D5.3 SPHINX Security Incident / Attack Simulator v1

WP5 – Analysis & Decision Making

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	**SPHINX D5.3 - Security Incident - Attack Simulator v1**		
Work Package	WP5 - Analysis and Decision Making		
Date of Delivery	**Contractual** M22	**Actual**	22
Nature	R - Report	**Dissemination Level**	P - Public
Lead Beneficiary	NTUA		
Responsible Author	Michael Kontoulis	**Email**	mkontoulis@epu.ntua.gr
		Phone	
Reviewer(s):	Ilias Trochidis [ViLabs], Yannis Nikoloudakis [HMU]		
Keywords	Simulation, cyber-attack vectors		

Document History

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.

Version	Issue Date	Stage	Changes	Contributor
0.1	16/09/2020	Draft	Indicative ToC	Sotiris Pelekis (NTUA)
0.2	19/10/2020	Draft	Finalised ToC	Michael Kontoulis (NTUA) , Sotiris Pelekis (NTUA)
0.3	23/9/2020	Draft	Content Update	Marco Manso (PDMFC)
0.4	05/10/2020	Draft	Content Update	Waqar Asif (TEC)
0..5	09/10/2020	Draft	Content Update	Radu Popescu (SIMAVI)
0.6	13/10/2020	Draft	Content Update	Michael Kontoulis (NTUA), Sotiris Pelekis (NTUA)
0.7	15/10/2020	Draft	Content Consolidation and submitted for internal review	Michael Kontoulis (NTUA), Sotiris Pelekis (NTUA)
0.71	18/10/2020	Draft	Internal Review 1	Ilias Trochidis (ViLabs)
0.72	27/10/2020	Draft	Internal Review 2	Yannis Nikoloudakis (HMU)
0.8	29/10/2020	Pre – Final	Incorporated Reviewers Comments	Michael Kontoulis (NTUA), Sotiris Pelekis (NTUA)
0.9	30/10/2020	Pre - Final	Quality Control	George Doukas (NTUA)
1.00	30/10/2020	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

The Security Incident / Attack Simulator constitutes one part of the Attack and Behaviours Simulators (ABS) component of SPHINX and is responsible for the simulation of the real-life cyber-attacks, described in the use cases of SPHINX. The purpose of this simulation is to validate the developed functions and evaluate the effectiveness of the other SPHINX components, by providing an easy way to reproduce these attacks.

This document describes in detail the state-of-the-art of cyber-attack testing, the purpose and the methodology for the implementation of the Security Incident / Attack Simulator component, and its relation with the other relative SPHINX deliverables and tasks. In addition, an overview of the currently selected tools that shall be used for the simulation processes is provided.

The next iteration of this deliverable D5.8: Security Incident/Attack Simulator v2, shall incorporate refinements and updates of the Attack Simulator component and the actual implementation set-ups of the selected cyber-attacks.

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.

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 Abbreviations

OSINT	Open-source intelligence
ABS	Attack and Behaviour Simulator
NFV	Network Function Virtualisation
C&C	Command and Control
C2C	Client to Client
IDS	Intrusion Detection System
MFA	Multi Factor Authentication
DoS	Denial of Service
DDoS	Distributed Denial of Service
IT	Information Technology
SNMP	Simple network management protocol
IP	Internet protocol address
ICMP	Internet control message protocol
TCP	Transmission Control Protocol
SMTP	Simple Mail Transfer Protocol
FTP	File Transfer Protocol
DNS	Domain Name Service
USB	Universal Serial Bus
ID	Identifier

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 and Scope

This document reports on the work and research conducted throughout the Task 5.3 of the SPHINX project, it provides an overview of the current state of security testing and explains the chosen methodology with which the simulation of cyber-attacks shall be performed within SPHINX.

With every passing day the need for protecting networks from cyber-attacks becomes clearer and clearer, even to the users most unfamiliar with the IT environments. In particular, cyber-attacks in critical infrastructures, such as hospitals, become highly publicised since they can even result to deaths,. [1] The potential impact from a cyber-attack in such health-care environments, highlights the necessity for the development of security systems, tailored to the specific requirements of these environments. The SPHINX project aims to create an additional defence layer that will work in tandem with existing hospital infrastructures and cyber-security defences, in order to provide foresight on potential attacks and ways to mitigate them, should they be successfully realised.

Obviously, SPHINX components cannot be tested in the actual real network of the pilots, since they are critical systems currently used in a production environment. As such, testing needs to be performed in other simulated environments that imitate, as realistically as possible, the original ones. This replication takes place in various other SPHINX modules as explained in the following sections of this document.

Task 5.3 is responsible for creating realistic cyber-attacks that will be used in those emulated environments in order to assess their efficacy. In this document, we will:

- Explain the different types of security testing and some useful tools for conducting them.
- Describe the nature of cyber-attacks, their phases and the various vectors of attacks used.
- Provide an initial approach of what cyber-attack schemes are needed, to test selected SPHINX use cases.

At the time of authorship of this document, the experimentation environment is not yet finalised and therefore the actual implementation of the selected cyber-attacks is not analysed in detail. The majority of such attacks depend heavily on the architecture of the testing environment, and thus cannot be presented separately. Consequently, this deliverable focuses mainly on the methodology, and less on the actual implementation, which will be covered in the next iteration of this deliverable which will include the demonstrator.

1.2 Structure of the deliverable

This document is structured as follows: Section 1 presents the purpose and the scope of the SPHINX Behaviour Emulation/ Experimentation environment and the relevant submodules of the respective ABS component, as well as its relation to other tasks; Section 2 presents an overview of security testing, the different ways it can be implemented and a short description of their strengths and their differences compared to our approach and a detailed overview of the lifecycle of a cyberattack and the various existing attack vectors; Section 3, describes which SPHINX modules are assessed by the ABS component and how; Section 4, contains the design principles of the security incident attack simulator, its architecture and its current state.

[1] https://www.nytimes.com/2020/09/18/world/europe/cyber-attack-germany-ransomeware-death.html

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.

1.3 Relation to other WPs and Tasks

This document, along with T5.3, is intrinsically linked to WP3 and WP4, as the majority of the components presented in them, are expected to get tested inside emulated environment provided by the ABS component. More specifically, it is tightly related to the tasks that partake in network inspections, such as Data Traffic Monitoring and Anomaly Detection (T4.1) and Homomorphic Encryption (T4.6), as they are the SPHINX components that should primarily be tested by the attack simulator. Furthermore, D5.3 is also linked to the SPHINX Sandbox component of T4.2, as it is able to provide the required isolation properties for the safe implementation of the simulations. Finally, the work described in this deliverable takes advantage of the foundations established in T5.4, whose technical outcomes also constitute submodules of the ABS component, which constitutes one of the modules on which cyber-attacks can be run against. This document is also connected to the SIEM of task T4.5, which is also able to detect attacks on the system. Finally, it is linked to the SPHINX use cases of the deliverables D2.4 and D2.7 which guide the focus of the testing.

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 Overview of Security Testing and Cyber-Attacks

2.1 Security Testing Methodology

The usual procedure used to ensure that a feature or a module works correctly usually involves checking that a specific task outputs the expected results. Such kind of testing is not sufficient for evaluating the integrity of a system against potential attackers. Attackers utilise cracks in the system, things that inherently go unnoticed by their creators. This includes using both unknown vulnerabilities and intentionally misusing legitimate features in ways the creators of the system cannot anticipate. This poses an inherent problem with security testing because it makes the conventional testing techniques much less effective. While it is still useful and can definitely protect against the most common misuses and architectural errors, new forms of testing are needed to cover the holes that conventional testing leaves. This testing aims to examine the behaviour of the system under conditions that are similar to actual attacks and observe the results. The intensity and exact purpose of these simulated attacks depend heavily on the type of security testing that is being utilised.

During security testing, testers take on the role of attackers and try to replicate the steps real attackers would take in order to penetrate the system. Merely asking attackers to gain access to the system, or steal information is not enough. The task should correspond to what exactly we want to test, whether it is reading encrypted files, shutting down live systems, impersonating clients etc. Another common purpose of a security testing is to determine how easy it is to acquire access to a system.

Another factor that needs to be determined before starting a test, is the amount of information given to the testers. In addition, it cannot be assumed that an attacker will have minimal information about the systems.. As such it is important to know beforehand what information the testing teams needs to have. A common solution is to simply give as much information as possible to the testing team, however that heavily depends on the kind of security test that will be conducted.

Similarly, depending on the kind of testing, access can and should be given to the testing team beforehand, if needed. This can include access to both the physical testing site or even a network. Whether access should be given at all, will also depend on what exactly is the purpose of the test.

Another choice to be made, is the amount of resources the attackers should have access to. This is correlated with the type of attack the administrators want to test their system against. If they anticipate that their system will only be a target for inexperienced attackers who will not have many resources or time and will only have access to the most common resources, then the testers need to stay within such limited parameters. If, on the other hand, testing needs to encompass any and all kinds of attackers, testers need more time and more resources in order to cover all the different scenarios.

This kind of testing produces some issues, including ethical concerns. That is why it is imperative that the testing team and the administrators, have established clear rules about what is allowed and what is not. This can be an especially big issue, concerning social engineering attacks. Additionally, security testing, in general, is often employed at improper stages of the developmental lifecycle, hampering its effectiveness. Implementing security testing too late in the development lifecycle of a system, may reveal fundamental inconsistencies within the system itself or/and its design, which may affect critically the implementation of the entire system. That is why this kind of security testing should not be held as a panacea and should not be the only measure in place to ensure the security of a system.

2.1.1 Penetration Testing

Penetration testing is one the most common forms of security testing employed in order to ensure the security of a system or application. It is done with the full knowledge of those responsible of the systems with the testers

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.

being in close communication about their activities. This is done both through actual verbal or written communication and through the manner the testing is concluded. Penetration testing, usually, is done as "loudly" as possible, without regarding whether the testers actions make them visible to the systems administrators, since they already know about it [1]. Although the people responsible for the security of the system, often called blue team, should be aware of most of the actions the testers take, their response, no matter if it aims to respond or mitigate their actions, isn't relevant to penetration testing. Their job is to merely observe the state of the system during the testing, try to ensure that testing does not go out of scope and finally give the ok for the attacker to conduct some more impactful attacks.[2]

Penetration testing often has a particular scope with specific parameters aiming usually to find specific networks or applications that are vulnerable to attackers in some way.

2.1.2 Red Teaming

Red teaming is in many ways similar to penetration regarding its scope and objectives, however it differs in some major ways. Whereas in penetration testing the focus is mainly on the system itself and its vulnerabilities, red teaming aims to test the response of the organisation as a whole. During a red team exercise, those responsible for the system, the blue team, may not even be aware of the exercise happening, or they will have as little information as possible. In addition, the methods and attacks employed by the red team, will aim to be identical to those used by real life attacks. That means, they will focus a lot more on not being detected and trying to find gaps in the general defence strategy of the organisation. On the other hand, the blue team will respond exactly as it would if a real attack took place, since they might not even be able to tell the difference [3].

After the exercise the blue team should report any indicators of compromise to the read team in order to create a fleshed-out report. Finally, the red team will share its tactics and techniques along with a complete timeline with all the events and will explain the techniques of defending against such tactics in the future.

2.1.3 Purple Teaming

The process of purple teaming involves both the aforementioned red teams and blue teams working in close collaboration and knowledge transfer throughout the whole period of testing, instead of only communicating at the end [4].

Purple teaming can be conducted as an one off kind of testing with specific targets more similar to red team exercises or to view purple teaming as a conceptual framework, where cyber-security issues are constantly tested and solved, potentially solving the issue of using security testing only on the end of a systems development lifecycle. Additionally, research on the opinions of security professionals has shown that purple teaming is a generally viewed positively but requires the participation of both experienced red team personnel and a strong collaborative effort between the managerial personnel of an organization.

2.1.4 Bug Hunting

Another method of conducting security testing in a more ad hoc manner, which has taken off in the last years is bug hunting. The bug bounty programs, can be run either internally or managed by third parties. These programs offer monetary rewards, in exchange for detailed reports on issues, bugs or vulnerabilities. The rewards are lucrative enough that many professionals, choose to take up bug hunting as their occupation. Programs like that have an edge over other more conventional methods since they attract a large number of participants each with different knowledge and skillset, which results in high numbers of success. That is probably why organisations that host or participate in such programs, do not only include small companies, but also huge technology companies and even federal agencies. [5]

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.

Unfortunately, they also have certain issues as research shows. Lack of trust has proven to be a reoccurring problem that security professionals cite as a major issue. They do not want to report specific bug, for fear of reprisal and getting tangled up in lengthy legal battles. Such, bugs may include those that are often found to be out of scope of certain programs. In the case of reporting, issues also appear when multiple participants report the same bug, leading to valuable effort being wasted since only the first one is rewarded usually, but there are steps organisation can take to mitigate such instances.

Additionally, some organisations suffer from similar trust issues themselves and prefer to participate and organise private bug bounty programs. This blurs the difference between a private bug bounty program and a traditional penetration testing.

Another issue is that experienced security professionals, tend to aggregate their effort to those programs with the highest monetary compensation, leaving the smaller companies with young professionals who are still learning their way around the trade, potentially lowering the effectiveness of such a system.

2.1.5 Automated Red Teaming

Extensive research has been done concerning the automation of the aforementioned Red Teaming process. Caldera[2] is an intelligent framework that can be used in order to automate the most basic, routine attacks. This specific framework does not mean to replace traditional cyber-security techniques but to supplement them by simplifying the most common and resource intensive tasks. This particular framework depends on two different systems the Virtual Red Team System and contains the necessary tools in order to simulate an adversary and the Logic for Advanced Virtual Adversaries, which forms the logical model that chooses the next actions.

2.1.6 Followed approach

The ABS implementation is planned to loosely follow many of the previously methods of security testing. The main differences arise from the fact that usually security testing targets the defence of the system itself however in the purpose of this task is to create valid chains of evidence through the use of actual cyber-attacks. As such, there is no intention of using the exact same means that are deployed in the context of common security testing. Detailed and technical specifications about out approach can be found in chapter 4.2 of this deliverable.

Of course, some of the selected tools are similar to the ones used in traditional security testing, such as port scanning tools (e.g. nmap) or malicious code (e.g. worms), but ABS implementation shall also include some additional techniques. These techniques shall take advantage of the traffic generation tools used and developed by the behaviour simulator to either simulate the behaviour of an illegitimate user using a legitimate account or to simulate the behaviour of malicious code, without actually using the malicious code itself. In addition, many of the scenarios that the attack simulator will orchestrate, will be designed in such a way that the attack shall always succeed or always fail.

2.2 Usual phases and methods of attacks

Cyber-attacks can and do take many forms amid different steps in order to reach their eventual goals. While there exists a general consensus on the various ways each type of cyber-attack can manifest, the exact explanation of how or what happened does not exist. Every case is unique since every attack focuses on different things and follows different trails. This diversity renders the whole process of understanding difficult since different experts interpretate the same things in different ways. The following subsections provide a generic and clear analysis that outlines the usual process of a cyber-attack [6].

[2] https://github.com/mitre/caldera

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.

2.2.1 Survey and enumeration

Enumeration is often the first step an attacker (real or simulated) will take when targeting a system. It involves gathering any and all available information about the system, technical information, about its users and administrators. Enumeration can be split in two different distinct forms. In Passive enumeration - where there is minimal, if any, interaction with the target system - the information is mainly gathered from an outside source while in the case of Active enumeration the information is acquired by interacting with the target system.

Passive

An important part of passive enumeration depends on Open-source intelligence (OSINT) [7]. OSINT depends on data gathered from publicly available sources. Depending on the target system these data can come from the following sources:

- Traditional Media

  This can include newspapers, television and radio.

- Internet

  This type includes blogs, forums, social media and other online publications.

- Public Government Data

  Publicly available reports, hearings, and websites.

- Academic and professional publications

  Publications from journal, conference hearings, academic papers, and dissertations.

- Commercial Data

  Assessment from private companies, private datasets, and even commercial imagery. Specifically information about companies' websites, employees full names, e-mails, compromised accounts, roles and utilised software, can be found using such tools like haveibeenpwned[3], wappalyzer[4], theHarvester[5] and WhatWeb[6].

- Other grey literature

  Other forms of grey literature such as whitepapers, technical reports or patents.

While not all those sources will always be available or even useful to attackers, the possibility of useful information being found always exists. For instance, Satellite imagery can offer valuable information about the physical space of an organisation to potential attackers. Moreover, information from social media, public or private dataset with stolen credential can also be used to breach passwords. Finally, technical reports also provide insight to potential vulnerabilities, especially those regarding older non upgraded systems.

Active

Active enumeration on the other hand is focused more on specific information. That is, information about the system that imply the existence of one or more potential attack vectors, which can then be used to actually exploit the system. An important method for active enumeration is port scanning. There are many different tools that allow the user to automate the port scanning progress, each with different output. Port scanning is

[3] https://haveibeenpwned.com/
[4] https://www.wappalyzer.com/
[5] https://github.com/laramies/theHarvester
[6] https://github.com/urbanadventurer/WhatWeb

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.

done significantly different in the context of a penetration test and a real red team attack. Some of the information active enumeration is interested in finding is the following:

- Usernames, Group names
- Hostnames
- IP tables and routing tables
- Open ports and network services
- Hidden directories
- Service settings and Audit configurations
- Application and banners
- SNMP and DNS Details
- Ciphers
- Other vulnerabilities

Advanced scanners, can even with the help of a basic scan, produce extensive reports with many possible existing vulnerabilities, if they do exist.

There is a large number of different steps, defenders can take to counter and disrupt scans, but care must be taken since many of those can harm the legitimate use of the network and greatly increase its upkeep and maintenance.

Some representative and popular scanning tools that are both used for attack (active web and network scanning) and defence (penetration testing and vulnerability assessments) purposes are NMAP[7], Nessus[8] and OpenVAS[9] vulnerability scanners, Nikto[10] and DirBuster[11] tool which comes with the Kali Linux distribution of Linux.

2.2.2 Delivery

The delivery step involves the method of transferring and installing the malware to the target system. Depending on the attack the delivery, can differ significantly. The most common cases refer to:

- sending an email with a malware infested attachment and waiting for someone to open it.
- using infected usb sticks.
- using stolen credentials or credentials surfing and gaining some form of “legitimate” access to the system.
- utilising vulnerabilities found in the system during the previous steps of attack

The delivery step is not applicable to some attacks, such as DoS and DDoS [8] attacks, since the producing exorbitant amount of traffic is in and by itself the attack.

2.2.3 Exploitation

At exploitation step, the attackers have already acquired some kind of access to the system. Using existing privileges, the attackers search for vulnerabilities in an effort to escalate the attack and subsequently make changes to the operation of the system, steal information or simply disrupting the operation of the system. In most of the cases, acquiring access to a specific system with a specific role is not enough, so they focus on upgrading their roles or gain access to every other connected system.

[7] https://nmap.org/
[8] https://www.tenable.com/products/nessus/nessus-essentials
[9] https://www.openvas.org/
[10] https://github.com/sullo/nikto
[11] https://tools.kali.org/web-applications/dirbuster

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.

2.2.4 Post Exploitation

At this step, attackers after having completed their immediate objectives either they end the attack or they try to create a backdoor to the system which shall enable them to regain access at any time. In the latter case, they try to erase as many traces of the attack as possible by following techniques such as erasing logs, spoofing information etc. in order to hinder the forensic process and cover their tracks.

2.3 Varieties of Attack vectors

2.3.1 Social Engineering

Social engineering is a non-technical strategy that cyber-attackers use to gain access to information. Attackers heavily rely on human interaction which is exploited by tricking people into breaking standard security practices. The success of such attacks depends on the ability of the attacker to manipulate a victim into performing certain actions that allow her/him to access confidential information. Currently, social engineering attacks are recognized as one of the greatest security threats faced by organizations. These attacks differ from conventional hacking tactics in a way that they do not necessarily involve compromising any software or system, rather they allow attackers to get legitimate and authorized access to confidential information. Social engineers tend to appeal to vanity, authority and greed. They are known to rely on the natural helpfulness of people or the personality traits of an individual. They convince people to open email attachments infected with malware, persuade unsuspecting individuals to divulge sensitive information or even scare people into installing malwares. There are multiple types of social engineering attacks. The most common ones are being illustrated in the following sections.

2.3.1.1 Baiting

Baiting relies on a careful analysis and background study of a domain. It involves an attacker leaving a malware infected device, such as a USB flash drive or CD in a place where someone is likely to find it. The success of baiting hinges on the notion that the person who finds it will load it into their computer. This will automatically install a malware into the victim's system thus allowing the attacker to get control.

2.3.1.2 Phishing

Phishing occurs when an adversary communicates with the victim portraying as a legitimate entity. The victim is convinced through the fraudulent communication to install malware on their device or share personal, business and/or financial information. Among various communication mediums, email is the most popular mode for initiating a phishing attack. Other mediums include, chat applications, social media, phone calls or spoofed websites. Some attacks also initiate charity pleas after natural disasters or tragedies.

2.3.1.3 Spear phishing

Spear phishing is a targeted type of phishing attack that focuses on specific individuals. An attacker gets hold of personal information that is specific to an individual and then uses it to gain trust and appear authentic. Most of the times, this information is taken from social network platforms by inspecting the victim's accounts. This personalized attack tactic leads to a higher success rate for tricking victims into granting personal information.

2.3.1.4 Pretexting

Pretexting involves fabrication of false circumstances. These force the victim into providing vital sensitive information to the attacker. For instance, an attacker pretends to be someone from the IT department and due to potential issue with a faulty system software, requires remote access of the system.

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.

2.3.1.5 Quid pro quo

In such an attack, the attacker requests private information from the victim in exchange for something desirable. This could include some monitory compensation in return. For instance, an attacker offers a free gift in exchange of some login credentials.

2.3.1.6 Tailgating

Tailgating is a form of physical social engineering. This entails the occurrence of an event where an unauthorized individual follows an authorized individual into an otherwise secure location. The goal of tailgating is to obtain confidential information or valuable property. This can take place in a multitude of ways, by either asking someone to hold the door open because they forgot their access cards or asking to borrow someone's phone or laptop to complete an urgent task and instead install a malware to steal data.

2.3.2 Credentials Stuffing

Credential stuffing refers to an attack methodology where attackers use lists of compromised user credentials. These attacks are based on the fact that users tend to reuse usernames and passwords across multiple platforms thus a compromised set of credentials are reused using bots into multiple platforms. It has been observed that around 0.1%[12] of breached credentials would work successfully in other services. Credential stuffing is known to succeed due to two main reasons. The first is the technological advancement which enables attackers to use bots for the purpose of attempting simultaneous access to different platforms, using differently masked IP addresses. The second is the broad availability of massive databases of breached credentials.

Credential stuffing can be avoided with the help of careful monitoring and analysis of the working environment. A few well-known mitigation approaches include the ones presented in the following sections.

2.3.2.1 Multi-Factor Authentication (MFA)

This approach ensures the identity of an individual with the use of some pre-known contact credentials. MFA mechanism sends an access token to a registered email address or a mobile phone to authenticate the legitimacy of the user. Attackers or its bots are unable to provide physical authentication thus rendering the MFA method a successful way of preventing credential stuffing.

2.3.2.2 CAPTCHA

As mentioned earlier, most credential stuffing approaches rely on the use of bots to ensure the harvested credentials can be tested onto multiple platforms. The introduction of CAPTCHA forces entities to perform certain actions that help prove them being human. Unlike MPA, CAPTCHA can be bypassed with the help of headless browsers.

2.3.2.3 Device Fingerprinting

A device fingerprint involves keeping track of parameters associated with a communication. These involve keeping logs of the operating system, language, browser, time zone and user agents that are being used to establish communication. If the same fingerprint is repeatedly being used, then it is likely that a credential stuffing attack is being exercised.

[12] https://owasp.org/www-community/attacks/Credential_stuffing

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.

2.3.2.4 IP blacklisting

Attackers tend to have a limited set of IP addresses that are being used to log into multiple systems. If the same set of IP addresses are being used to login to multiple accounts, then there is a possibility of a credential stuffing attack thus initiating the need for IP blacklisting.

2.3.2.5 Email addresses and user IDs

Credential stuffing heavily relies on the victim using the same credentials in multiple platforms. This is easily exploitable if the platforms accept email addresses as account IDs. In order to reduce credential stuffing, a simple yet affective approach is the use of an account ID different from an email address. This reduces the possibility of credential stuffing.

2.3.3 Network Attacks

A network attack is an attempt to get unauthorized access to a network. This can be done with either a passive or an active attack with the intention of stealing data or performing other malicious activities. In case of a passive attack, the intruder intends to monitor or steal sensitive information without making any changes to the data that is being relayed through the network. In case of an active attack, the intruder affects the system by altering the ongoing communication. This is done either by deleting content, encrypting information or simply by adding noise to the relayed message. In both these cases, attackers force their way into a corporate network and gain access to internal systems. Once inside, they complement their attack with other attacks for instance, malware dissemination or vulnerability exploitation. There are multiple types of network attacks including the ones that are described in the following subsections.

2.3.3.1 Unauthorized access

Unauthorized access refers to attacks which harvest over weak credentials or compromised accounts. An attacker accesses an account without proper permissions when the victim has used a weak password, the account lacks protection against social engineering attacks or was compromised at an earlier instance and the victim is still using the same credentials.

2.3.3.2 Denial-Of-Service Attack (DoS)

A Denial-Of-Service attack is one of the most powerful weapons on the internet. It targets online services and websites. The aim behind a DDoS attack is to overwhelm the network resources. In a DoS attack, adversaries exploit the limitation in capability of an online platform for handling user requests. When the service is overwhelmed with more requests then it can handle, it is consequently brought down. The incoming traffic can consist of connection requests or fake messages. A DDoS attack is usually combined with an extortion threat of a more devastating attack unless the company pays a requested ransom.

As already mentioned, a DoS attack requires overwhelming the resources of a system. This is usually achieved with the help of Botnets. This variation of DoS is also known as Distributed Denial of Service (DDoS). Botnets comprise of a network of remotely controlled hacked computer bots. Bots are configured to form a network and they are often referred to as zombie computers. These are used to flood the targeted website, server or network with more requests/messages than what it can accommodate. Botnets also make use of IP spoofing [9] in order to mask/ hide the origin of the flood of requests. With the emergence of the Internet of Things, DDoS attacks have become an even a bigger problem. The possibility of having low end IoT devices that can communicate over the internet with a minimalistic architecture allows adversaries to exploit these devices and their security features. These devices allow adversaries to have a set of low-end devices which can all be tuned to send messages to the target resource. Some of the most common DDoS attack types include:

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.

- UDP Flood

A UDP flood targets the incoming buffer of a target service. An attacker floods the target with User Datagram Protocol (UDP) packets. The goal of the attack is to flood the random ports of a remote host. This forces the applications running on the victim's system to repeatedly check the attacked port. As there is no legitimate message being sent to the target port, the attacker receives a ICMP 'Destination Unreachable' packet. As a result, this overwhelms the system resources and thus renders it unavailable to legitimate users.

- ICMP (Ping) Flood

ICMP flood is similar to a UDP flood. The main difference here is the type of message request. In an ICMP flood, the attacker overwhelms the target resource with an ICMP Echo request otherwise known as a ping request. The attacker repeatedly sends ping requests to the victim's system without waiting for a reply. This is also possible with the help of Botnets where multiple bots are directed toward the victim system to send messages. This attack consumes both incoming and outgoing bandwidth since the victim server will often attempt to respond with an Echo reply packet thus resulting in system slowdown.

- SYN Flood

A SYN Flood [10] exploits the underlying procedures of the commonly utilised TCP protocol. A TCP connection works with a three way handshake [11] where a SYN request is initiated to show the intent of establishing a connection. This SYN request needs to be answered with a SYN-ACK response from the host and then confirmed by an ACK response from the requester. In a SYN Flood, the attacker interrupts this handshake to pretend that the connection is not being established. In order to achieve this confusion, the attacker sends multiple SYN requests and then does not respond with an ACK message after receiving the SYN-ACK. To make it more problematic, the adversary sends multiple SYN messages from a spoofed pool of IP addresses. The victim system on the other hand, keeps on waiting for an acknowledgement for each of the request until the connection can time out. During this time, the victim cannot close down the connection by sending an RST packet and therefore the connection stays open. Before the connection can time out, another SYN packet will arrive. This leaves an increasingly large number of connections half-open. Eventually, as the victim server's connection overflow tables fill, service to legitimate clients will be denied, and the server may even malfunction or crash.

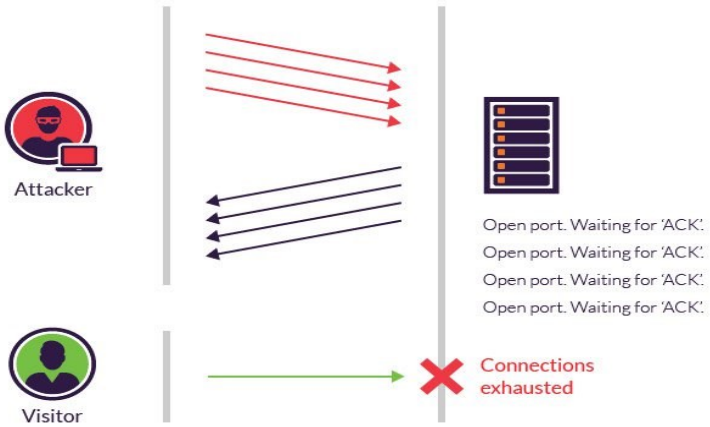

Figure 1: Progression of a SYN flood

- Ping of Death

A Ping Of Death (POD) attack [12] involves an adversary sending multiple ping requests to a victim system. An IP packet length is limited to 65,535 bytes which includes the header. However, the Data Link Layer is bounded by the maximum frame size which is far less than the packet length. For instance, an Ethernet network is bounded by 1500 bytes. In this case, a large IP packet is broken down into smaller data frames which are then

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.

combined together at the receiving end. The receiver reassembles all IP fragments to complete the IP packet. In case of the POD attack the received IP packets are larger than 65,535 bytes and this results in an overflow of the memory buffer at the receiving end thus causing a denial of service for legitimate packets.

- Slowloris attack

Slowloris[13] is a targeted attack. It involves an attacker taking down a particular web service by targeting messages only to a particular port. The attacker achieves that by holding as many open connections to this specific port of the targeted web server as possible. The attacker then occupies these connections by sending only partial requests. Slowloris attacks constantly send more HTTP headers and avoid completing the requests. The victim keeps these false connections open in the hope of getting messages. This as a result overflows the connection pool and leads to denial of service for additional requests from legitimate users.

- HTTP Flood

In an HTTP flood, the attacker exploits the HTTP GET and POST mechanism to occupy unnecessary resources from the victim's end point. An HTTP flood does not use malformed packets. Instead, the attacker uses less bandwidth to bring down the system. The attacker forces the victim to allocate maximum resources possible for the response of a single request. This can be achieved by initiating a small but tedious request.

2.3.4 Brute Force Attacks

A brute force attack refers to the use of a trial and error approach to get the correct login information. Adversaries tend to use all possible combinations to guess the correct credentials. A brute force attack is complementary to social engineering which allows adversaries to have an idea of potential credentials that would work on the targeted platforms. Attackers use brute force attacks for a multitude of reasons including but not limited to collecting activity data, stealing personal data and valuables, spreading malware, hijacking a system or ruining a website's reputation. There are multiple types of brute force attacks including:

- Simple brute force attack

Adversaries try to logically guess user credentials without the aid of any external source. This approach works on extremely simple scenarios where passwords and pins are as simple as *guest12345*. Many IoT devices that are out there use simple login credentials. These are usually id:*admin* and password:*admin*. The intention of manufacturers is that users alter these credentials as soon as they first get the device, however due to lack of knowledge, the second tend to leave them unchanged. As a result, adversaries use these credentials to launch a simple brute force attack.

- Dictionary attack

A dictionary attack is the most basic tool for a brute force attack. In such an attack, the adversary uses a list of possible passwords clearly marked with ranks giving them an idea of what credentials could work. This list includes the most common passwords along with some information about the environment. Thus, it gives the attacker a better chance of penetrating the system.

2.3.5 Brute Force Attack versus Credential Stuffing

Brute force attacks are very similar to credential stuffing. However, there are several important differences:

[13] https://github.com/gkbrk/slowloris

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.

- A brute force attack tries to guess the credentials of an individual without any associated context. The attacker uses random strings, common passwords or dictionaries of common phrases.
- Brute force attacks heavily depend on the use of simple, guessable passwords. If the user uses long, complex passwords, then the possibility of a successful attack is lower.
- Credential stuffing relies on having information from previous breaches. This increases the success rate of credential stuffing.

Well known password dictionaries, that can be used for brute-force attacks and credential stuffing, can be found as open source software[14] [15].

[14] https://github.com/philipperemy/tensorflow-1.4-billion-password-analysis
[15] https://github.com/topics/password-dictionaries

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 Capability Assessment of SPHINX Modules

This section provides a brief presentation of the main SPHINX components that are closely linked with the ABS component and specifically the Incident/Attack Simulations. Within this task, we will create network traffic abnormalities that will need to be detected by the following components.

3.1.1 Data Traffic Monitoring

DTM is a rule-based intrusion detection system. Following, some representative examples of threats that the DTM component can detect are listed:

- Connections to and from IPs with poor reputation or that are known to be hostile or compromised (bot command & control server, spam network).
- DoS & DDoS attacks targeted to hospital network
- DDoS using computers in the hospital network to attack remote targets
- DNS protocol threats (ie DNS Reply Sinkhole, Cache Poisoning Attempts, Non-Compliant DNS traffic on DNS port, DNS queries for suspect domains)
- Exploits based on known network communication (ie Possible Internet Explorer VBscript failure to handle error case information disclosure CVE-2014-6332 Common Construct, CVE-2015-0235 Exim Buffer Overflow Attempt (HELO), D-link DI604 Known Malicious Router DNS Change GET Request, ossible Firefox PDF.js Same-Origin-Bypass CVE-2015-4495 M2)
- FTP threats (ie PASS overflow attempt, RMDIR overflow attempt, FTP large SYST command, FTP CWD ~root attempt, FTP Outbound Java Downloading jar over FTP)
- IMAP threats (ie IMAP rename overflow attempt, IMAP authenticate overflow attempt, IMAP unsubscribe overflow attempt)
- Malware based on known network communication (ie Android Trojan MSO.PJApps, Android Trojan Fake10086, Android/Smspacem CnC, Android.Walkinwat)
- Netbios threats (ie Microsoft Windows NETAPI Stack Overflow Inbound - MS08-067, Microsoft Windows SMB Client Race Condition Remote Code Execution, Microsoft Windows Server 2003 Active Directory Pre-Auth BROWSER ELECTION Heap Overflow Attempt, SMB CoGetInstanceFromFile little endian attempt, DCERPC DCOM ExecuteShellCommand Call - Likely Lateral Movement)
- Usage of torrent download application
- POP3 threats (ie AUTH overflow attempt, LIST overflow attempt, DELE negative argument attempt)
- TCP port scans, UDP port scans
- Database server (MySql, SQL Server, Oracle etc) threats (ie root login attempt from a remote IP, brute force login attempts, calling sensitive function or procedures from a remote IP)
- Web servers & clients threats (there are rules for IIS, Apache, Tomcat servers for SQL Injection vulnerabilities and other known vulnerabilities and attacks)
- Worms based on known network communication (ie Rimecud Worm, W32/Njw0rm, Slammer Worm)

3.1.2 Anomaly Detection

The AD component uses anomaly detection algorithms in order to identify suspect variations in the traffic data. AD can detect:

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.

- compromised hosts in the hospital network that send data outside the network. This is detected based on sudden and persistent increase of the volume of traffic relative to historical values of the volume of traffic.
- vertical and horizontal port scanning
- spam sending hosts
- hosts suspect of being infected with malware based on change of typical DNS traffic.

Along with the simulations of normal traffic in the context of T5.4, the Attack Simulator is intended to produce attack traffic inside the emulated network topologies. By sniffing this traffic and using summarisation protocols such as NetFlow, sflow, the ABS component is planned to be used for producing appropriate datasets for training the Anomaly Detection Component.

3.1.3 SIEM

The main objective behind the SPHINX Security Information and Event Management component is to provide a solid log management tool, for security-related events that the security administrators can rely on for responding to security incidents as early as possible. Thus, the SIEM component will encompass log collection and normalization, data correlation, alert management and reporting, to guarantee near real-time analysis of security alerts, which have been generated by network hardware and applications.

The SIEM can detect quite a few different threats, some of the most prominent ones are the following

- Brute force attacks

By monitoring events where there are a certain number of failed login attempts. When this specific event occurs, it should trigger an alert and send an email to the security team. In the case of brute force attacks the SIEM can detect with the help logs gathered from the user authentication system.

- Portscan detection

In the case of portscans, the SIEM is going to be watching for access attempts on different ports from the same source IP in a very short timeframe.

3.1.4 Homomorphic Encryption

The Homomorphic Encryption (HE) tool developed for the SPHINX project comprises of multiple modules. These include, search within the encrypted domain, double-sided blinded search, and data anonymization. The tool makes use of the state-of-art homomorphic encryption to ensure that user data is safe from adversarial attacks.

The search in the encrypted domain module, is built around the concept that healthcare data is of high interest to adversaries, due to the nature of inferences that can be made from such data. To get access to such information, intruders tend to attack various interfaces including the platform that hosts all the information. Healthcare databases which host such information store it in plain text, thus a database penetration attack will give them access to all stored information. The search in the encrypted domain module overcomes this limitation by encrypting all information stored in the healthcare database. Intruders will only find encrypted data when they try to access information being relayed through the network or when they get illegitimate access to the healthcare database. In case legitimate users want to access any information stored on the database, they can use the HE tool to send an encrypted query, to receive a filename that contains the desired content.

The double-sided blinded search tool makes use of the capabilities of the “search in the encrypted domain” tool to geographically distant healthcare institutes, to share information. Sharing healthcare information is crucial and it is necessary to have this as secure as possible. The double-sided blinded search tool allows

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.

healthcare professionals to search in each other's databases. The searching entity sends an encrypted request to the database where they want to execute the search. This is then executed and a response is generated that contains the name of the file that contains the search query. Details of the encryption and search tool can be found in D4.6 Homomorphic encryption embedded engine v1.

The data anonymization tool is another important component that ensures the privacy of user data. It takes network traffic meta-data as input and anonymizes all user-related information. This involves anonymization of source and destination IP addresses. The tool replaces legitimate IP addresses with surrogate IP addresses and maintains a homomorphically encrypted map between the actual IP and the surrogate addresses. This map comprises of a one-way cipher, thus allowing the tool to maintain homogeneous mapping. The tool maps the same original IP to the same surrogate IP with the help of this one-way cipher. As a result, the SPHINX toolkit can carefully monitor the necessary traffic patterns that are being generated by the same IP address.

In this simulation, the homomorphic encryption will be tested indirectly. In many use cases the encrypted and anonymised data are stolen by the attackers. To test the effectiveness of the homomorphic module, the attacker will try to decrypt the stolen data.

3.1.5 Forensic Data Collection Engine (FDCE)

These Forensic Data Collection Engine provides the basis required for supporting the processing and storage of data gathered from the various SPHINX modules into a unified structure, in order to discover the relationships between devices and the related evidence, and produce a timeline of cyber-security incidents, including a map of affected devices and a set of meaningful chain of evidence (linked evidence).

The FDCE is a data aggregator and will therefore not be directly observing the simulated cyber-attacks. Instead, it will be relying on the observation of the previously presented tools, and through them produce the observed chain of evidence. Ideally, the FDCE, if provided with the necessary data, shall be able to capture the nature of the simulated attack and its impact on the system, for each different simulated scenario .

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 Overview of Security Incident/Attack Simulator

4.1 Design Principles

The security incident/attack simulator is an essential part of the ABS component that cannot operate as a standalone application. The attack simulator itself comprises three different sub-components, each responsible for simulating a different attack. The component is designed to fulfil the stakeholders requirements, as laid out in deliverable D2.6, the use cases, as laid out in deliverable D2.4 and will be updated according to their next iterations, to ensure smooth interoperability with the other parts of the ABS module and the experimentation environment.

4.1.1 Stakeholders Requirements

According to deliverable D2.6, the ABS component has four (4) basic functional requirements ABS-F-10, ABS -F-20, ABS -F-30, and ABS -F-40. These requirements fulfil some of the functional requirements provided by the stakeholders. The table below illustrates the functional requirement fulfilment between the ABS and the stakeholders. (Table 1: Functional Requirements of ABS module).

Requirement	Dependency	Description	Fulfilled by task
ABS-F-010	STA-F-010	Produce normal behaviour datasets	5.4
ABS-F-020	STA-F-010	Evaluate the produced datasets compared to original datasets	5.3, 5.4
ABS-F-030	STA-F-010	Produce datasets incorporating cyber-attacks	5.3
ABS-F-040	STA-F-010	Use the output in the experimentation environment	5.3, 5.4

Table 1 Functional Requirements of ABS module

The architecture requirements consider the ABS component as the common outcome of T5.3 and T5.4. In the last column of table 1 this co-operation is provided. Specifically, task T5.3 is not expected to produce datasets representing normal behaviour from the users but is involved in all the other requirements of the ABS component.

4.1.2 Conformance with the SPHINX Use Cases

As already mentioned, the main purpose of the ABS component, is to validate the capabilities of the other SPHINX components. To achieve this, it would not be efficient to solely develop and test cyber-attacks at random. The attacks must be similar to the ones that the real systems of the pilot hospitals are expected to face. The best point of reference that guides the nature of these attacks is found in deliverable “D2.4 Use Cases definition and requirements document v1”, which describes the applicable use cases. While these use cases provide a very good initial view of the work that needs to be done, since D2.4 was developed early during the first phase of the SPHINX project. This has led to changes that will be addressed in deliverable D2.7 “Use Cases definition and requirements document v2” due in M32. This evolving process heavily affects the design of

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.

simulated attacks, as in most of the cases adjustment is needed to better address the new parameters. As such, this deliverable will present a more general overview of the possible attacks that are related to the SPHINX use cases along with some relevant tools in order to achieve them in a safe environment.

4.1.2.1 UC1 / UC3 / UC4 / UC6 / UC9

All these use cases are grouped together since they all have in common one of the most popular and effective attack vectors, which is users downloading and running a malicious executable themselves. One of the biggest problems to prevent this attack, is the fact that the initiators of the attack are the legitimate users themselves, taking actions they are allowed to take in the confines of the system. Even if it is ensured that only the minimum necessary permissions are given to each user, often those trusted users can fall victims to such attacks.

4.1.2.1.1 UC1: Attacking Obsolete Operating Systems in Hospital

The first use case describes a hospital using old and vulnerable software, which have well-known vulnerabilities that can be exploited by an attacker. This use case describes legacy equipment being used that should not be in use anymore, and due to lack of critical updates are now vulnerable to known attacks. The user downloads some software from a web site that leverages social engineering techniques to target users in the hospital.

The ABS implementation

To implement this attack, more than simple attack scripts are required. In the simulated environment of ABS component, a node with outdated software shall be deployed. The node shall have known vulnerabilities, that can be exploited by an external attacker, something that the attack will be able to detect using port scanning before realising the attack itself. Ideally, this software will be either one currently used in the pilot environment, or if such outdated software does not exist, a relevant software program with known vulnerabilities will be installed on a virtual node. The attack will attempt to establish a connection with an outside server, from where the attacker shall download malicious software and run it on the system. In this particular use case, the attack will turn the hospitals devices into botnets, that can be used to conduct DDoS attacks, to the hospital itself and to any target on the internet. SPHINX should be able to detect both accessing known flagged websites, connection attempts to C&C server, the botnet behaviour and generally unusual traffic. It should also be able to detect any known vulnerabilities.

Some common bot programs that were easily found online and have been used for the creation of well-known malware datasets, such as the CTU-13 dataset [13] are *Rbot, Neris, Virut, DonBot, Murlo, Nsis.ay, Ares*[16]and *Conficker,* which is described in D2.4. After such malwares are installed, communication with the attacker is essential by means of a remote shell or automated Client to Client(C2C). The malware is usually going to send out a keep alive signal periodically to let the server know it is still active and to see if there is anything it should do. When the server is ready, it will execute the next step of the attack on the infected host machine. Some common C2C communication mechanisms are the following:

- Common Types — Bind Shell/Reverse Shell/Web Shell
- Standard Protocols — HTTP/HTTPS, DNS, SMTP, SSH, ICMP, Telnet, SMB, FTP, IRC, Torrent
- Various Techniques — Encryption, Circumvention [TOR (The Onion Router) & I2P (Invisible Internet Project)], Port Evasion, Fast Flux (Dynamic DNS)]

4.1.2.1.2 UC3: Rootkit Malware Attack in a Cancer Treatment Institute

[16] https://github.com/sweetsoftware/Ares

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.

In UC3, the user downloads the malicious malware from an email attachment which uses insecure protocols, indicating its fraudulent origin. These emails are not sent to a specific user but to several users in the network. The particular malware attached shall be a rootkit, which will communicate information back to the attacker.

The ABS implementation

Firstly, the replication of the email traffic directed to the users of the network shall be performed, with different intervals and volumes in different tests, to cover different techniques utilized by the attackers. Then the attached rootkits will be installed. For the testing, multiple different types of rootkits will be considered that behave differently and use different protocols to communicate back to the attacker. The SPHINX components are expected to detect the usage of email on an unencrypted channel (without TLS), detect the high number of emails in a short interval from the same sender, recognise a suspicious external remote connection.

In the case of infected email being sent to the user, an effective tool that can be used is the social engineering toolkit[17], which is included in Kali Linux and can fulfill all the different requirements as described in the use cases. Additionally, there are common rootkit software, available as open source software[18].

4.1.2.1.3 UC4: Theft of Health Data by Exploiting Vulnerable Software

Similar to the previous use case, in UC4 the method of delivering the malware to the system is the same, through a file sent by email. The major difference is that the malware delivered, contains a remote access trojan, which behaves as a key logger and steals information such as user passwords.

The ABS implementation

Firstly, the malware will be propagated as email attachments. The trojan will be installed and send data to external IPs. SPHINX should be able to detect the connection if they are aimed to IPs with low reputation, by recognising a suspicious external remote connection and the raise in traffic volume when the stolen data is sent away and finally by ensuring that the file are encrypted using homomorphic encryption making them useless to an attacker.

As already mentioned, a tool to send phishing emails is the social engineering toolkit, which is described previously. Common Trojan software are also found as open source tools[19]. Zeus[20] is one of the most popular and has been used in the creation of the famous IDS-2018 dataset [14].

4.1.2.1.4 UC6: Ransomware Attack to Healthcare Data

In UC6 the systems are infected through an email attachment, downloaded by the users. In this attack the infected files contain a ransomware, which will encrypt all the files it has access to.

The ABS implementation

The initial stages of the attack, with emails containing malware will also be replicated in this use case. In addition, we will attempt to replicate the behaviour of a ransomware attack, which spreads to other devices and encrypts all the files. SPHINX should detect suspicious network activity caused by the cryptoworm propagation, identifying the source devices, in addition to email relevant activity that was previously mentioned. SPHINX may also utilise system logs to detect the mass encryption of files before the attack is fully completed.

[17] https://github.com/trustedsec/social-engineer-toolkit
[18] https://github.com/milabs/awesome-linux-rootkits, https://github.com/d30sa1/RootKits-List-Download
[19] https://github.com/threatland/TL-TROJAN
[20] https://github.com/Visgean/Zeus

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 already mentioned, a tool to send phishing emails is the social engineering toolkit, which is described previously. Ransomware payloads are available as open source software meant for testing purposes[21].

4.1.2.1.5 UC9: Compromised BYOD Enables Stealing of Patient Data

In this use case, a legitimate user has his/her device infected through the use of email. The difference in this case, is that the user gets his/her device infected by using a different network, which is unmonitored from the SPHINX modules. The malware in this case is a keylogger that transmits data when the device connects to the internet.

The ABS implementation

Similar to the other use cases, there is no initial precursor to the attack, since it takes place in a unmonitored network, therefore SPHINX cannot detect it. But if the device connects to the internet at a later date through the monitored network, SPHINX will be able to detect the anomalous traffic and the connection to unreliable low reputation IPs.

As already mentioned, a tool to send phishing emails is the social engineering toolkit, which is described previously.

4.1.2.2 UC2: Hijacking Access to National Healthcare Databases

Use Case 2, essentially describes an attacker scanning the networking and enumerating the system and using brute force techniques using common user passwords combinations or stolen credentials from well-known datasets. After acquiring access with a stolen admin account, the attacker then changes the settings of a device in order to take down a service.

The ABS implementation

Like in UC1, scanning tools shall be used to perform external scans to the network. The scanning will be initiated by an address which is commonly used in the network, instead of a different one. In addition, brute force techniques using dictionaries of common user – password combinations and datasets of known stolen combinations shall be employed. SPHINX components should be able to detect the scanning activities and the brute force attack to our system.

As already mentioned in a previous section, some scanning tools that are going to be used are NMAP, Nessus and OpenVAS vulnerability scanners, Nikto and DirBuster.

4.1.2.3 UC5: Tampering with Medical Devices

In UC5 the vector of attack used is a physical medium, specifically a usb stick that contains the malware, which infects a device immediately. While the use case does not specify the exact nature of the malware, its behaviour resembles of that a worm attack.

The ABS implementation

The malware will be installed directly to a specific VM in the network, without any precursor since there is none, that SPHINX can detect, since SPHINX does not have immediate feedback about the physical security of the organization. The malware installed will commence a worm style attack that will attempt to spread to connected device. This traffic, contaminating other devices should be identified by the SPHINX components.

[21] https://github.com/mauri870/ransomware, https://github.com/leonv024/RAASNet

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.

There are plenty examples of worm malware on the internet, but at this point NetWorm is the selected candidate (GitHub.[22]).

4.1.2.4 UC7: Distributed Denial-of-Service Attack in Regional Hospital

UC7 describes a botnet DDoS attack that targets both the main system of the organisation and the remote services servers in particular. In the use case, that hospital is not aware of the attack for a long period of time since they lack the means to easily detect the system availability.

The ABS implementation

We will realise several types of DDoS attacks towards the services of the network. If possible, the attack will originate from many different hosts simulating a botnet behaviour as best as possible and not from a single host. The attack will also target the remote health service server in particular. SPHINX should be able to detect the DDoS based on high number of requests / second and detect the DDoS based on sudden change of volume usage relative to previous volume usage.

There exist a large number of DDoS tools that target different parts of a system. At this point slowloris[23], golden eye[24] and hulk[25] are considered the most appropriate candidates.

4.1.2.5 UC8: Compromising Health Services through Cryptocurrency Mining

UC8 describes a user with administrator privileges, using them to install a crypto-currency mining software that uses the resources of the existing computer systems and causes issues for all the users of the systems.

The ABS implementation

Mining software shall be installed directly to a node, since there are no precursors in this use case. SPHINX components should be able to detect the typical connections used by the mining software.

Many different open source implementations of crypto currency miners exist, especially for popular crypto currencies such as Bitcoin[26].

4.1.2.6 UC10: Taking Control of Connected Medical Devices

In UC10, the attacker uses network scans to detect existing vulnerabilities in various devices connected to the network. The attacker proceeds to steal the patients' data and alter the ones in the database. The changed data should go unnoticed since only one patient's data were altered.

The ABS implementation

Standard network scanning shall be performed that the SPHINX platform should be able to detect. Afterwards it shall be attempted to alter some database data. These changes are expected to be picked up in the logs, which SPHINX should be able to detect. In addition, the data generated by the external connection are also expected to be picked up.

The selected scanning tools that are going to be used are NMAP, Nessus and OpenVAS vulnerability scanners, Nikto and DirBuster.

[22] https://github.com/pylyf/NetWorm
[23] https://github.com/gkbrk/slowloris
[24] https://github.com/jseidl/GoldenEye
[25] https://github.com/Mr4FX/Hulk-ddos-attack
[26] https://github.com/topics/bitcoin-mining

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.

4.1.2.7 UC11: Intrusion in the Clinical Centre's Wireless Network

The attacker in UC11, uses a brute force attack on vulnerable Wi-Fi routers, by taking advantage the weak encryption of outdated protocols which severely limits the possible combination passwords making a brute force attack possible. In addition, he concludes a network scan from the internal network.

The ABS implementation

A brute force attack shall be implemented against the system like in the previous use cases along with a network scan. The major difference in this use case is that the scan and the attack are done internally. The SPHINX platform will be able to identify the brute force attack used to access the WIFI router and detect the network scan.

For the brute force attack Patator[27] tool is selected, which utilises a variety of protocols, including smtp, ftp database end points, etc. Hydra [28], Medusa [29] are also under consideration.

4.1.2.8 UC12: Hacking Health IT Systems

UC12 essentially describes a patient, taking on the role of a traditional pen tester, with internal access to the network, with the major difference that she/ he is never asked for permission, like a proper penetration tester would do.

The ABS implementation

Traditional penetration test shall be implemented first using scanning tools and as a second step it will try to use any detected vulnerabilities and weaknesses of the system. In this case, SPHINX will detect a new unauthorised device trying to enter the care unit's network. It will also detect suspicious network activity caused by the network scanning.

As already mentioned in a previous section, some scanning tools that can be used, and are traditionally used in the process of penetration testing are NMAP, Nessus and OpenVAS vulnerability scanners, Nikto and DirBuster.

4.1.2.9 UC13: Exploiting Remote Patient Monitoring Services

In this use case, the attack is done through a malware called VPNFilter. The issue is that this malware in the use case does not infect the hospital's infrastructure but the equipment in the patient's home where the remote equipment is located. This means that the neither the hospital staff nor the SPHINX platform will have access to the data and traffic that traverse the network of the remote patient. The only data SPHINX will be able to digest is those sent by the remote app eCare of the edge platform.

The ABS implementation

In this use case the implementation is constrained to the data sent by the edge, eCare testbed client. In that case in order to simulate this attack, the contents of the client application shall be altered before these are sent back to the server. Data shall contain structurally correct data, e.g. data that will be accepted normally by the server but they will be different from the data the client was supposed to send. In another scenario, the data shall contain malicious code aiming to breach the defences of the server. An example of that would be code for an SQL injection. SPHINX should be able to continuously monitor the end-to-end communication in the network and identifies the absence of encrypted end-to-end communication.

For this use case custom scripts embedded in the communication of eCare clients shall be executed.

[27] https://github.com/lanjelot/patator
[28] https://redteamtutorials.com/2018/10/25/hydra-brute-force-https/
[29] https://github.com/jmk-foofus/medusa

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.

4.1.2.10 UC14: Zero Day Attack to eHealth Services

UC14 describes attackers finding a zero day vulnerability. Since it is a zero day attack, there would be no information that would enable the SPHINX platform to notice the vulnerability beforehand. The attacker proceeds to install a malware through the vulnerability and to steal data and sell them back to the hospital along with information about the vulnerability itself.

The ABS implementation

Zero-day attack shall be simulated as a simple attack against a known vulnerability giving access to a terminal from where the attacker gains access to the database. SPHINX should protect the stolen data with homomorphic encryption not allowing access to them even if stolen and should be able to detect the anomalous behaviour in the network, like the communication with the malware command and control centre.

Since no specific attack is specified by the use case one of the aforementioned attacks shall be reused.

4.1.2.11 UC15: Theft of Hospital Equipment

UC15 presents a different approach than all the rest. Here the real vulnerability exploited, is that of the physical security of the hospital and the management of its electronic devices. The attack uses the “normal” actions allowed to the user of the stolen account. In this case the detection of this attack is extremely difficult because it can start instantaneously without any precursor and it uses the legit capabilities already assigned to users.

The ABS implementation

In this case the attack will not actually use, any hacking scripts or scanning software. The attack will simply use existing legitimate capabilities. The difference is that these functions may not correspond exactly to the usual traffic, by being used in different times and different frequencies. SPHINX will be able to understand these differences in the use of an equipment.

We will not use any dedicated tools and rely on the improper use of existing functions, that legitimate users can already do in our environment.

4.1.2.12 UC16: Intercepting Cross-border Healthcare Data Exchange

UC16 describes the cross-border communication protocol between two different hospitals and how they exchange data on patients. Users fail to confirm the digital signature of the attackers’ email, and sends a link to them that gives them access. In addition, this protocol is using untrusted technologies such as unencrypted email without TLS, that allows the attacker to gain access to their data.

The ABS implementation

In this use case again, there is no actual malware injected to the system. Instead the attack will involve sending unsecured smtp traffic. The other part of the attack simulation will involve accessing a service on the system with proper credentials, something that should be detected by the SPHINX system, since the connection is done by an untrusted and unusual address.

As already mentioned, a tool to send phishing emails is the social engineering toolkit, which is described previously, besides that we will not use any dedicated tools and rely on the improper use of existing functions, that legitimate users can already do inside the emulated environment.

4.2 Experimentation Environment

In general, security testing, while effective, has a few inherent issues closely related to the environment where such testing takes place. The first approach mandates that the target system is usually live in a production environment. The testing needs to be conducted while the system operates normally. While this is a standard

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.

practise in certain areas of the industry, like bug bounty programs, this usually requires organisations with enough resources to fix issues quickly. In addition, this poses an issue with testing on critical systems. Critical systems, especially those found in hospitals, cannot afford any downtime.

The other major method of conducting security testing is by constructing a virtual environment e.g. a sandbox, isolated from the main system, which replicates the main system. This method comes with its own problems, namely the difficulty in accurately emulating the real network, which may necessitate generalisations.. Additionally, creating datasets that represent real traffic in simulated environments causes its own issues, in terms of actual realism. Nonetheless, this approach is beneficial in the Attack Simulator case, because it prevents the experiments' interaction with real critical systems. In addition, it allows for the utilisation of mock data, adhering to privacy and data protection regulations, and this is why this approach will be mostly followed during the experimental stages of T5.3.

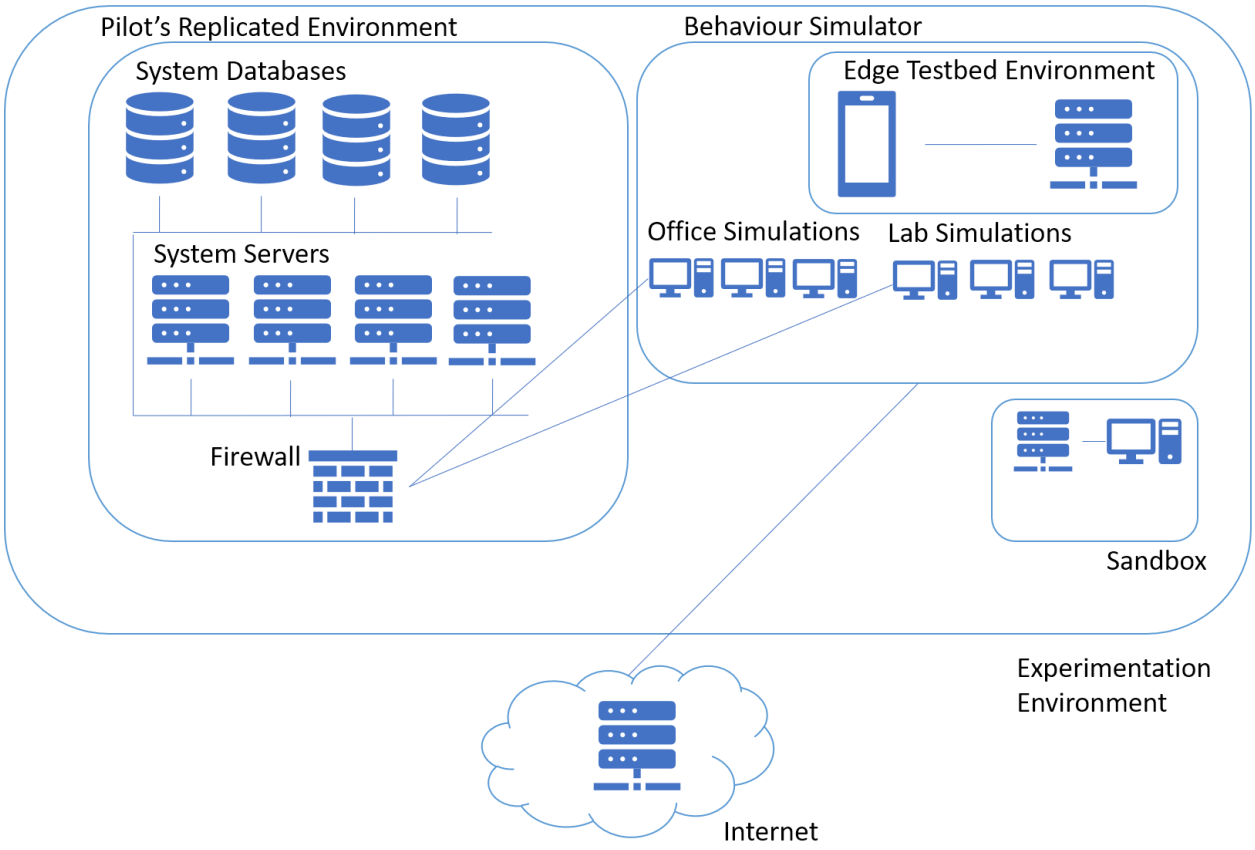

Figure 2 High level experimentation environment architecture

The experimentation environment shall be hosted in the premises of the pilots and it shall include virtual machines (VM) imitating all the major services that currently run on their real network. These services shall include services such LIS servers, databases, and firewalls. In conjunction with these VMs, several other SPHINX software components shall be used to emulate the rest of the hospitals systems and simulate their behaviour. These components are briefly described in the following sections.

4.2.1 Behaviour Simulator

The deliverable D5.4 describes the efforts of SPHINX project, to recreate some of the systems that compose the pilot's ecosystem, and most importantly the behaviour of the users in them. Those systems are mainly hospital networks that include medical devices, various staff workstations and machines, IoT networks and relevant servers, and a large number of medical lab subnets that need to be simulated. The behaviour simulator will be able to simulate arbitrary network topologies through NFV technologies and these networks are

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.

expected to be similar to the real hospital networks in terms of infrastructure and the network traffic that flows in between those networks and between themselves and the internet.

4.2.2 The SPHINX Sandbox

The SPHINX Sandbox will be a security mechanism for separating running programs, usually to run software that can harm the general network. The proposed shared environment will be often used to execute untested or untrusted programs or code, from unverified or untrusted third parties, suppliers, users or websites, without risking harm to the host machine or operating system. The proposed sandbox will provide a tightly controlled set of resources for guest programs to run in, hosted in SPHINX Cloud Backend. The attack simulation and network emulations will take place in the testbed environments created within T5.4 while the whole system is planned to be hosted inside the Sandbox for isolation and security purposes.

4.2.3 Remote Healthcare Monitoring

4.2.3.1 The eCare Platform

Remote medical and health devices allow healthcare professionals to closely follow patients outside of the office, either through telehealth (video consultation) or remote patient monitoring, the latter involving connected devices, Apps and/or web-applications. Since remote healthcare provision involves interaction with outside networks and systems (e.g., the patient's home environment), it also brings a number of challenges in what regards cyber-security since it creates new, often insecure, entry points for hackers and rising data security and liability risks. For example, *Users of mobile devices are increasingly subjected to malicious activity that is pushing malware apps to their phones, tablets and other devices running Android and iOS* [D2.1 (1.5.2)]. Since these devices are outside the healthcare organisation's control, there is also a lack of visibility and control over personal devices, as well as the absence of awareness of these devices' vulnerabilities that attackers could take advantage of.

SPHINX addresses use-cases involving remote patient monitoring, aiming to prevent and detect suspicious and illicit activity. For this purpose, SPHINX partner EDGE brings the eCare Platform Testbed (see D5.4 for a detailed description), which is a simulation environment that allows generating requests and interactions between **patients** (i.e., simulated eCare Platform users), **devices** (i.e., simulated healthcare IoT equipment) and **healthcare services**.

In order to evaluate the SPHINX effectiveness in a remote healthcare environment scenario, the eCare Platform Testbed is adapted to general abnormal and malicious user behaviour.

4.2.3.2 Possible simulation of Attack Vectors in eCare platform

The types of attacks that can be simulated and emulated in the context of the eCare Platform are numbered in **Error! Reference source not found.** and are described next.

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.

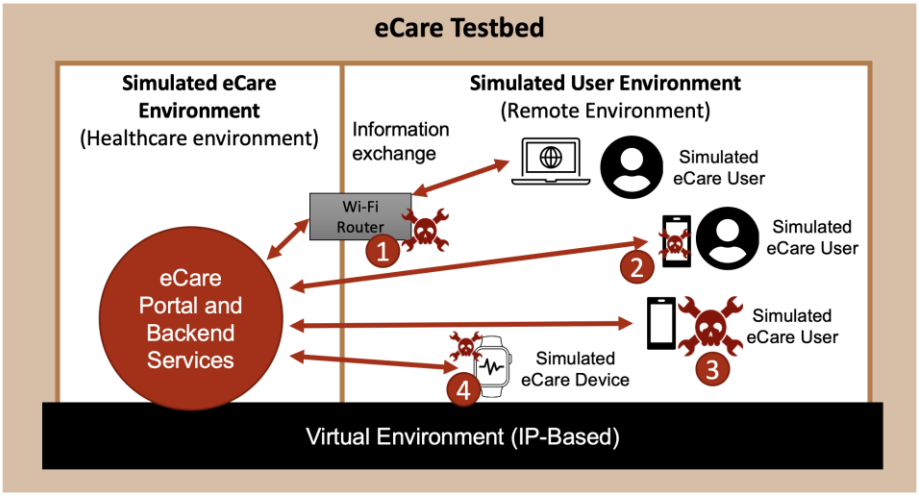

Figure 3 – Simulated and Emulated Attacks for the Remote Healthcare Environment Scenario

- **Number 1**: An opportunist hacker is able to crack the password of the patient Wi-Fi network. By monitoring the web-traffic, the attacker is able to steal the patient's access to the healthcare portal (eCare).
- **Number 2**: Unknowingly, the eCare user installs a malicious App that allows an opportunist hacker to gain access to the user smartphone, including gaining access the eCare App.
- **Number 3**: An eCare user starts to interact with the system with malicious intent.
- **Number 4**: An opportunist hacker gains access to an eCare device.

The several attack vectors may provide to the attacker remote access to the eCare server. Possible exploitations and causes include.

Exploitation	Effect
Attempt to retrieve large amounts of data related to the user.	Abnormal increase of traffic related with a **user**.
Attempt to access records that the user is not authorised to access.	Abnormal high number of requests received from a **user**. Generation of several errors and warnings on the server related with non-authorised access operations from a **user** or **device**.
Attempt to insert fake/garbage healthcare data.	Generation of several errors and warnings on the server related with abnormal data, lack of data integrity or illegal operations performed by a **user** or **device**.
Attempt to affect service availability.	A very high number of requests is made to the portal, overwhelming its network connections and impeding other users from accessing it.

Table 2 Possible exploitation of EDGE testbed

Expected response from SPHINX: SPHINX should detect the presence of an abnormal behaviour between a user (malicious hacker) and the eCare healthcare portal. Collected information should include origin of the attack (e.g., IP address, traceroute and respective geolocation if available), time of the attach (start and stop) and related events (e.g., system logs, traffic).

SPHINX should detect the following abnormal behaviour:

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.

- Abnormal high number of access requests
- Abnormal increase in the portal inbound and outbound traffic (from a single user or device)
- Abnormal data insert/modification operations from a single user or device, including failed attempts generating a high number of errors.
- Attempt to access a high number of non-authorised records.

4.3 Technical Details

Being a subcomponent of the ABS module, the security incident attack simulator uses the interfaces for its communication with the internet and the sandbox as outlined in the architecture. In addition, it provides the operational data for the CIP interface.

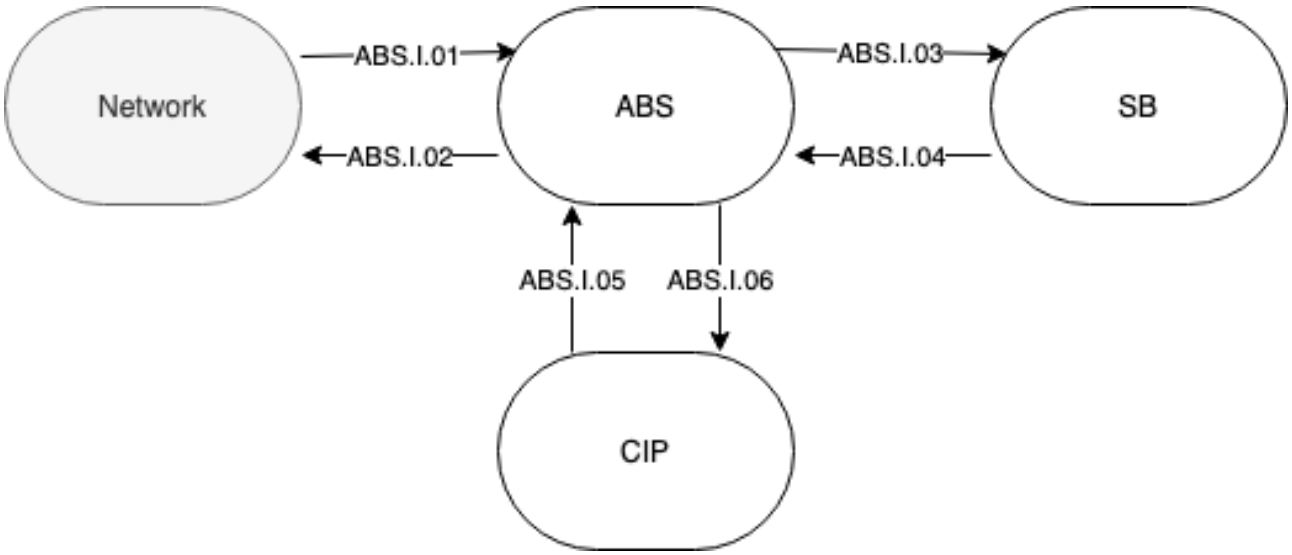

Figure 4 ABS Component Diagram

As demonstrated, the security incident attack simulator itself is a complex tool that is intrinsically linked with the behaviour simulator and the experimentation environment. It is essentially composed by three different sub-components the Flask application, the Local Malware Execution, and the Abnormal Parametrisation of the Behaviour Simulator.

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.

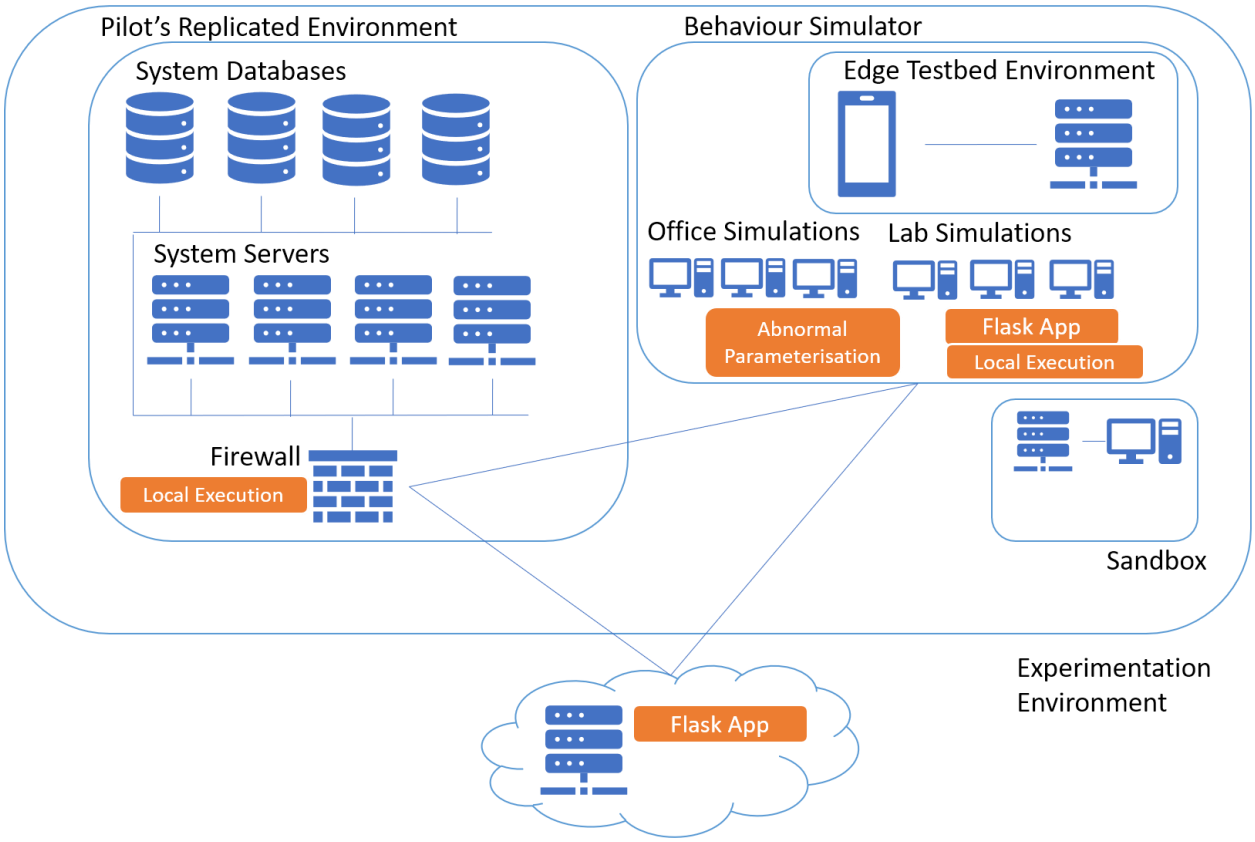

Figure 5 High level experimentation environment architecture with Attack Simulator Modules

4.3.1 Flask Application

The first subcomponent of the attack simulator aims to simulate all the traditional attacks that are made by either external or internal attackers. These attacks include all steps of a traditional cyber-attack, such as port scanning and actual exploitation attacks.

The working environment of this sub-component includes an application, developed using the Flask [30] framework. This application, which is under development, works as a front-end for the coordination of the Docker containers of the back-bone environment (either created manually or through Containernet [31] as described in D5.4) and potentially as a coordinator for some of the functions needed for the execution of the attack scenarios. This application is described in more detail in D5.4 focusing more on traffic generation issues. However, additionally to the traffic generation capabilities that are provided by this GUI, cyber-attack-oriented submodules and functionalities are also being added to this initial architecture. In this context, there has already been implemented a user menu that allows the user to create and deploy DoS attacks. This kind of attacks are discussed in section 2.3.3.2 of this document. The page, depicted in Figure 6, works as a front end, orchestrating the docker containers that are running in the background to exchange DoS flows between each other using the hping[32] linux tool.

[30] https://palletsprojects.com/p/flask/
[31] https://containernet.github.io/#overview
[32] http://www.hping.org/

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.

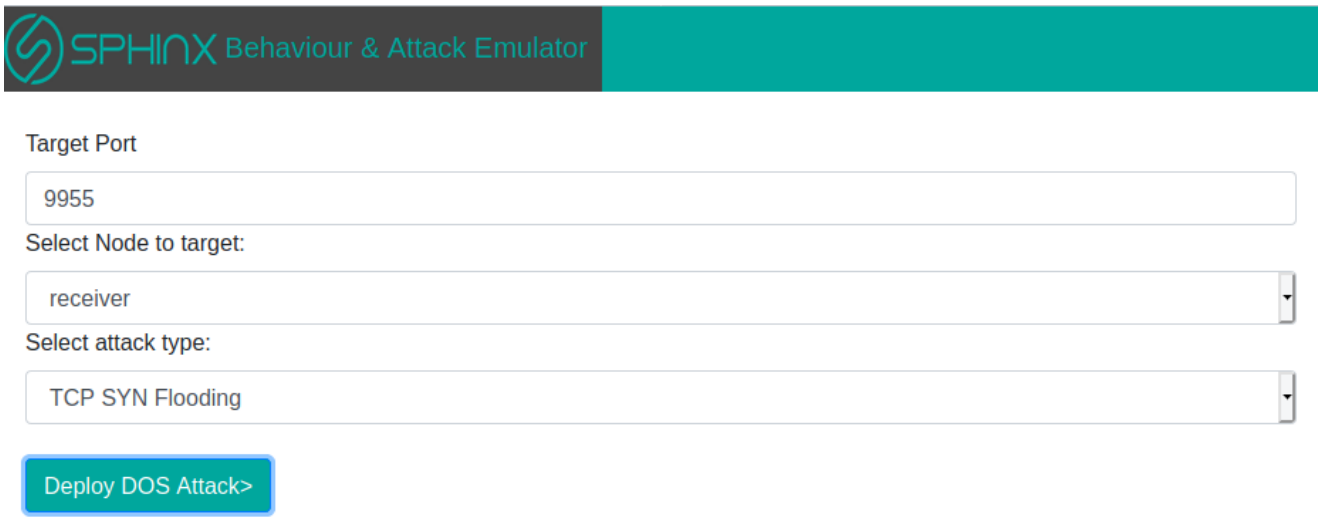

Figure 6: The ABS page providing the GUI for DoS deployment

More specifically an already implemented attack scenario, which has also been presented during the SPHINX review meeting in July 2020, consists of a network setting where two containers act as TCP or UDP client and server, exchanging normal network traffic while an attacking container, in this case based on the Kali distribution of Linux is performing DoS attacks as visualized in Figure 7. The user can use the abovementioned menu select amongst different DoS attacks such as TCP SYN and UDP floods as well as the targeted port of the victim node of the network.

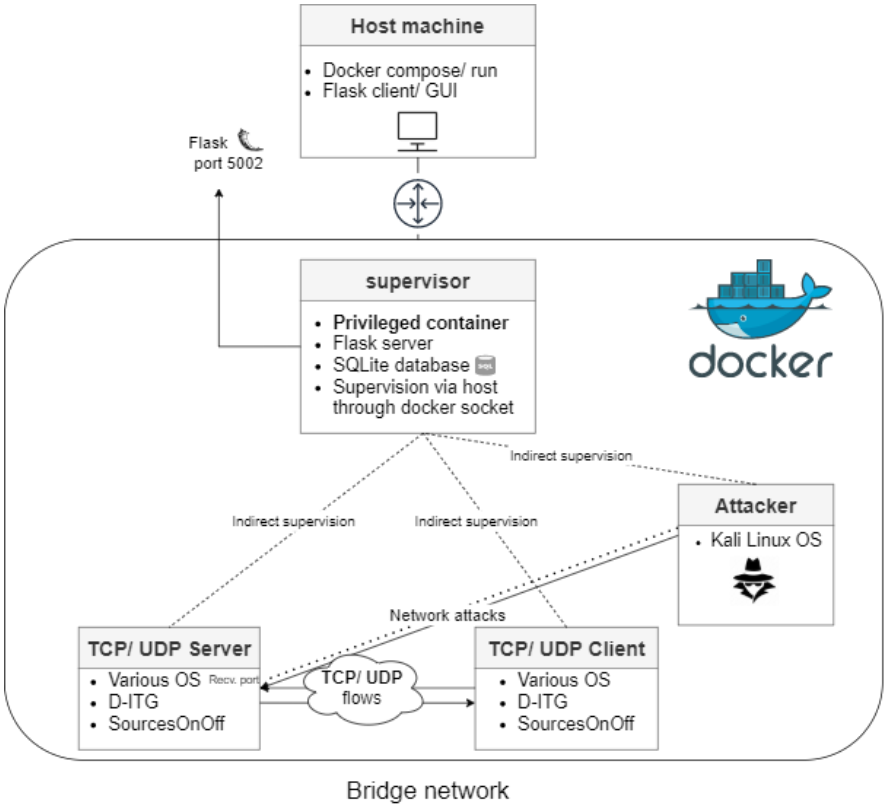

Figure 7: Example of Docker-based network topology for DoS deployment through the ABS.

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.

In Figure 8 the a terminal of the sniffing results at the victim node are presented (using the *tcpdump*[33] *sniffer*) along with an overview of the Docker resources usage (*docker stats*) during a UDP flood. It is obvious that the victim node receives overwhelming UDP traffic from spoofed IP addresses resulting to a congestion of the Network I/O resources of the Docker Engine which leads to a system slowdown and eventually to denial-of-service.

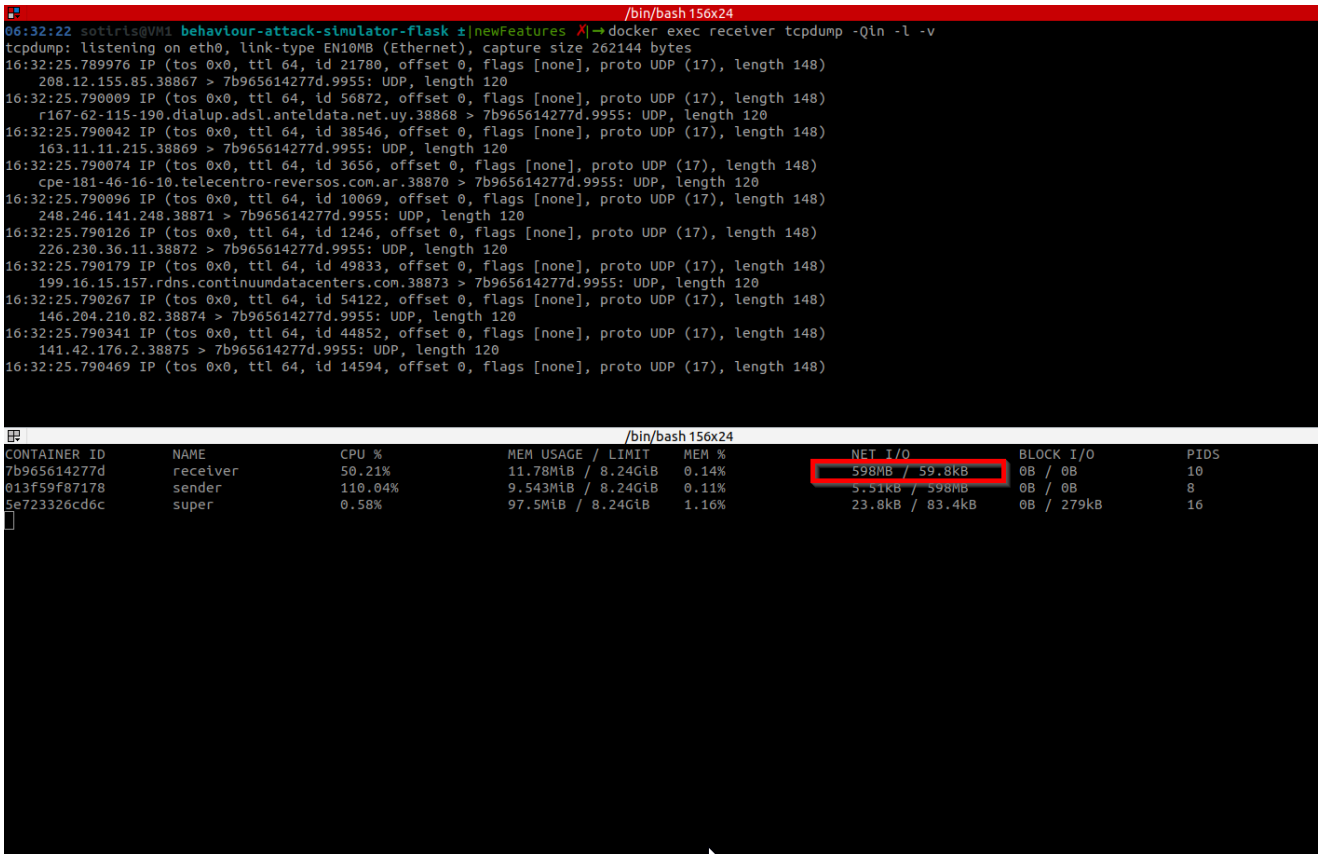

Figure 8: Overwhelming the resources of a receiving docker container during a UDP flood deployed by the ABS component

Until now, the ABS Attack Simulation implements a simulation of TCP floods and UDP floods that have been previously described in the context of DoS Attacks. In parallel, these attacks fall in the scope of UC7 that was presented in section 4.1.2.4. Of course, this is an early phase implementation which will be further be expanded after the creation and finalization of the experimentation platform.

Use cases that exhibit similar characteristics and can be executed similarly in the ABS component include: UC2, UC5, UC7, UC10, UC11, UC12.

4.3.2 Local Malware Execution

Stemming from the design principles of ABS component, the second of its parts is essentially a malware that relies on the immediate and direct actions of the users themselves in order to infect the system.

To implement these attacks, a list of executables shall be drawn tailored to each different example, which are going to be installed in a target virtual machine during the recreation of each scenarios. The installation will be done manually to better reflect the reality of the scenarios. If in a later stage it is deemed necessary, scripts shall be used to utilise a semi automate process.

[33] https://www.tcpdump.org/

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.

Following is an initial list of gathered malware that fulfils the needs outlines by the use cases. This is a live list of malware, that will continue to evolve along with the experimentation environment and the updated use cases deliverable.

Name	Source	Description
Bitcoin Miner	https://github.com/topics/bitcoin-mining	CPU based, basic bitcoin miner.
Sasser Worm	Metasploit Name: exploit/name/ftp/sasser_ftpd_port	Worm that infects computers running vulnerable versions of windows xp
Netcat	https://nc110.sourceforge.io/	Tool to open external tcp or upd connections
Diamoprhine Rootkit	Metasploit Name: exploit/linux/localdiamorphine_rootkit_signal_priv_esc	Elevate the privileges of arbitrary processes to UID 0 (root)
Capture: HTTP Javascript Keylogger	Metasploit Name: auxiliary/server/capture/http_javascript_keylogger	Web server that demonstrates keystroke logging through Javascript

Table 3 List of collected malware

Use cases with this type of method include: UC1, UC3, UC4, UC6, UC8, UC9

4.3.3 Abnormal Parametrisation of the Behaviour Simulator

The third and final part of ABS component is responsible for simulating the behaviour of malicious users who do not use any kind of malware. Instead these users, rely on actions allowed to them by the system to achieve their malicious goals.

In such cases, the implementation relies on the capabilities of the ABS component and the experimentation environment in order to produce this 'malicious' traffic and not actual malware. These actions will be replicated manually in the experimentation environment, but if it is needed, these can also be automated using scripts. Of course, this automated generated traffic will be different from the one normally generated by the behaviour simulator.

The replicated user actions will include:

- Browsing unusual websites
- Accessing the system's databases
- Uploading large amounts of data
- Downloading large amounts of data
- Deleting records
- Other functions allowed by the systems that can be exploited

A secondary use of the abnormal parametrisation of the behaviour simulator subcomponent, which may have some limited application, is the simulation of behaviour of malware in cases where it will not be possible to actually run the malware. In these cases, simulation its actions should produce enough evidence for their detection by the SPHINX modules. Currently, this secondary function is planned to be used in the replication of a single use case, but if required this function can be also used elsewhere.

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.

No actual implementation of this subcomponent currently exists since it requires some finalisation of the experimentation environment and the behaviour simulator.

Use cases with this type of method include: UC13, UC14, UC15, UC16.

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 Summary & Next Steps

This deliverable presented the level of research conducted throughout Task 5.3 of the SPHINX H2020 project, along with the overall development status of the respective submodules of the ABS component on M22 of the project's lifetime. Within this first research and development phase, the following accomplishments have been achieved:

- Identification of the methodologies typically employed to test and assess systems against cyber-attacks in the industry and their utilisation in order to form the unique methodology to be utilised by the Security Incident Attack Simulator.
- Identification of the SPHINX modules to be trained and tested, for their well-functioning, before entering the SPHINX final solution along with the methodological steps to achieve that.
- Identification of the SPHINX Use Cases requirements which result to a methodological plan of future design and development for the further steps of the Attack Simulator features, capabilities and back-bone attack simulation tools.
- Development of a user-friendly preliminary front-end application capable of creating network attacks and deploying them inside an simulated environment.

The final iteration of the security incident attack simulator will be presented in the demonstrator D5.8. This final iteration will evolve according to all the future iterations of the documents previously mentioned such D2.7 Use Cases definition and Pilot Overview document v2, D2.8 SPHINX requirements and guidelines v2, D2.9 Use Cases definition and Pilot Overview document v3 and D2.10 SPHINX requirements and guidelines v3.

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.

6 References

[1] "Software penetration testing - IEEE Journals & Magazine." https://ieeexplore.ieee.org/abstract/document/1392709 (accessed Oct. 30, 2020).

[2] "The Basics of Hacking and Penetration Testing - 1st Edition." https://www.elsevier.com/books/the-basics-of-hacking-and-penetration-testing/engebretson/978-1-59749-655-1 (accessed Oct. 30, 2020).

[3] "What is purple teaming and how can it strengthen your security?," *Redscan*, Jul. 12, 2018. https://www.redscan.com/news/purple-teaming-can-strengthen-cyber-security/ (accessed Oct. 30, 2020).

[4] S. Chowdhury, "PERCEPTIONS OF PURPLE TEAMS AMONG CYBERSECURITY PROFESSIONALS," thesis, Purdue University Graduate School, 2019.

[5] N. Alomar and E. Qiu, "Bug Bounty Hunter, Red Teamer, or Pen tester? A Closer Look at the Roles of Security Teams in Vulnerability Discovery," p. 5.

[6] "7 Steps of a Cyber Attack and What You Can Do to Protect Your Windows Privileged | BeyondTrust." https://www.beyondtrust.com/blog/entry/7-steps-cyber-attack-can-protect-windows-privileged-accounts (accessed Oct. 30, 2020).

[7] "The U.S. Intelligence Community," *Routledge & CRC Press*. https://www.routledge.com/The-US-Intelligence-Community/Richelson/p/book/9780813349183 (accessed Oct. 30, 2020).

[8] S. Behal, K. Kumar, and M. Sachdeva, "Characterizing DDoS attacks and flash events: Review, research gaps and future directions," *Comput. Sci. Rev.*, vol. 25, pp. 101–114, Aug. 2017, doi: 10.1016/j.cosrev.2017.07.003.

[9] W. Chen and D.-Y. Yeung, "Defending against TCP SYN flooding attacks under different types of IP spoofing," in *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*, 2006, pp. 38–38.

[10] M. Bogdanoski, T. Suminoski, and A. Risteski, "Analysis of the SYN Flood DoS Attack," *Int. J. Comput. Netw. Inf. Secur. IJCNIS*, vol. 5, no. 8, Art. no. 8, Jun. 2013.

[11] M. Kührer, T. Hupperich, C. Rossow, and T. Holz, "Hell of a Handshake: Abusing {TCP} for Reflective Amplification DDoS Attacks," presented at the 8th {USENIX} Workshop on Offensive Technologies ({WOOT} 14), 2014, Accessed: Oct. 13, 2020. [Online]. Available: https://www.usenix.org/conference/woot14/workshop-program/presentation/kuhrer.

[12] M. Kenney, "Ping of death," *Insecure Org*, vol. 2, 1996.

[13] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, 2014.

[14] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in *ICISSP*, 2018, pp. 108–116.

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.