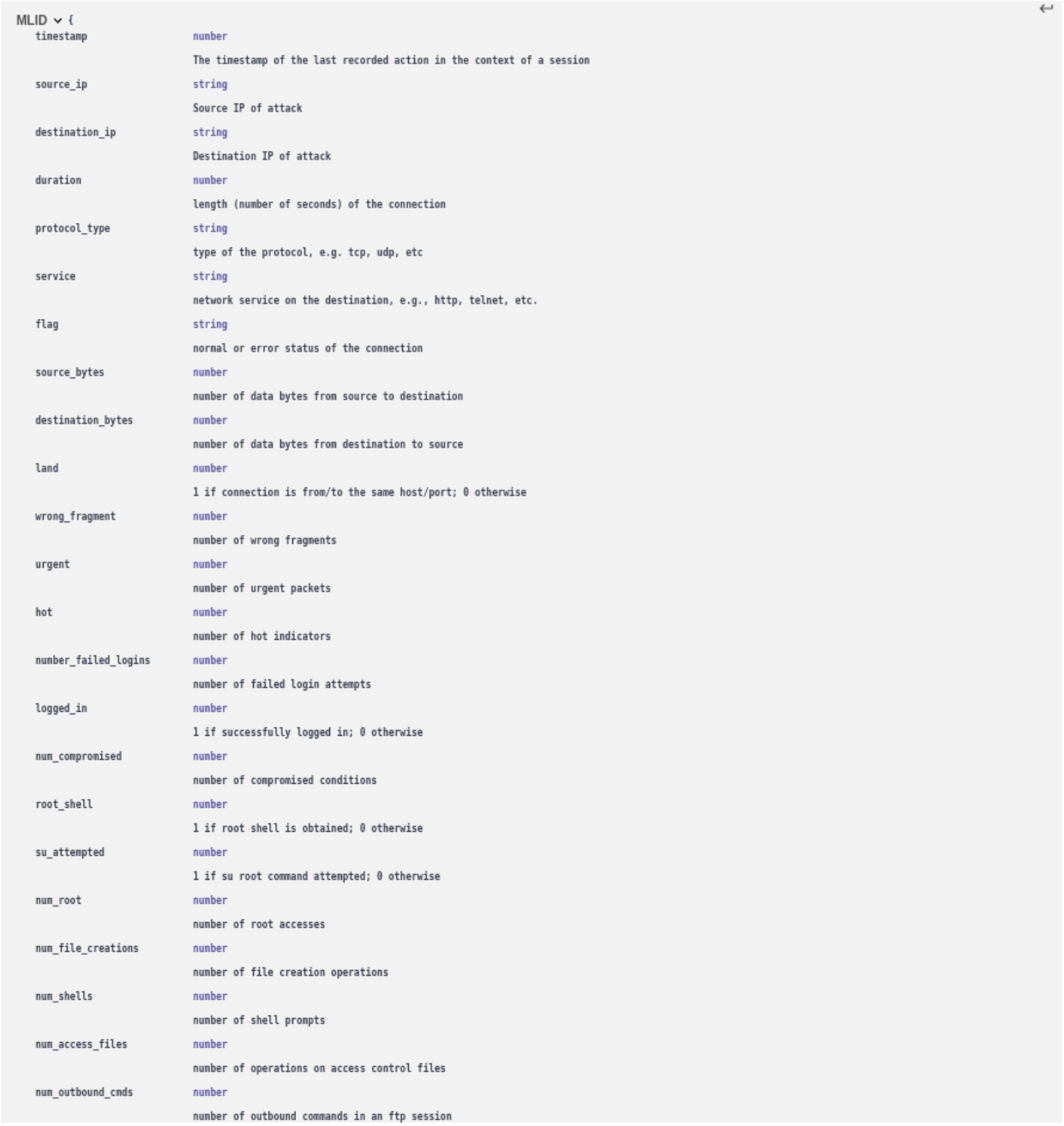
date	string *example: January 8th 2020, 13:13:55.255*	The full date of the incident
destination_ip	string	The IP of the destination of the packet (the Server's IP)
destination_port	number	The PORT of the destination of the packet (the Server's PORT)
source_ip	string	The IP of the source packet (the attack IP)
source_port	number	The PORT of the source packet (the attack PORT)
smtp_line	string *example: AUTH PLAIN XXXXXXXX*	The command used in the SMTP attack
sessionid	string	The session id created by the SMTP honeypot

}

Figure 10 *SMTP Data model*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

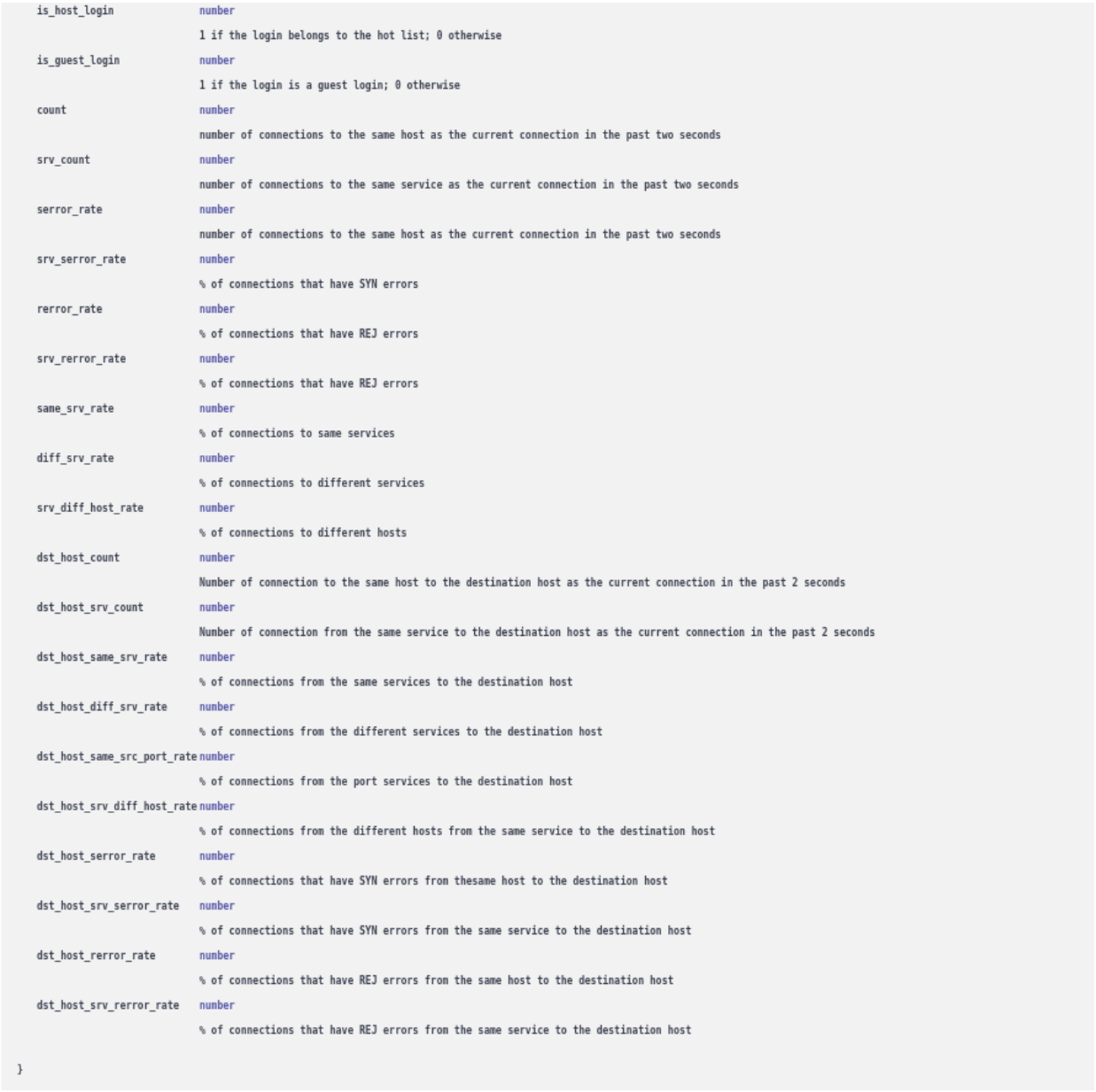
Figure 10 shows the data model associated with the SMTP API endpoint. By performing a GET request to this endpoint, we are able to retrieve all records relevant to the SMTP service. Each record contains some basic network information (such as source/destination port and IP) for a specific connection. It also contains metadata specific to the SMTP service (e.g. sessionid , smtp_line).



MLID ∨ {

timestamp	number	
		The timestamp of the last recorded action in the context of a session
source_ip	string	
		Source IP of attack
destination_ip	string	
		Destination IP of attack
duration	number	
		length (number of seconds) of the connection
protocol_type	string	
		type of the protocol, e.g. tcp, udp, etc
service	string	
		network service on the destination, e.g., http, telnet, etc.
flag	string	
		normal or error status of the connection
source_bytes	number	
		number of data bytes from source to destination
destination_bytes	number	
		number of data bytes from destination to source
land	number	
		1 if connection is from/to the same host/port; 0 otherwise
wrong_fragment	number	
		number of wrong fragments
urgent	number	
		number of urgent packets
hot	number	
		number of hot indicators
number_failed_logins	number	
		number of failed login attempts
logged_in	number	
		1 if successfully logged in; 0 otherwise
num_compromised	number	
		number of compromised conditions
root_shell	number	
		1 if root shell is obtained; 0 otherwise
su_attempted	number	
		1 if su root command attempted; 0 otherwise
num_root	number	
		number of root accesses
num_file_creations	number	
		number of file creation operations
num_shells	number	
		number of shell prompts
num_access_files	number	
		number of operations on access control files
num_outbound_cmds	number	
		number of outbound commands in an ftp session

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.



is_host_login	number	1 if the login belongs to the hot list; 0 otherwise
is_guest_login	number	1 if the login is a guest login; 0 otherwise
count	number	number of connections to the same host as the current connection in the past two seconds
srv_count	number	number of connections to the same service as the current connection in the past two seconds
serror_rate	number	number of connections to the same host as the current connection in the past two seconds
srv_serror_rate	number	% of connections that have SYN errors
rerror_rate	number	% of connections that have REJ errors
srv_rerror_rate	number	% of connections that have REJ errors
same_srv_rate	number	% of connections to same services
diff_srv_rate	number	% of connections to different services
srv_diff_host_rate	number	% of connections to different hosts
dst_host_count	number	Number of connection to the same host to the destination host as the current connection in the past 2 seconds
dst_host_srv_count	number	Number of connection from the same service to the destination host as the current connection in the past 2 seconds
dst_host_same_srv_rate	number	% of connections from the same services to the destination host
dst_host_diff_srv_rate	number	% of connections from the different services to the destination host
dst_host_same_src_port_rate	number	% of connections from the port services to the destination host
dst_host_srv_diff_host_rate	number	% of connections from the different hosts from the same service to the destination host
dst_host_serror_rate	number	% of connections that have SYN errors from thesame host to the destination host
dst_host_srv_serror_rate	number	% of connections that have SYN errors from the same service to the destination host
dst_host_rerror_rate	number	% of connections that have REJ errors from the same host to the destination host
dst_host_srv_rerror_rate	number	% of connections that have REJ errors from the same service to the destination host

}

Figure 11 *MLID Data model*

Figure 11 shows the data model associated with the MLID API endpoint. By performing a GET request to this endpoint, we are able to retrieve MLID data based on some criteria such as the category of the service (i.e. SSH, FTP, HTTP, or SMTP) and the time range of the attack we are interested in.

3.5 Honeypot Management Dashboard

In order to improve the usability and facilitate the deployment, maintenance and management of SPHINX HPs, a dedicated front-end UI, called HP Dashboard was built. The HP Dashboard makes it really easy for SPHINX admins to deploy multiple HP instances on a device (physical or virtual), interact with them and manage them as well.

In order to start interacting with the Dashboard the user has to log into the system using unique credentials (username/password).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

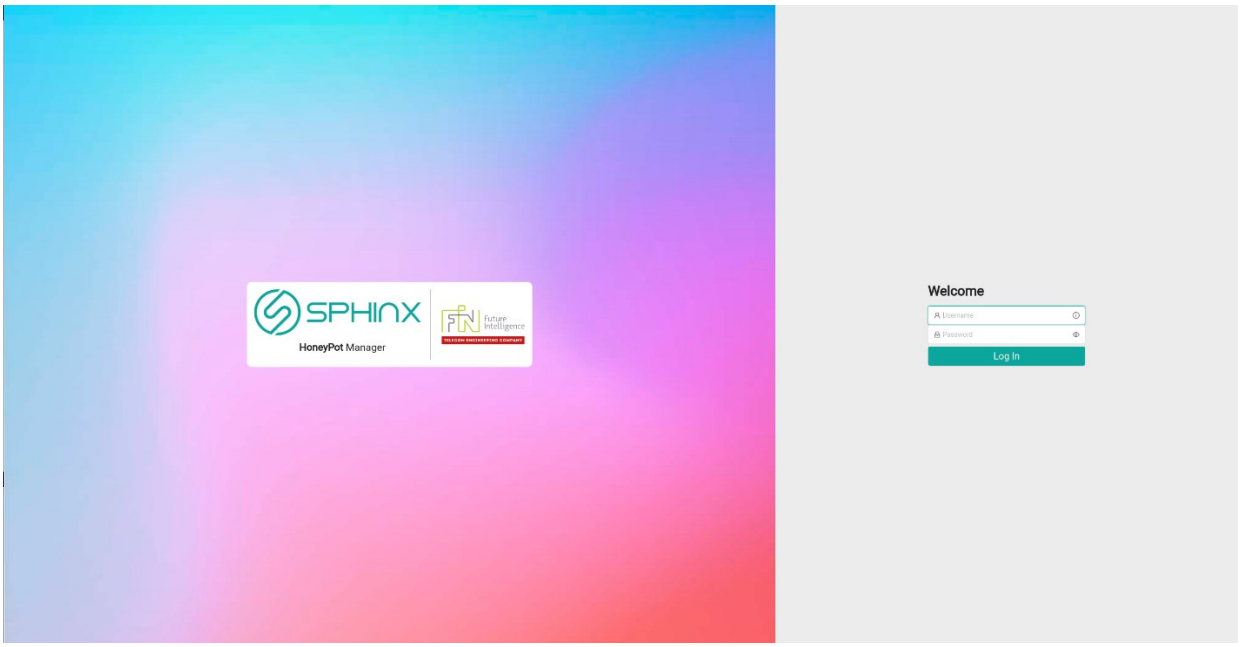


Figure 12 *Login Screen*

In the home page (Figure 13), users can easily see the number of HPs deployed on a particular system, as well as the number of times that each of the four services supported by the SPHINX HPs at present, exists in the system.

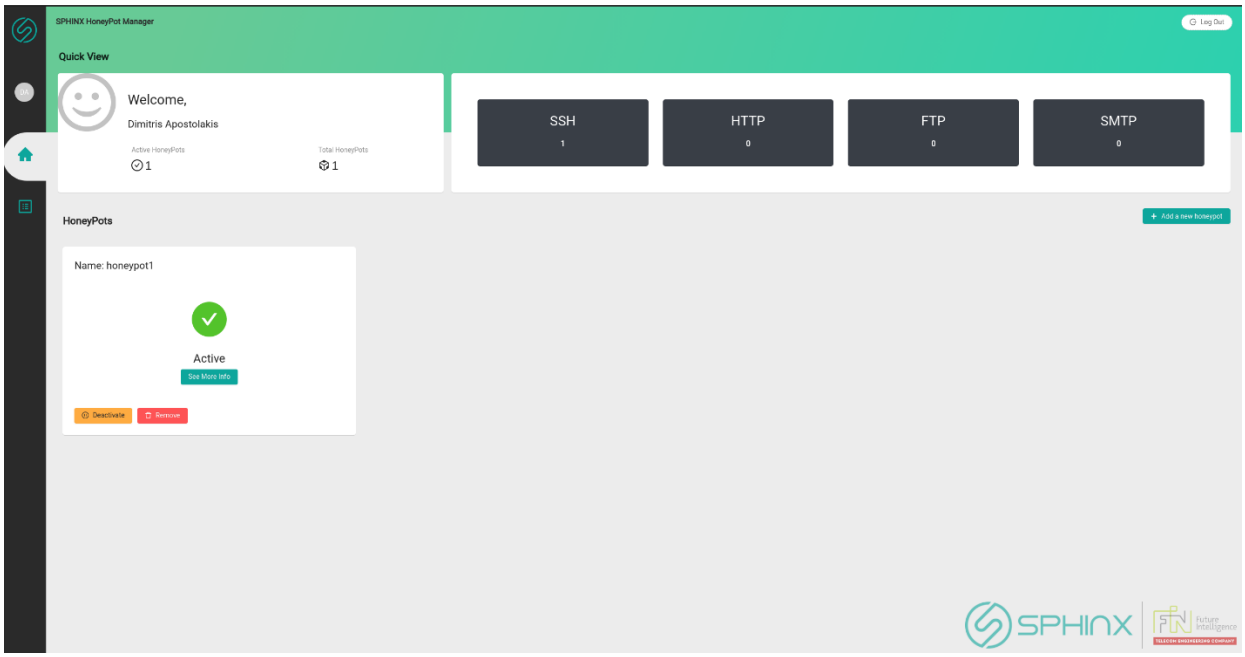


Figure 13 *Dashboard User Interface/Home*

By selecting "*Add a new honeypot*" users are able to create HPs really fast; in the pop-up window that appears on the screen (Figure 14) the user must simply provide a name for the new HP instance, and also define the emulated services within this instance and the corresponding ports to be used.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

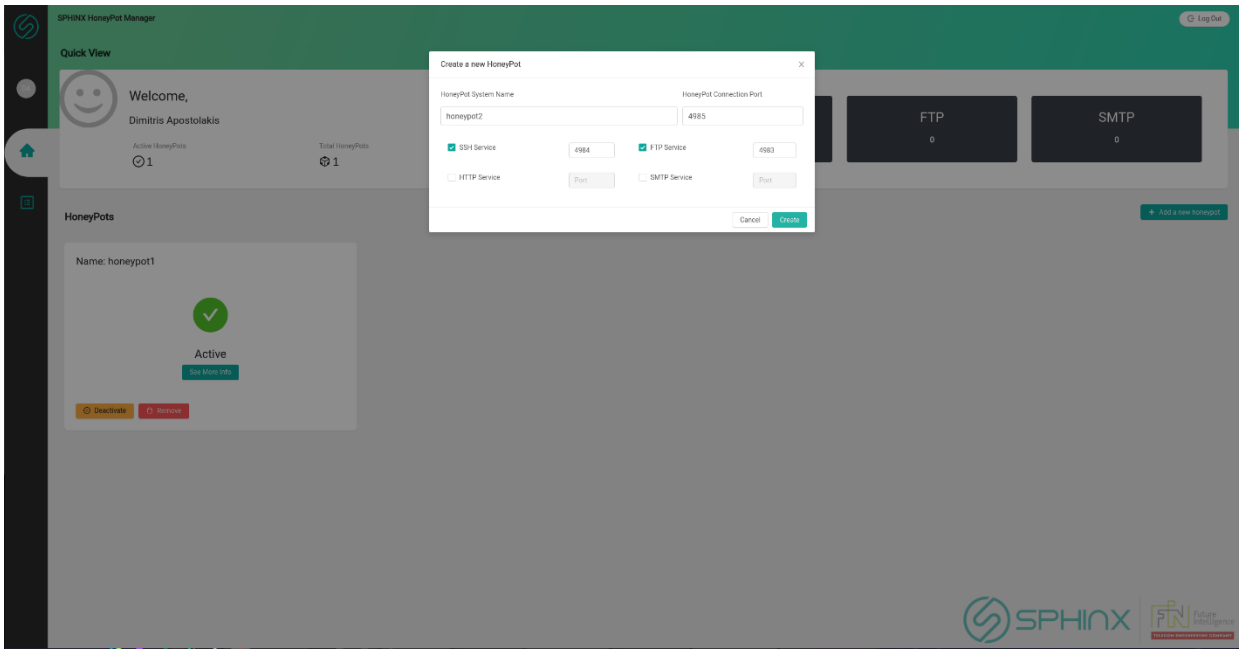


Figure 14 *Create new HP – pop-up Window*

Users can always retrieve detailed information about a particular HP instance created, by just pressing the "*See More Info*" button of the respective instance-pane displayed on the home page (Figure 15).

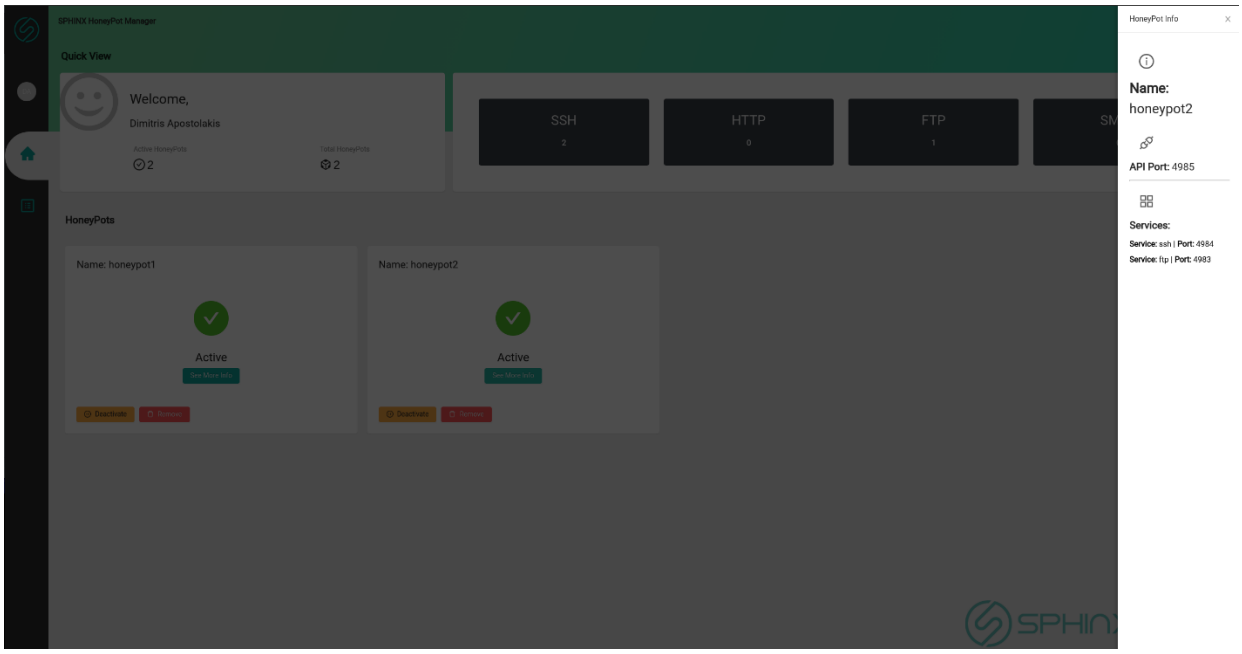


Figure 15 *Retrieve information about an existing HP*

Furthermore, the HP Dashboard provides direct access to the attack information gathered by a deployed HP and the related MLID data generated (Figure 16).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

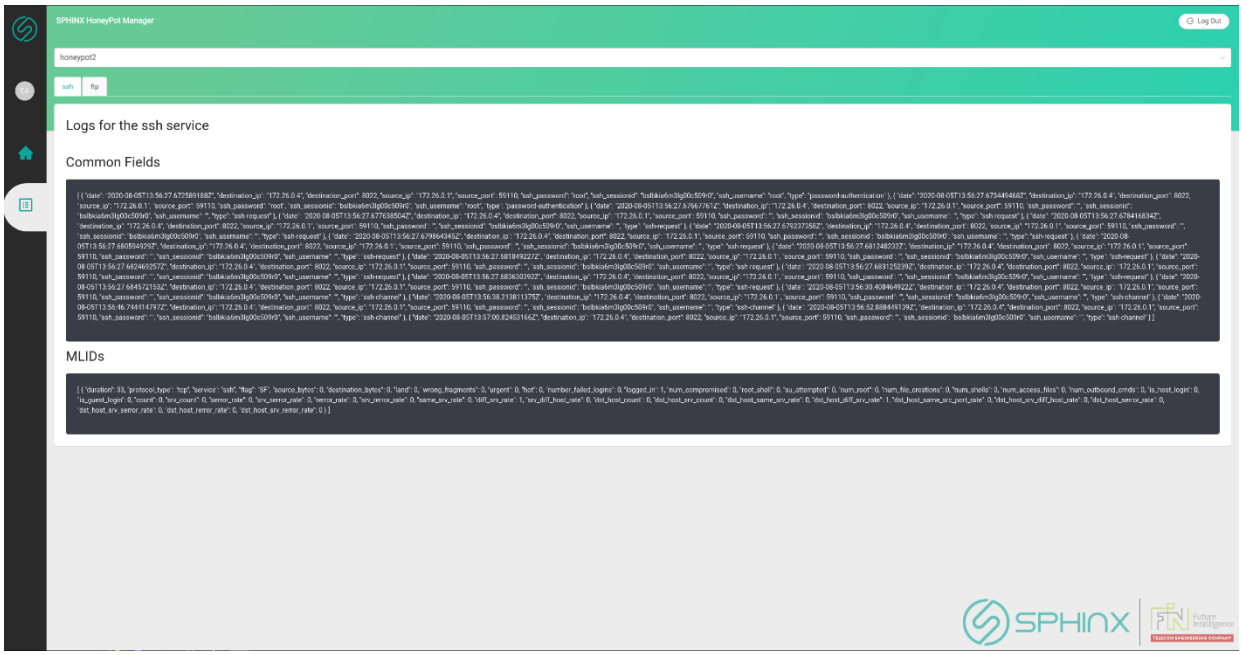


Figure 16 *Retrieve HP attack information and MLID data*

Finally, admins can easily activate/deactivate or remove deployed HPs through the HP Dashboard, at any time, by just pressing the corresponding buttons in the home page (Figure 17)

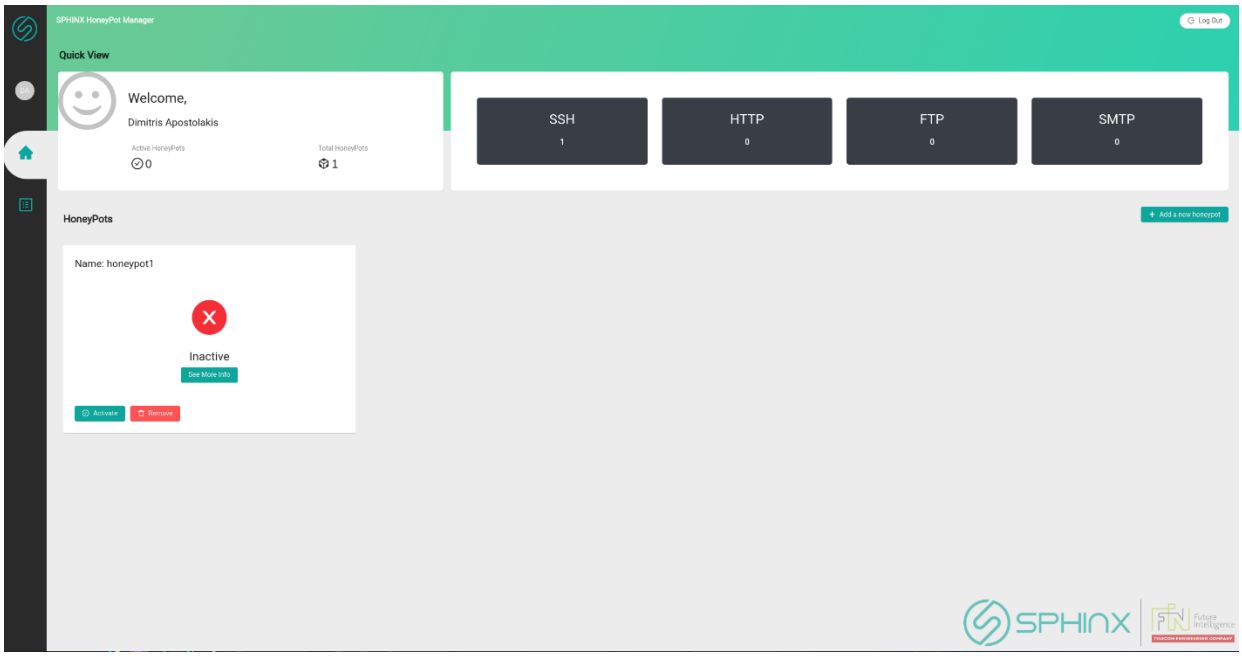


Figure 17 *Activate/De-activate/Remove HPs*

3.6 Honeypot flavours

The following two Honeypot flavours can be distinguished in SPHINX: 1) the hardware (HW) flavour and 2) the software (SW) flavour. The HW one supports the implementation of low-to-medium level interaction HPs while the SW one is able to additionally support high level interaction HPs and is suited for being deployed in virtual environments such as cloud deployments. The next subsections present in more detail these two HP flavours,

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

showing the strengths and weaknesses of each, thus also illustrating the context in which each of them could be beneficial.

3.6.1 Hardware flavour

HW flavoured Honeypots are low-to-medium level interaction HPs realized as physical (hardware) appliances that enable us to reduce both capital expenses (CapEX) and operating expenses (OpEX). These appliances are low-cost, ARM-based computing platforms with low power consumption and small physical size, able to support a full capability operating system and lightweight detection algorithms, and quick enough to effectively handle the attacks.

The most advanced and powerful HPs in this category are based on the XCZU3EG, member of the Xilinx Zynq UltraScale+ multiprocessor System on Chip (MPSoC) family [13]. The XCZU3EG combines an ARM-based processing system (PS) and a low-power FPGA of 154K programmable logic cells (PL) into a single package, thus enabling fast interconnection between them and power efficiency. This device is capable of operating as a honeypot and a router system able to efficiently handle computing intensive tasks (such as data encryption/decryption) by offloading them to the high-performance FPGA (or PL) part of the device.

3.6.2 Software flavour

SW flavoured Honeypots are realized as virtual appliances on powerful, x86-based systems offering much more computational resources and greater processing capabilities when compared to the HW flavoured ones. This flavour is mainly used for implementing high interaction honeypots, as they are complex solutions involving real operating systems and applications. Once we have created a virtual HP system, it is easy enough to re-create after it has been compromised by simply restarting the honeypot virtual session. However, since VM software runs a real copy of the software in a virtual OS environment, each HP needs as much CPU, memory, and hard drive space as the stand-alone OS needs. Consequently, running multiple SW flavoured HPs at the same time can significantly tax a high-performance system wasting large amounts of energy and valuable resources just to maintain fake services and data.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

4 Performance Evaluation

In this section we evaluate the availability and performance of the honeypot components by retrieving the generated data through the HP API. The tests were performed on a HW flavoured honeypot, based on the quad-core Cortex-A72 (ARM v8) 64-bit microarchitecture.

4.1 Evaluation of the HP API

In order to perform a more complete and thorough evaluation of the HP API the JMeter tool [14], depicted in Figure 18, was used.

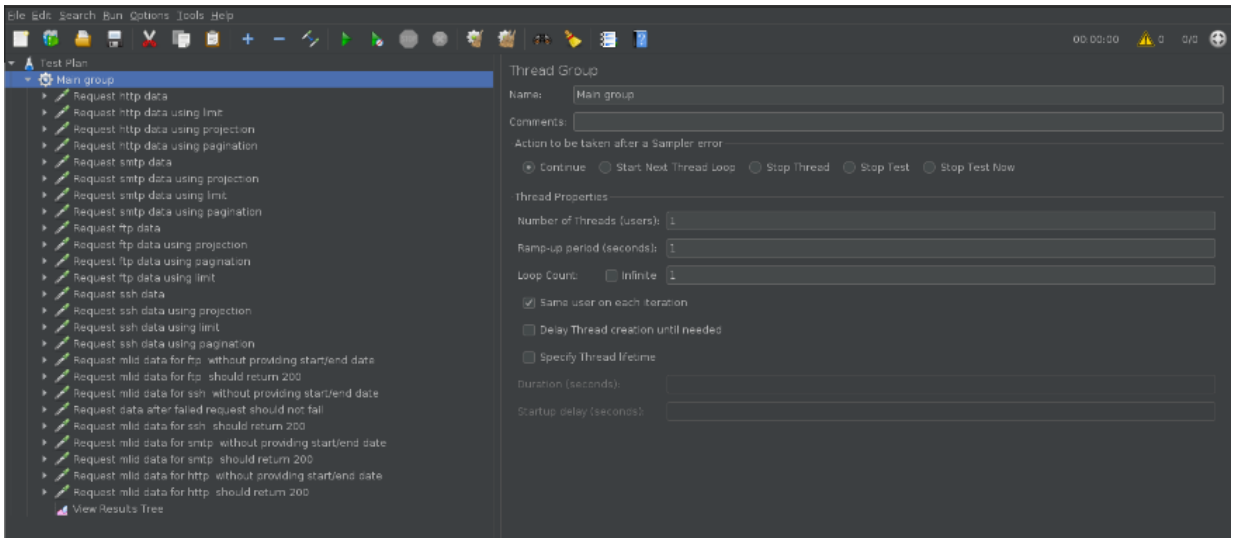


Figure 18 *Apache JMeter Main Screen*

The Apache JMeter™ is an open source, 100% pure Java software application that offers an easy way to validate the implemented API in terms of availability, output validity (response conforming to the applications interfaces out-put specifications) and input invalidity (request data not conforming to the interface input specifications). To this end, we defined, for each HTTP API endpoint contained in Table 3, several requests emulating the HTTP requests to the HP API. As an example, Figure 19, demonstrates a request to the endpoint for retrieving attack logs for the HTTP service of HP core, while Figure 20 depicts the assertion received for this request indicating with a code 200 the success of the operation. It is noted that the endpoints were tested for both valid and invalid (erroneous) input.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

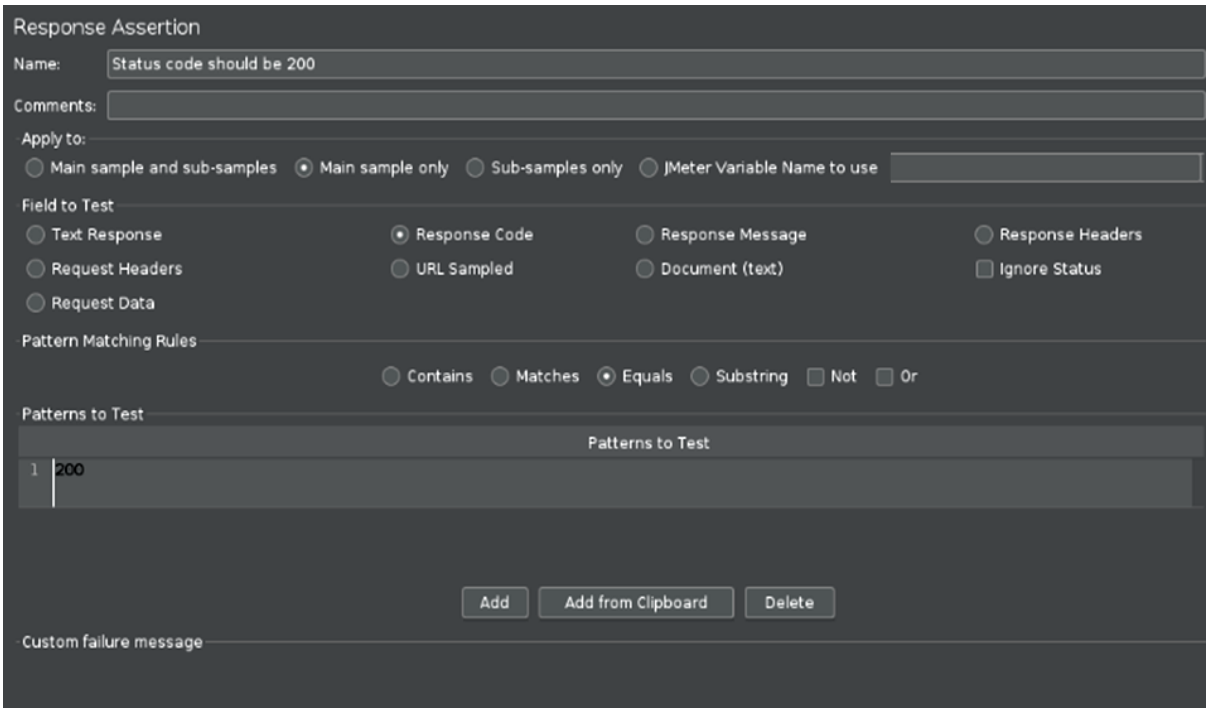


Figure 19 *Apache JMeter HTTP request*

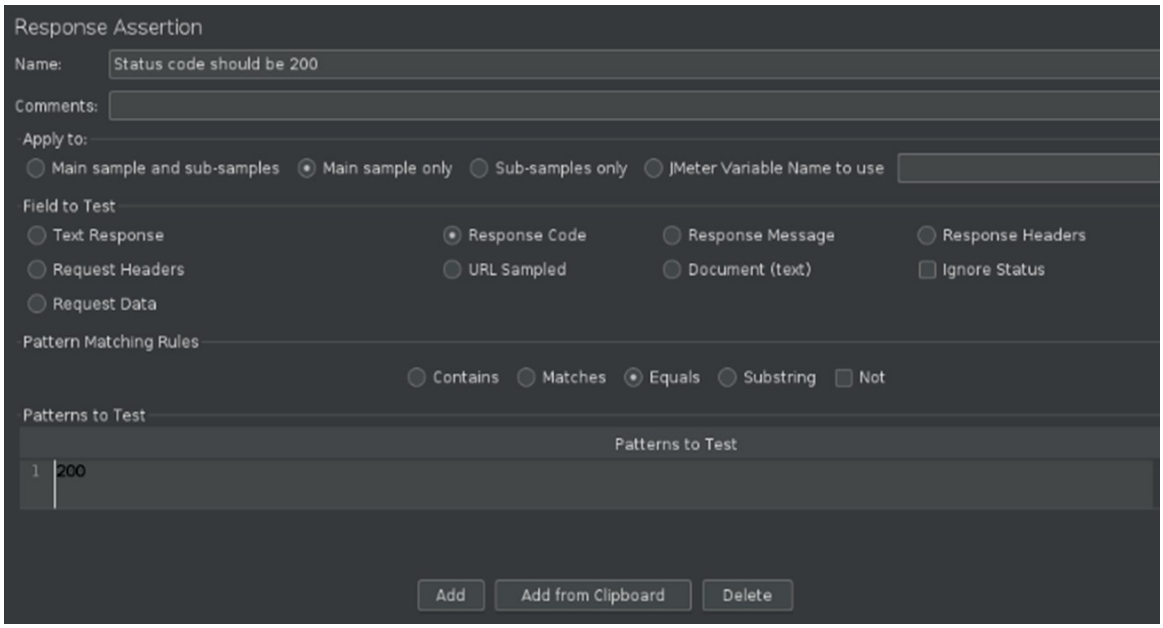


Figure 20 *Apache JMeter assertion response*

SN	Test Label	API Endpoint URL
1	Request http data	http://10.0.255.15:8090/api/v1/http
2	Request http data using limit	http://10.0.255.15:8090/api/v1/http?limit=100
3	Request http data using projection	http://10.0.255.15:8090/api/v1/http?projection=source_ip

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

4	Request http data using pagination	http://10.0.255.15:8090/api/v1/http?limit=1&offset=10
5	Request smtp data	http://10.0.255.15:8090/api/v1/smtp
6	Request smtp data using projection	http://10.0.255.15:8090/api/v1/smtp?projection=source_ip
7	Request smtp data using limit	http://10.0.255.15:8090/api/v1/smtp?limit=100
8	Request smtp data using pagination	http://10.0.255.15:8090/api/v1/smtp?limit=1&offset=10
9	Request ftp data	http://10.0.255.15:8090/api/v1/ftp
10	Request ftp data using projection	http://10.0.255.15:8090/api/v1/ftp?projection=source_ip
11	Request ftp data using pagination	http://10.0.255.15:8090/api/v1/ftp?limit=1&offset=10
12	Request ftp data using limit	http://10.0.255.15:8090/api/v1/ftp?limit=100
13	Request ssh data	http://10.0.255.15:8090/api/v1/ssh
14	Request ssh data using projection	http://10.0.255.15:8090/api/v1/ssh?projection=source_ip
15	Request ssh data using limit	http://10.0.255.15:8090/api/v1/ssh?limit=100
16	Request ssh data using pagination	http://10.0.255.15:8090/api/v1/ssh?limit=1&offset=10
17	Request mlid data for ftp without providing start/end date	http://10.0.255.15:8090/api/v1/mlid
18	Request mlid data for ftp should return 200	http://10.0.255.15:8090/api/v1/mlid?service=ftp&start-date=2019-05-25T06:23:56.325Z&end-date=2040-05-25T06:23:56.325Z
19	Request mlid data for ssh without providing start/end date	http://10.0.255.15:8090/api/v1/mlid
20	Request http data after failed request should not fail	http://10.0.255.15:8090/api/v1/http
21	Request mlid data for ssh should return 200	http://10.0.255.15:8090/api/v1/mlid?service=ssh&start-date=2019-05-25T06:23:56.325Z&end-date=2040-05-25T06:23:56.325Z
22	Request mlid data for smtp without providing start/end date	http://10.0.255.15:8090/api/v1/mlid
23	Request mlid data for smtp should return 200	http://10.0.255.15:8090/api/v1/mlid?service=smtp&start-date=2019-05-25T06:23:56.325Z&end-date=2040-05-25T06:23:56.325Z
24	Request mlid data for http without providing start/end date	http://10.0.255.15:8090/api/v1/mlid

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

25	Request mlid data for http should return 200	http://10.0.255.15:8090/api/v1/mlid?service=http&start-date=2019-05-25T06:23:56.325Z&end-date=2040-05-25T06:23:56.325Z

***Table 3** Tested API endpoints for HP API component*

The JMeter configuration file describing and configuring all the requests depicted in Table 3 is appended in the Annex III: JMeter configuration file

4.2 Evaluation results

The JMeter tool does not only enable us to validate the functional behaviour of each API endpoint individually, but it also allows us to perform stress tests on our system; for example our test scenarios aim to prove the system's reliability in case of repeated "Request http *data*" requests, "Request smtp *data* " requests, "Request ftp *data* " HTTP requests, etc. In addition, the overall performance of this system can be evaluated through this tool by measuring the response time to any HTTP request made.

Table 4 presents the results for each tested endpoint. The received *HTTP response codes* verified the availability of the tested API endpoints whereas the HTTP *Assertion success* column indicates the tested API endpoints conformance to their specifications. Finally, the *Latency* column demonstrates the capability of the system to handle fast the submitted requests (the majority of the responses were below 15 msec).

Note: the response codes mean: 200 indicates successful requests and 400 indicates that a request with the wrong format was performed. A complete list of the HTTP codes can be found in IANA's Hypertext Transfer Protocol (HTTP) Status Code Registry [15].

SN	Test Label	Response Code	HTTP Assertion success	Latency (ms)
1	Request http data	200	TRUE	212
2	Request http data using limit	200	TRUE	9
3	Request http data using projection	200	TRUE	4
4	Request http data using pagination	200	TRUE	3
5	Request smtp data	200	TRUE	13
6	Request smtp data using projection	200	TRUE	8
7	Request smtp data using limit	200	TRUE	15
8	Request smtp data using pagination	200	TRUE	5
9	Request ftp data	200	TRUE	3
10	Request ftp data using projection	200	TRUE	8
11	Request ftp data using pagination	200	TRUE	2
12	Request ftp data using limit	200	TRUE	2
13	Request ssh data	200	TRUE	12
14	Request ssh data using projection	200	TRUE	9
15	Request ssh data using limit	200	TRUE	13
16	Request ssh data using pagination	200	TRUE	2

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

17	Request mlid data for ftp without providing start/end date	400	TRUE	2
18	Request mlid data for ftp should return 200	200	TRUE	18
19	Request mlid data for ssh without providing start/end date	400	TRUE	1
20	Request data after failed request should not fail	200	TRUE	8
21	Request mlid data for ssh should return 200	200	TRUE	16
22	Request mlid data for smtp without providing start/end date	400	TRUE	2
23	Request mlid data for smtp should return 200	200	TRUE	20
24	Request mlid data for http without providing start/end date	400	TRUE	1
25	Request mlid data for http should return 200	200	TRUE	20

Table 4 *Results for each tested endpoint*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

5 Conclusions

In the first iteration of this task, we have implemented the first version of the SPHINX AI Honeypots. SPHINX Honeypots are not a solution for ensuring security, it is a good tool that supplements other security technologies in order to form an alternative active defence system. More specifically, SPHINX Honeypots aim to lure the attackers in using their provided services and learn from their attacks, in order to afterwards modify and deploy the necessary security controls that will address the detected attacks. To achieve this, SPHINX HPs emulate commonly used services/protocols to serve an attractive for exploitation system to the cyber attackers. Currently, a SPHINX HP consists of six components namely, the HP Core, the HP Message Queue, the HP Data Consumer, the HP Storage DB, the HP Data Processor and the HP API, and is able to support up to four emulated services (i.e. SSH, FTP, HTTP, SMTP). Apart from emulating common services to gather attack information from the intruders, SPHINX HPs also perform sophisticated algorithms in order to properly process the attack information and generate data in a format that AI algorithms can understand, manage and use to detect attacks. Additionally, SPHINX HPs support HTTP (synchronous) communication with other SPHINX components that can access the HP data via the REST API named HP API. Finally, the HP Dashboard was built to improve the usability and facilitate the deployment, maintenance and management of SPHINX HPs.

For the next iteration phase (M24-M32) we plan on further securing the HPs, fixing bugs, and also expanding the portfolio of emulated services/protocols. Furthermore, we aim to enhance HPs in the area of the communication with other SPHINX components, so that they can also support asynchronous communication through Kafka.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Annex I: **References**

[1] T. Mezquita, "Honeypot," *CyberHoot*, Feb. 21, 2020. https://cyberhoot.com/cybrary/honeypot/ (accessed Aug. 13, 2020).

[2] "What is a Honeypot | Honeynets, Spam Traps & more | Imperva," *Learning Center*. https://www.imperva.com/learn/application-security/honeypot-honeynet/ (accessed Aug. 13, 2020).

[3] M. Shukla and P. Verma, "Honeypot: Concepts, Types and Working," vol. 3, no. 4, p. 3, 2015.

[4] R. A. Grimes, *Honeypots for Windows*, 1st ed. edition. Berkeley, CA : New York: Apress, 2005.

[5] "What is a honeypot?," *www.kaspersky.com*, Aug. 11, 2020. https://www.kaspersky.com/resource-center/threats/what-is-a-honeypot (accessed Aug. 13, 2020).

[6] "Golang, The Go programming language," [Online]. Available: https://golang.org/.

[7] "Beego Framework," [Online]. Available: https://beego.me/.

[8] "Dockerclient, Golang package," [Online]. Available: https://godoc.org/github.com/docker/docker/client.

[9] "Go-sqlite3, Golang SQLite3 package," [Online]. Available: https://github.com/mattn/go-sqlite3.

[10] "SQLite, Lightweight database," [Online]. Available: https://www.sqlite.org/index.html.

[11] "NSL-KDD dataset," [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html.

[12] "Swagger, API Development for Everyone," [Online]. Available: https://swagger.io/.

[13] "Zynq UltraScale+ Device Technical Reference Manual," p. 1177, 2019.

[14] "Apache JMeter™," [Online]. Available: https://jmeter.apache.org/.

[15] "Hypertext Transfer Protocol (HTTP) Status Code Registry, IANA," [Online]. Available: https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Annex II: **HP API swagger file**

Below the description of the HP API using the swagger format is given. The swagger format adheres to the OpenAPI Specification and as such enables other entities (e.g. other machines) to be able to read and use the provided API. In more detail the following content provides information regarding the API endpoints, their parameters and expected responses when they are used. Further to that, the data models that are used for holding and representing the exchanged information are portrayed.

```
openapi: "3.0.0"
info:
 version: 1.0.0
 title: SPHINX
 contact:
  name: Manos Kounelakis
  email: mkounelakis@f-in.eu

servers:
 - url: http://localhost:8090
  description: Localhost API server

paths:
 /api/v1/mlid:
  get:
   tags:
    - MLID
   summary: Retrieve a list of all mlids
   parameters:
    - in: query
      name: service
      schema:
       type: string
      description: Name of service you want to retrieve mlids for . Must be one of (http,ssh,ftp,smtp)
      example: 'http'
      required: true
    - in: query
      name: start-date
      schema:
       type: string
      description: Base date used for filtering results (Must be in ISO 8601 format)
      example: 'ex. 2011-10-05T14:48:00.000Z'
      required: true
    - in: query
      name: end-date
      schema:
       type: string
      description: End date used for filtering results (Must be in ISO 8601 format)
      example: 'ex.2011-10-06T14:48:00.000Z'
      required: true
   responses:
    '200':
      description: All MLID data
      content:
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        application/json:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/MLID'
    '400':
      description: Missing service start date or end date
      content:
        application/json:
          schema:
            type: object
            properties:
              message :
                type: string
                description: Error message


/api/v1/ssh:
  get:
    tags:
      - SSH
    parameters:
      - in: query
        name: projection
        schema:
          type: string
        description: A comma seperated string containing only the fields you want to retrieve
        example: 'ex. type,sessionid'
      - in: query
        name: limit
        schema:
          type: number
        description: Max number of results to return
        example: 'ex. 20'
      - in: query
        name: sortBy
        schema:
          type: string
        description: Field to sort by results
      - in: query
        name: order
        schema:
          type: string
        description: sortBy order (asc for ascending ,desc for descending)
        example: 'ex. asc'
      - in: query
        name: offset
        schema:
          type: number
        description: Offset for retrieving results(skips results).Can be used in conjuction with 'limit' field
        example: 'ex. 10'
      - in: query
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
      name: filter
      schema:
        type: string
      description: comma separated filter conditions (must use the exact field names as the schema schema)
      example: 'ex. type:password-authentication,ssh_username:root'
    description: Retrieve ssh data
    responses:
     '200':
      description: All ssh data
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/SSH'

/api/v1/http:
  get:
    tags:
      - HTTP
    description: Retrieve http data
    parameters:
      - in: query
        name: projection
        schema:
          type: string
        description: A comma seperated string containing only the fields you want to retrieve
        example: 'ex. http_host,sessionid'
      - in: query
        name: limit
        schema:
          type: number
        description: Max number of results to return
        example: 'ex. 20'
      - in: query
        name: sortBy
        schema:
          type: string
        description: Field to sort by results
      - in: query
        name: order
        schema:
          type: string
        description: sortBy order (asc for ascending ,desc for descending)
        example: 'ex. asc'
      - in: query
        name: offset
        schema:
          type: number
        description: Offset for retrieving results(skips results).Can be used in conjuction with 'limit' field
        example: 'ex. 10'
      - in: query
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
      name: filter
      schema:
        type: string
      description: comma separated filter conditions (must use the exact field names as the schema schema)
      example: 'ex. http_method:GET,http_proto:HTTP/1.1'
    responses:
     '200':
      description: All Http data
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/HTTP'

/api/v1/ftp:
  get:
    tags:
      - FTP
    description: Retrieve ftp data
    parameters:
      - in: query
        name: projection
        schema:
          type: string
        description: A comma seperated string containing only the fields you want to retrieve
        example: 'ex. ftp_command'
      - in: query
        name: limit
        schema:
          type: number
        description: Max number of results to return
        example: 'ex. 20'
      - in: query
        name: sortBy
        schema:
          type: string
        description: Field to sort by results
      - in: query
        name: order
        schema:
          type: string
        description: sortBy order (asc for ascending ,desc for descending)
        example: 'ex. asc'
      - in: query
        name: offset
        schema:
          type: number
        description: Offset for retrieving results(skips results).Can be used in conjuction with 'limit' field
        example: 'ex. 10'
      - in: query
        name: filter
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
      schema:
        type: string
      description: comma separated filter conditions (must use the exact field names as the schema schema)
    responses:
     '200':
      description: All Http data
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/FTP'

/api/v1/smtp:
 get:
   tags:
    - SMTP
   description: Retrieve smtp data
   parameters:
    - in: query
      name: projection
      schema:
        type: string
      description: A comma seperated string containing only the fields you want to retrieve
      example: 'ex. smtp_line'
    - in: query
      name: limit
      schema:
        type: number
      description: Max number of results to return
      example: 'ex. 20'
    - in: query
      name: sortBy
      schema:
        type: string
      description: Field to sort by results
    - in: query
      name: order
      schema:
        type: string
      description: sortBy order (asc for ascending ,desc for descending)
      example: 'ex. asc'
    - in: query
      name: offset
      schema:
        type: number
      description: Offset for retrieving results(skips results).Can be used in conjuction with 'limit' field
      example: 'ex. 10'
    - in: query
      name: filter
      schema:
        type: string
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        description: comma separated filter conditions (must use the exact field names as the schema schema)
    responses:
      '200':
        description: All smtp data
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/SMTP'



components:
  schemas:
    MLID:
      title: MLID
      type: object
      properties:
        timestamp:
          type: number
          description: The timestamp of the last recorded action in the context of a session
        source_ip:
          type: string
          description: Source IP of attack
        destination_ip:
          type: string
          description: Destination IP of attack
        duration:
          type: number
          description: length (number of seconds) of the connection
        protocol_type:
          type: string
          description:  type of the protocol, e.g. tcp, udp, etc
        service:
          type: string
          description: network service on the destination, e.g., http, telnet, etc.
        flag:
          type: string
          description: normal or error status of the connection
        source_bytes:
          type: number
          description: number of data bytes from source to destination
        destination_bytes:
          type: number
          description: number of data bytes from destination to source
        land:
          type: number
          description: 1 if connection is from/to the same host/port; 0 otherwise
        wrong_fragment:
          type: number
          description: number of wrong fragments
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
urgent:
  type: number
  description: number of urgent packets
hot:
  type: number
  description: number of hot indicators
number_failed_logins:
  type: number
  description: number of failed login attempts
logged_in:
  type: number
  description: 1 if successfully logged in; 0 otherwise
num_compromised:
  type: number
  description: number of compromised conditions
root_shell:
  type: number
  description: 1 if root shell is obtained; 0 otherwise
su_attempted:
  type: number
  description: 1 if su root command attempted; 0 otherwise
num_root:
  type: number
  description: number of root accesses
num_file_creations:
  type: number
  description: number of file creation operations
num_shells:
  type: number
  description: number of shell prompts
num_access_files:
  type: number
  description: number of operations on access control files
num_outbound_cmds:
  type: number
  description: number of outbound commands in an ftp session
is_host_login:
  type: number
  description: 1 if the login belongs to the hot list; 0 otherwise
is_guest_login:
  type: number
  description: 1 if the login is a guest login; 0 otherwise
count:
  type: number
  description: number of connections to the same host as the current connection in the past two seconds
srv_count:
  type: number
  description: number of connections to the same service as the current connection in the past two seconds
serror_rate:
  type: number
  description: number of connections to the same host as the current connection in the past two seconds
srv_serror_rate:
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
      type: number
      description: '% of connections that have SYN errors'
    rerror_rate:
      type: number
      description: '% of connections that have REJ errors'
    srv_rerror_rate:
      type: number
      description: '% of connections that have REJ errors'
    same_srv_rate:
      type: number
      description: '% of connections to same services'
    diff_srv_rate:
      type: number
      description:  '% of connections to different services'
    srv_diff_host_rate:
      type: number
      description: '% of connections to different hosts'
    dst_host_count:
      type: number
      description: Number of connection to the same host to the destination host as the current connection in the
past 2 seconds
    dst_host_srv_count:
      type: number
      description: Number of connection from the same service to the destination host as the current connection in
the past 2 seconds
    dst_host_same_srv_rate:
      type: number
      description: '% of connections from the same services to the destination host'
    dst_host_diff_srv_rate:
      type: number
      description: '% of connections from the different services to the destination host'
    dst_host_same_src_port_rate:
      type: number
      description: '% of connections from the port services to the destination host'
    dst_host_srv_diff_host_rate:
      type: number
      description: '% of connections from the different hosts from the same service to the destination host'
    dst_host_serror_rate:
      type: number
      description: '% of connections that have SYN errors from thesame host to the destination host'
    dst_host_srv_serror_rate:
      type: number
      description: '% of connections that have SYN errors from the same service to the destination host'
    dst_host_rerror_rate:
      type: number
      description: '% of connections that have REJ errors from the same host to the destination host'
    dst_host_srv_rerror_rate:
      type: number
      description: '% of connections that have REJ errors from the same service to the destination host'

  SSH:
    type: object
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
  properties:
    date:
      type: string
      description: The full date of the incident
      example: January 8th 2020, 13:13:55.255
    destination_ip:
      type: string
      description: The IP of the destination of the packet (the Server's IP)
    destination_port:
      type: number
      description: The PORT of the destination of the packet (the Server's PORT)
    source_ip:
      type: string
      description: The IP of the source packet (the attack IP)
    source_port:
      type: number
      description: The PORT of the source packet (the attack PORT)
    ssh_password:
      type: string
      description: The password used in the attack
      example: 123456
    sessionid:
      type: string
      description: The session id created by the ssh honeypot
    ssh_username:
      type: string
      description: The username used in the attack
      example: root
    type:
      type: string
      description: The method of authentication used [password-authentication, ssh-certificate]
      example: password-authentication
  HTTP:
    type: object
    properties:
      date:
        type: string
        description: The full date of the incident
        example: January 8th 2020, 13:13:55.255
      destination_ip:
        type: string
        description: The IP of the destination of the packet (the Server's IP)
      destination_port:
        type: number
        description: The PORT of the destination of the packet (the Server's PORT)
      source_ip:
        type: string
        description: The IP of the source packet (the attack IP)
      source_port:
        type: number
        description: The PORT of the source packet (the attack PORT)
      http_header:
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        type: string
        description: All the lines used in the HTTP header of the HTTP Request
        example: 'http.header.accept:image/webp,image/apng,image/*,*/*;q=0.8 http.header.accept-encoding:gzip,
deflate, br http.header.accept-language:en-US,en;q=0.9,el-GR;q=0.8,el;q=0.7
http.header.connection:keep-alive http.header.referer:http://127.0.0.1:8080/ http.header.sec-fetch-mode:no-cors
http.header.sec-fetch-site:same-origin http.header.user-agent:Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/78.0.3904.70 Safari/537.36'
      http_host:
        type: string
        description: The virtual Host, used by the Honeypot
        example: 127.0.0.1:80
      http_method:
        type: string
        description: The method used in the HTTP Request
        example: GET
      http_proto:
        type: string
        description: The protocol used in the HTTP Request
        example: HTTP/1.1
      sessionid:
        type: string
        description: The session id created by the HTTP honeypot
      url:
        type: string
        description: The url used in the HTTP Request
      payload:
        type: string
        description: The total answer of the HTTP Response, made by the HTTP Honeypot
  FTP:
    type: object
    properties:
      date:
        type: string
        description: The full date of the incident
        example: January 8th 2020, 13:13:55.255
      destination_ip:
        type: string
        description: The IP of the destination of the packet (the Server's IP)
      destination_port:
        type: number
        description: The PORT of the destination of the packet (the Server's PORT)
      source_ip:
        type: string
        description: The IP of the source packet (the attack IP)
      source_port:
        type: number
        description: The PORT of the source packet (the attack PORT)
      ftp_command:
        type: string
        description: The command used in the FTP attack
        example:  PASS admin
      ftp_username:
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
      type: string
      description: Username used for ftp authentication
    ftp_password:
      type: string
      description: Password used for ftp authentication
    sessionid:
      type: string
      description: The session id created by the FTP honeypot
SMTP:
  type: object
  properties:
    date:
      type: string
      description: The full date of the incident
      example: January 8th 2020, 13:13:55.255
    destination_ip:
      type: string
      description: The IP of the destination of the packet (the Server's IP)
    destination_port:
      type: number
      description: The PORT of the destination of the packet (the Server's PORT)
    source_ip:
      type: string
      description: The IP of the source packet (the attack IP)
    source_port:
      type: number
      description: The PORT of the source packet (the attack PORT)
    smtp_line:
      type: string
      description: The command used in the SMTP attack
      example: AUTH PLAIN XXXXXXXX
    sessionid:
      type: string
      description: The session id created by the SMTP honeypot
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Annex III: **JMeter configuration file**

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.3">
 <hashTree>
  <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan" enabled="true">
   <stringProp name="TestPlan.comments"></stringProp>
   <boolProp name="TestPlan.functional_mode">false</boolProp>
   <boolProp name="TestPlan.tearDown_on_shutdown">true</boolProp>
   <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
   <elementProp name="TestPlan.user_defined_variables" elementType="Arguments" guiclass="ArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
    <collectionProp name="Arguments.arguments">
     <elementProp name="HOST" elementType="Argument">
      <stringProp name="Argument.name">HOST</stringProp>
      <stringProp name="Argument.value">localhost</stringProp>
      <stringProp name="Argument.metadata">=</stringProp>
     </elementProp>
     <elementProp name="PORT" elementType="Argument">
      <stringProp name="Argument.name">PORT</stringProp>
      <stringProp name="Argument.value">8090</stringProp>
      <stringProp name="Argument.metadata">=</stringProp>
     </elementProp>
    </collectionProp>
   </elementProp>
   <stringProp name="TestPlan.user_define_classpath"></stringProp>
  </TestPlan>
  <hashTree>
   <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Main group" enabled="true">
    <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
    <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
     <boolProp name="LoopController.continue_forever">false</boolProp>
     <stringProp name="LoopController.loops">1</stringProp>
    </elementProp>
    <stringProp name="ThreadGroup.num_threads">1</stringProp>
    <stringProp name="ThreadGroup.ramp_time">1</stringProp>
    <boolProp name="ThreadGroup.scheduler">false</boolProp>
    <stringProp name="ThreadGroup.duration"></stringProp>
    <stringProp name="ThreadGroup.delay"></stringProp>
    <boolProp name="ThreadGroup.same_user_on_next_iteration">true</boolProp>
   </ThreadGroup>
   <hashTree>
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request http
data" enabled="true">
     <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
      <collectionProp name="Arguments.arguments"/>
     </elementProp>
     <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
     <stringProp name="HTTPSampler.port">${PORT}</stringProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        <stringProp name="HTTPSampler.protocol">http</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/api/v1/http</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp name="HTTPSampler.connect_timeout"></stringProp>
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp name="Assertion.custom_message"></stringProp>
          <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
          <boolProp name="Assertion.assume_success">false</boolProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request http data
using limit" enabled="true">
        <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
          <collectionProp name="Arguments.arguments">
            <elementProp name="limit" elementType="HTTPArgument">
              <boolProp name="HTTPArgument.always_encode">false</boolProp>
              <stringProp name="Argument.value">100</stringProp>
              <stringProp name="Argument.metadata">=</stringProp>
              <boolProp name="HTTPArgument.use_equals">true</boolProp>
              <stringProp name="Argument.name">limit</stringProp>
            </elementProp>
          </collectionProp>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">http</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/api/v1/http</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp name="HTTPSampler.connect_timeout"></stringProp>
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
      </HTTPSamplerProxy>
      <hashTree>
       <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
        <collectionProp name="Asserion.test_strings">
         <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
       </ResponseAssertion>
       <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request http data
using projection" enabled="true">
       <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">
         <elementProp name="projection" elementType="HTTPArgument">
          <boolProp name="HTTPArgument.always_encode">false</boolProp>
          <stringProp name="Argument.value">source_ip</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
          <boolProp name="HTTPArgument.use_equals">true</boolProp>
          <stringProp name="Argument.name">projection</stringProp>
         </elementProp>
        </collectionProp>
       </elementProp>
       <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
       <stringProp name="HTTPSampler.port">${PORT}</stringProp>
       <stringProp name="HTTPSampler.protocol">http</stringProp>
       <stringProp name="HTTPSampler.contentEncoding"></stringProp>
       <stringProp name="HTTPSampler.path">/api/v1/http</stringProp>
       <stringProp name="HTTPSampler.method">GET</stringProp>
       <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
       <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
       <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
       <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
       <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
       <stringProp name="HTTPSampler.connect_timeout"></stringProp>
       <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
       <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
        <collectionProp name="Asserion.test_strings">
         <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
      </ResponseAssertion>
      <hashTree/>
    </hashTree>
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request http data using pagination" enabled="true">
      <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">
          <elementProp name="limit" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">1</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">limit</stringProp>
          </elementProp>
          <elementProp name="offset" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">10</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">offset</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
      <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
      <stringProp name="HTTPSampler.port">${PORT}</stringProp>
      <stringProp name="HTTPSampler.protocol">http</stringProp>
      <stringProp name="HTTPSampler.contentEncoding"></stringProp>
      <stringProp name="HTTPSampler.path">/api/v1/http</stringProp>
      <stringProp name="HTTPSampler.method">GET</stringProp>
      <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
      <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
      <stringProp name="HTTPSampler.connect_timeout"></stringProp>
      <stringProp name="HTTPSampler.response_timeout"></stringProp>
    </HTTPSamplerProxy>
    <hashTree>
      <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be 200" enabled="true">
        <collectionProp name="Asserion.test_strings">
          <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
      </ResponseAssertion>
      <hashTree/>
    </hashTree>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request smtp data" enabled="true">
      <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments"/>
      </elementProp>
      <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
      <stringProp name="HTTPSampler.port">${PORT}</stringProp>
      <stringProp name="HTTPSampler.protocol">http</stringProp>
      <stringProp name="HTTPSampler.contentEncoding"></stringProp>
      <stringProp name="HTTPSampler.path">/api/v1/smtp</stringProp>
      <stringProp name="HTTPSampler.method">GET</stringProp>
      <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
      <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
      <stringProp name="HTTPSampler.connect_timeout"></stringProp>
      <stringProp name="HTTPSampler.response_timeout"></stringProp>
    </HTTPSamplerProxy>
    <hashTree>
      <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be 200" enabled="true">
        <collectionProp name="Asserion.test_strings">
          <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
      </ResponseAssertion>
      <hashTree/>
    </hashTree>
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request smtp data using projection" enabled="true">
      <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">
          <elementProp name="projection" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">source_ip</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">projection</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
      <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
      <stringProp name="HTTPSampler.port">${PORT}</stringProp>
      <stringProp name="HTTPSampler.protocol">http</stringProp>
      <stringProp name="HTTPSampler.contentEncoding"></stringProp>
      <stringProp name="HTTPSampler.path">/api/v1/smtp</stringProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp name="HTTPSampler.connect_timeout"></stringProp>
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp name="Assertion.custom_message"></stringProp>
          <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
          <boolProp name="Assertion.assume_success">false</boolProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request smtp
data using limit" enabled="true">
        <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
          <collectionProp name="Arguments.arguments">
            <elementProp name="limit" elementType="HTTPArgument">
              <boolProp name="HTTPArgument.always_encode">false</boolProp>
              <stringProp name="Argument.value">100</stringProp>
              <stringProp name="Argument.metadata">=</stringProp>
              <boolProp name="HTTPArgument.use_equals">true</boolProp>
              <stringProp name="Argument.name">limit</stringProp>
            </elementProp>
          </collectionProp>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">http</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/api/v1/smtp</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp name="HTTPSampler.connect_timeout"></stringProp>
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
      <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
        <collectionProp name="Asserion.test_strings">
          <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
      </ResponseAssertion>
      <hashTree/>
    </hashTree>
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request smtp
data using pagination" enabled="true">
      <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">
          <elementProp name="limit" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">1</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">limit</stringProp>
          </elementProp>
          <elementProp name="offset" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">10</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">offset</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
      <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
      <stringProp name="HTTPSampler.port">${PORT}</stringProp>
      <stringProp name="HTTPSampler.protocol">http</stringProp>
      <stringProp name="HTTPSampler.contentEncoding"></stringProp>
      <stringProp name="HTTPSampler.path">/api/v1/smtp</stringProp>
      <stringProp name="HTTPSampler.method">GET</stringProp>
      <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
      <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
      <stringProp name="HTTPSampler.connect_timeout"></stringProp>
      <stringProp name="HTTPSampler.response_timeout"></stringProp>
    </HTTPSamplerProxy>
    <hashTree>
      <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
        <collectionProp name="Asserion.test_strings">
          <stringProp name="49586">200</stringProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
       </ResponseAssertion>
       <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request ftp data"
enabled="true">
       <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments"/>
       </elementProp>
       <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
       <stringProp name="HTTPSampler.port">${PORT}</stringProp>
       <stringProp name="HTTPSampler.protocol">http</stringProp>
       <stringProp name="HTTPSampler.contentEncoding"></stringProp>
       <stringProp name="HTTPSampler.path">/api/v1/ftp</stringProp>
       <stringProp name="HTTPSampler.method">GET</stringProp>
       <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
       <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
       <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
       <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
       <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
       <stringProp name="HTTPSampler.connect_timeout"></stringProp>
       <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
       <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
        <collectionProp name="Asserion.test_strings">
         <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
       </ResponseAssertion>
       <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request ftp data
using projection" enabled="true">
       <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">
         <elementProp name="projection" elementType="HTTPArgument">
          <boolProp name="HTTPArgument.always_encode">false</boolProp>
          <stringProp name="Argument.value">source_ip</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
          <boolProp name="HTTPArgument.use_equals">true</boolProp>
          <stringProp name="Argument.name">projection</stringProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
          </elementProp>
         </collectionProp>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">http</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/api/v1/ftp</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp name="HTTPSampler.connect_timeout"></stringProp>
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
       </HTTPSamplerProxy>
       <hashTree>
        <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
         <collectionProp name="Asserion.test_strings">
          <stringProp name="49586">200</stringProp>
         </collectionProp>
         <stringProp name="Assertion.custom_message"></stringProp>
         <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
         <boolProp name="Assertion.assume_success">false</boolProp>
         <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
       </hashTree>
       <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request ftp data
using pagination" enabled="true">
        <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
         <collectionProp name="Arguments.arguments">
          <elementProp name="limit" elementType="HTTPArgument">
           <boolProp name="HTTPArgument.always_encode">false</boolProp>
           <stringProp name="Argument.value">1</stringProp>
           <stringProp name="Argument.metadata">=</stringProp>
           <boolProp name="HTTPArgument.use_equals">true</boolProp>
           <stringProp name="Argument.name">limit</stringProp>
          </elementProp>
          <elementProp name="offset" elementType="HTTPArgument">
           <boolProp name="HTTPArgument.always_encode">false</boolProp>
           <stringProp name="Argument.value">10</stringProp>
           <stringProp name="Argument.metadata">=</stringProp>
           <boolProp name="HTTPArgument.use_equals">true</boolProp>
           <stringProp name="Argument.name">offset</stringProp>
          </elementProp>
         </collectionProp>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
      <stringProp name="HTTPSampler.port">${PORT}</stringProp>
      <stringProp name="HTTPSampler.protocol">http</stringProp>
      <stringProp name="HTTPSampler.contentEncoding"></stringProp>
      <stringProp name="HTTPSampler.path">/api/v1/ftp</stringProp>
      <stringProp name="HTTPSampler.method">GET</stringProp>
      <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
      <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
      <stringProp name="HTTPSampler.connect_timeout"></stringProp>
      <stringProp name="HTTPSampler.response_timeout"></stringProp>
    </HTTPSamplerProxy>
    <hashTree>
      <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
        <collectionProp name="Asserion.test_strings">
          <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
      </ResponseAssertion>
      <hashTree/>
    </hashTree>
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request ftp data
using limit" enabled="true">
      <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">
          <elementProp name="limit" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">100</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">limit</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
      <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
      <stringProp name="HTTPSampler.port">${PORT}</stringProp>
      <stringProp name="HTTPSampler.protocol">http</stringProp>
      <stringProp name="HTTPSampler.contentEncoding"></stringProp>
      <stringProp name="HTTPSampler.path">/api/v1/ftp</stringProp>
      <stringProp name="HTTPSampler.method">GET</stringProp>
      <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
      <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
      <stringProp name="HTTPSampler.connect_timeout"></stringProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp name="Assertion.custom_message"></stringProp>
          <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
          <boolProp name="Assertion.assume_success">false</boolProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request ssh data"
enabled="true">
        <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">http</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/api/v1/ssh</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp name="HTTPSampler.connect_timeout"></stringProp>
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp name="Assertion.custom_message"></stringProp>
          <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
          <boolProp name="Assertion.assume_success">false</boolProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request ssh data
using projection" enabled="true">
        <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        <collectionProp name="Arguments.arguments">
          <elementProp name="projection" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">source_ip</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">projection</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
      <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
      <stringProp name="HTTPSampler.port">${PORT}</stringProp>
      <stringProp name="HTTPSampler.protocol">http</stringProp>
      <stringProp name="HTTPSampler.contentEncoding"></stringProp>
      <stringProp name="HTTPSampler.path">/api/v1/ssh</stringProp>
      <stringProp name="HTTPSampler.method">GET</stringProp>
      <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
      <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
      <stringProp name="HTTPSampler.connect_timeout"></stringProp>
      <stringProp name="HTTPSampler.response_timeout"></stringProp>
    </HTTPSamplerProxy>
    <hashTree>
      <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
        <collectionProp name="Asserion.test_strings">
          <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
      </ResponseAssertion>
      <hashTree/>
    </hashTree>
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request ssh data
using limit" enabled="true">
      <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">
          <elementProp name="limit" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">100</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">limit</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
      <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">http</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/api/v1/ssh</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp name="HTTPSampler.connect_timeout"></stringProp>
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp name="Assertion.custom_message"></stringProp>
          <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
          <boolProp name="Assertion.assume_success">false</boolProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request ssh data
using pagination" enabled="true">
        <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
          <collectionProp name="Arguments.arguments">
            <elementProp name="limit" elementType="HTTPArgument">
              <boolProp name="HTTPArgument.always_encode">false</boolProp>
              <stringProp name="Argument.value">1</stringProp>
              <stringProp name="Argument.metadata">=</stringProp>
              <boolProp name="HTTPArgument.use_equals">true</boolProp>
              <stringProp name="Argument.name">limit</stringProp>
            </elementProp>
            <elementProp name="offset" elementType="HTTPArgument">
              <boolProp name="HTTPArgument.always_encode">false</boolProp>
              <stringProp name="Argument.value">10</stringProp>
              <stringProp name="Argument.metadata">=</stringProp>
              <boolProp name="HTTPArgument.use_equals">true</boolProp>
              <stringProp name="Argument.name">offset</stringProp>
            </elementProp>
          </collectionProp>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">http</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/api/v1/ssh</stringProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp name="HTTPSampler.connect_timeout"></stringProp>
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp name="Assertion.custom_message"></stringProp>
          <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
          <boolProp name="Assertion.assume_success">false</boolProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request mlid data
for ftp  without providing start/end date" enabled="true">
        <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">http</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/api/v1/mlid</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp name="HTTPSampler.connect_timeout"></stringProp>
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="51508">400</stringProp>
          </collectionProp>
          <stringProp name="Assertion.custom_message"></stringProp>
          <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
          <boolProp name="Assertion.assume_success">false</boolProp>
          <intProp name="Assertion.test_type">8</intProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
      </ResponseAssertion>
      <hashTree/>
    </hashTree>
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request mlid data for ftp  should return 200" enabled="true">
      <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">
          <elementProp name="service" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">true</boolProp>
            <stringProp name="Argument.value">ftp</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">service</stringProp>
          </elementProp>
          <elementProp name="start-date" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">2019-05-25T06:23:56.325Z</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">start-date</stringProp>
          </elementProp>
          <elementProp name="end-date" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">2040-05-25T06:23:56.325Z</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">end-date</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
      <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
      <stringProp name="HTTPSampler.port">${PORT}</stringProp>
      <stringProp name="HTTPSampler.protocol">http</stringProp>
      <stringProp name="HTTPSampler.contentEncoding"></stringProp>
      <stringProp name="HTTPSampler.path">/api/v1/mlid</stringProp>
      <stringProp name="HTTPSampler.method">GET</stringProp>
      <boolProp name="HTTPSampler.follow_redirects">false</boolProp>
      <boolProp name="HTTPSampler.auto_redirects">true</boolProp>
      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
      <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
      <stringProp name="HTTPSampler.connect_timeout"></stringProp>
      <stringProp name="HTTPSampler.response_timeout"></stringProp>
    </HTTPSamplerProxy>
    <hashTree>
      <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be 200" enabled="true">
        <collectionProp name="Asserion.test_strings">
          <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
       </ResponseAssertion>
       <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request mlid data for ssh  without providing start/end date" enabled="true">
       <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments"/>
       </elementProp>
       <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
       <stringProp name="HTTPSampler.port">${PORT}</stringProp>
       <stringProp name="HTTPSampler.protocol">http</stringProp>
       <stringProp name="HTTPSampler.contentEncoding"></stringProp>
       <stringProp name="HTTPSampler.path">/api/v1/mlid</stringProp>
       <stringProp name="HTTPSampler.method">GET</stringProp>
       <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
       <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
       <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
       <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
       <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
       <stringProp name="HTTPSampler.connect_timeout"></stringProp>
       <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
       <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be 200" enabled="true">
        <collectionProp name="Asserion.test_strings">
         <stringProp name="51508">400</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
       </ResponseAssertion>
       <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request data after failed request should not fail" enabled="true">
       <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments"/>
       </elementProp>
       <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
       <stringProp name="HTTPSampler.port">${PORT}</stringProp>
       <stringProp name="HTTPSampler.protocol">http</stringProp>
       <stringProp name="HTTPSampler.contentEncoding"></stringProp>
       <stringProp name="HTTPSampler.path">/api/v1/http</stringProp>
       <stringProp name="HTTPSampler.method">GET</stringProp>
       <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
      <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
      <stringProp name="HTTPSampler.connect_timeout"></stringProp>
      <stringProp name="HTTPSampler.response_timeout"></stringProp>
    </HTTPSamplerProxy>
    <hashTree>
      <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Response Assertion"
enabled="true">
        <collectionProp name="Asserion.test_strings">
          <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
      </ResponseAssertion>
      <hashTree/>
    </hashTree>
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request mlid data
for ssh  should return 200" enabled="true">
      <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">
          <elementProp name="service" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">true</boolProp>
            <stringProp name="Argument.value">ssh</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">service</stringProp>
          </elementProp>
          <elementProp name="start-date" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">2019-05-25T06:23:56.325Z</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">start-date</stringProp>
          </elementProp>
          <elementProp name="end-date" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">2040-05-25T06:23:56.325Z</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">end-date</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
      <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
      <stringProp name="HTTPSampler.port">${PORT}</stringProp>
      <stringProp name="HTTPSampler.protocol">http</stringProp>
      <stringProp name="HTTPSampler.contentEncoding"></stringProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        <stringProp name="HTTPSampler.path">/api/v1/mlid</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp name="HTTPSampler.connect_timeout"></stringProp>
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp name="Assertion.custom_message"></stringProp>
          <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
          <boolProp name="Assertion.assume_success">false</boolProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request mlid data
for smtp  without providing start/end date" enabled="true">
        <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">http</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/api/v1/mlid</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp name="HTTPSampler.connect_timeout"></stringProp>
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="51508">400</stringProp>
          </collectionProp>
          <stringProp name="Assertion.custom_message"></stringProp>
          <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
          <boolProp name="Assertion.assume_success">false</boolProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        <intProp name="Assertion.test_type">8</intProp>
       </ResponseAssertion>
       <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request mlid data for smtp  should return 200" enabled="true">
       <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">
         <elementProp name="service" elementType="HTTPArgument">
          <boolProp name="HTTPArgument.always_encode">true</boolProp>
          <stringProp name="Argument.value">smtp</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
          <boolProp name="HTTPArgument.use_equals">true</boolProp>
          <stringProp name="Argument.name">service</stringProp>
         </elementProp>
         <elementProp name="start-date" elementType="HTTPArgument">
          <boolProp name="HTTPArgument.always_encode">false</boolProp>
          <stringProp name="Argument.value">2019-05-25T06:23:56.325Z</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
          <boolProp name="HTTPArgument.use_equals">true</boolProp>
          <stringProp name="Argument.name">start-date</stringProp>
         </elementProp>
         <elementProp name="end-date" elementType="HTTPArgument">
          <boolProp name="HTTPArgument.always_encode">false</boolProp>
          <stringProp name="Argument.value">2040-05-25T06:23:56.325Z</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
          <boolProp name="HTTPArgument.use_equals">true</boolProp>
          <stringProp name="Argument.name">end-date</stringProp>
         </elementProp>
        </collectionProp>
       </elementProp>
       <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
       <stringProp name="HTTPSampler.port">${PORT}</stringProp>
       <stringProp name="HTTPSampler.protocol">http</stringProp>
       <stringProp name="HTTPSampler.contentEncoding"></stringProp>
       <stringProp name="HTTPSampler.path">/api/v1/mlid</stringProp>
       <stringProp name="HTTPSampler.method">GET</stringProp>
       <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
       <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
       <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
       <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
       <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
       <stringProp name="HTTPSampler.connect_timeout"></stringProp>
       <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
       <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be 200" enabled="true">
        <collectionProp name="Asserion.test_strings">
         <stringProp name="49586">200</stringProp>
        </collectionProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
       </ResponseAssertion>
       <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request mlid data for http  without providing start/end date" enabled="true">
       <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments"/>
       </elementProp>
       <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
       <stringProp name="HTTPSampler.port">${PORT}</stringProp>
       <stringProp name="HTTPSampler.protocol">http</stringProp>
       <stringProp name="HTTPSampler.contentEncoding"></stringProp>
       <stringProp name="HTTPSampler.path">/api/v1/mlid</stringProp>
       <stringProp name="HTTPSampler.method">GET</stringProp>
       <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
       <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
       <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
       <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
       <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
       <stringProp name="HTTPSampler.connect_timeout"></stringProp>
       <stringProp name="HTTPSampler.response_timeout"></stringProp>
      </HTTPSamplerProxy>
      <hashTree>
       <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be 200" enabled="true">
        <collectionProp name="Asserion.test_strings">
         <stringProp name="51508">400</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
       </ResponseAssertion>
       <hashTree/>
      </hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Request mlid data for http  should return 200" enabled="true">
       <elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">
         <elementProp name="service" elementType="HTTPArgument">
          <boolProp name="HTTPArgument.always_encode">true</boolProp>
          <stringProp name="Argument.value">http</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
          <boolProp name="HTTPArgument.use_equals">true</boolProp>
          <stringProp name="Argument.name">service</stringProp>
         </elementProp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
          <elementProp name="start-date" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">2019-05-25T06:23:56.325Z</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">start-date</stringProp>
          </elementProp>
          <elementProp name="end-date" elementType="HTTPArgument">
            <boolProp name="HTTPArgument.always_encode">false</boolProp>
            <stringProp name="Argument.value">2040-05-25T06:23:56.325Z</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <boolProp name="HTTPArgument.use_equals">true</boolProp>
            <stringProp name="Argument.name">end-date</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
      <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
      <stringProp name="HTTPSampler.port">${PORT}</stringProp>
      <stringProp name="HTTPSampler.protocol">http</stringProp>
      <stringProp name="HTTPSampler.contentEncoding"></stringProp>
      <stringProp name="HTTPSampler.path">/api/v1/mlid</stringProp>
      <stringProp name="HTTPSampler.method">GET</stringProp>
      <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
      <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
      <stringProp name="HTTPSampler.connect_timeout"></stringProp>
      <stringProp name="HTTPSampler.response_timeout"></stringProp>
    </HTTPSamplerProxy>
    <hashTree>
      <ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Status code should be
200" enabled="true">
        <collectionProp name="Asserion.test_strings">
          <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
      </ResponseAssertion>
      <hashTree/>
    </hashTree>
    <ResultCollector guiclass="ViewResultsFullVisualizer" testclass="ResultCollector" testname="View Results Tree"
enabled="true">
      <boolProp name="ResultCollector.error_logging">false</boolProp>
      <objProp>
        <name>saveConfig</name>
        <value class="SampleSaveConfiguration">
          <time>false</time>
          <latency>true</latency>
          <timestamp>true</timestamp>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
          <success>true</success>
          <label>true</label>
          <code>true</code>
          <message>true</message>
          <threadName>false</threadName>
          <dataType>true</dataType>
          <encoding>false</encoding>
          <assertions>true</assertions>
          <subresults>false</subresults>
          <responseData>false</responseData>
          <samplerData>false</samplerData>
          <xml>false</xml>
          <fieldNames>true</fieldNames>
          <responseHeaders>false</responseHeaders>
          <requestHeaders>false</requestHeaders>
          <responseDataOnError>false</responseDataOnError>
          <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
          <assertionsResultsToSave>0</assertionsResultsToSave>
          <url>true</url>
        </value>
       </objProp>
       <stringProp name="filename">/home/mkcodergr/Documents/fint
documents/testplans/ai_honeypot/results.csv</stringProp>
      </ResultCollector>
      <hashTree/>
    </hashTree>
  </hashTree>
 </hashTree>
</jmeterTestPlan>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.