ENI ISG - PoC Proposal Template

1 PoC Project Details

1.1 PoC Project

PoC Number (assigned by ETSI):

PoC Project Name: Securing against Intruders and other threats through a NFV-enabled Environment (**SHIELD)**

PoC Project Host: Telefonica

Short Description:

This PoC has the aim to demonstrate how an AI based network security management framework operates against different types of threats. This will be achieved through the combination of orchestration in a NFV-based environment, AI to detect and report attacks and an intent based automatic policy decision and enforcement.

Particularly, the PoC will demonstrate the ENI architecture concepts by combining tailored security Virtualized Network Functions (VNFs) or virtualized Network Security Functions (vNSF) to monitor and mitigate threats in the network traffic, a Machine Learning Framework, with specific algorithms that detect attacks, and a intent based policy language that automatically enforce the policies in vNSF by following policy-driven closed control loops as proposed by ENI.

The resulting framework implementation is called SHIELD, and it has been developed as part of the EU H2020 project SHIELD (https://www.shield-h2020.eu/)

1.2 PoC Team Members

Table 1.1

	Organization name	ISG ENI participant (yes/no)	Contact (Email)	PoC Point of Contact (see note 1)	Role (see note 2)	PoC Components
1	Telefonica	yes	Diego R. Lopez diego.r.lopez@telefonica.com Antonio Pastor antonio.pastorperales@telefonica.com	X	Network Operator	-Use case definitions -Business model definition
2	Space Hellas	no	Georgios Gardikis ggar@space.gr		Manufacturer	- Integrator - VNFs provider - DARE (see Note 3)
3	ORION	no	Olga Segou osegou@orioninnovations.gr		Manufacturer	- Integrator - VNFs provider
4	Demokritos (NCSR)	no	Eleni Trouva trouva@iit.demokritos.gr		University	- Integrator - VNFs provider

NOTE 1: Identify the PoC Point of Contact with an X.
NOTE 2: The Role will be network operator/service provider, infrastructure provider, application provider or other as given in the Definitions of ETSI Classes of membership.
NOTE 3: DARE stands for Data Analysis and Remediation Engine.

All the PoC Team members listed above declare that the information in this proposal is conformant to their plans at this date and commit to inform ETSI timely in case of changes in the PoC Team, scope or timeline.

1.3 PoC Project Scope

1.3.1 PoC Goals

The PoC will propose and demonstrate a specific implementation of a security use case. The use case is aligned with [UC#2.4 Policy-based network slicing for IoT security] but may also be considered as aligned with parts of other use cases, e.g. where mitigation requires modification of the traffic. In particular, the SHIELD framework will focus on Artificial Intelligence (AI) cyber-attack detection to identify some attackers and isolate them. Also, it will address multiple requirements [GR.1,GR.2, SOM.1, SP.1, SP.2/A/B, SP.3, SP.4, SP.5, DCA.2., DCA.4, GPM.1, GPM.2, GPR.5, DL.1, IWOS.1, OR.3]

PoC Project Goal #1: Demonstrate through a practical implementation how an AI framework can detect network attacks over an NFV network, with different Machine Learning (ML) algorithms and their combination.

PoC Project Goal #2: Demonstrate a practical framework of a policy-driven control loop, by combining AI-based attack detection and proposing mitigation through an intent-based security policy to the network operator. This is accomplished by applying it through the translation into a specific configuration of VNFs.

PoC Project Goal #3: Demonstrate how the use of remote attestation[1] technology, i.e. a method by which a host (client) authenticates its hardware and software configuration to a remote host (server), allows to avoid device and data collecting corruption and tampering.

1.3.2 PoC Topics

PoC Topics identified in this clause need to be taken for the PoC Topic List identified by ISG ENI and publicly available, i.e. the three topics identified in clause 4.5 of the ENI PoC Framework. PoC Teams addressing these topics commit to submit the expected contributions in a timely manner.

Table 1.2

PoC Topic Description	Related WI	Expected Contribution	Target Date See Note 4)
Network security _-> Policy-based network slicing for IoT security	ENI 008	Propose AI managed security service use case Feasibility of an intent based policy management	1/04/2019
Service and network requirements -> Security and privacy	ENI 007	Propose attestation requirements to improve security in ENI (Trust Monitoring)	1/04/2019
Cognition (global)	ENI 005	Integration of different ML algorithms	1/04/2019
NOTE 4: The dates are indicative			

[1] ETSI GR NFV-SEC 007 and https://www.ietf.org/id/draft-pastor-i2nsf-nsf-remote-attestation-06.txt

1.4 PoC Project Stages/Milestones

Table 1.3

PoC Milestone	Stages/Milestone description	Target Date	Additional Info
P.P.1	PoC Presentation	4/12/2018	Presentation during #ENI8
P.S	PoC Project submission	18/12/2018	Official PoC submission
P.PU	PoC user story detailed	15/01/2019	Selecting and detailing SHIELD use case storyboard for public demonstration to fit in existing ENI use case (UC#2.4). Contribution to ENI use case WI
P.P.2	PoC public announce	22/01/2019*	Public Web announce in SHIELD media (web, twitter, etc.), *Once it is approved
P.PT	PoC Testing phase	22/02/2019	Testbed setup and running
P.D1	PoC Demo 1	25/02/2019	ICISSP-2019 workshop and demo booth
P.C.2	PoC Expected Contributions	1/04/2019	Propose contributions on Table1.2 topics at ENI#9 or before, addressing ENI 005, ENI 007 and ENI 008
P.R	PoC Report	1/04/2019	PoC-Project-End Feedback with final presentation.
P.E	PoC Project End	1/04/2019	
NOTE 5: The dates are indicative			

1.5 Additional Details

Several additional demos are still in discussion within EU H2020 SHIELD[2].

H2020 SHIELD web portal details information and include multiple references to implementation[3] and design.

2 PoC Technical Details

2.1 PoC Overview

The SHIELD project aims at securing against intruders and other threats through a vNSF-enabled environment using a Cognitive engine. SHIELD PoC will use the architecture and components described below to demonstrate the capacity to detect and mitigate attacks. See also Figure 1, which contains the SHIELD architecture overview and reference points, as well as Tables 2.1 and 2.2 in subclause 2.2.8, which contain the mapping between the ENI Functional Blocks and ENI external reference points with the SHIELD PoC. Specifically, we can demonstrate how to stop a cyber-attack by isolating a device after the DARE (details in PoC Architecture), acting as an ENI system, detects the attack in the network. Upon the detection, it proposes a solution to the network administrator and executes confirmed actions automatically on the network following the defined policy(ies).

The PoC combines different components over a NFV architecture. A specific type of VNF (vNSF) is capable of monitoring the network based on network flows. This information is fed to the Cognitive engine that uses anomaly-based unsupervised engine[4] (automatic identification of asymptomatic items from the majority of the data) combined with a supervised learning model[5] (train algorithms to detect specific patterns) within a distribute processing

[2] https://www.shield-h2020.eu/

[3] https://github.com/shield-h2020

[4] https://www.datascience.com/blog/python-anomaly-detection

[5] https://en.wikipedia.org/wiki/Supervised_learning

architecture . As a result, an attack identification happens, and the cognitive response includes a “recipe” or policy to mitigate. This security policy is presented to the network administration as an enhanced solution. The administrator can then accept this policy to be applied. The enforcement process involves identification of the best VNF with mitigation capacities (e.g. firewalls) and their specific configuration tailored to the vendor language. This enforcement isolates the infected IP/device.

Two different use cases can be addressed:

- Denial of Service (DoS): Following use case [UC#2.4 Policy-based network slicing for IoT security] a compromised/malicious device generates a DoS attack based on resource consumption. Dare acting as ENI system can detect and request to block the attacker IP.
- Ransomware attack: In this case, the attack is performed by a worm that expand over a network, infecting PCs and other devices to encrypt all data. Finally, it will request a ransom for the decryption key. DARE acting as ENI system can detect and request to isolate the infected device in early stages.

More detailed information concerning the operation of the PoC engine may be found in the next subsection. In addition, references in section 1.5 should also be taken into account. For even deeper information, source code is available on the SHIELD web portal (https://www.shield-h2020.eu/).

2.2 PoC Architecture

The SHIELD architecture is articulated around different components, illustrated in Figure 1. This PoC will leverage this architecture, focusing on the components related with the ENI system (performed by DARE and its interfaces) to implement the security use case.

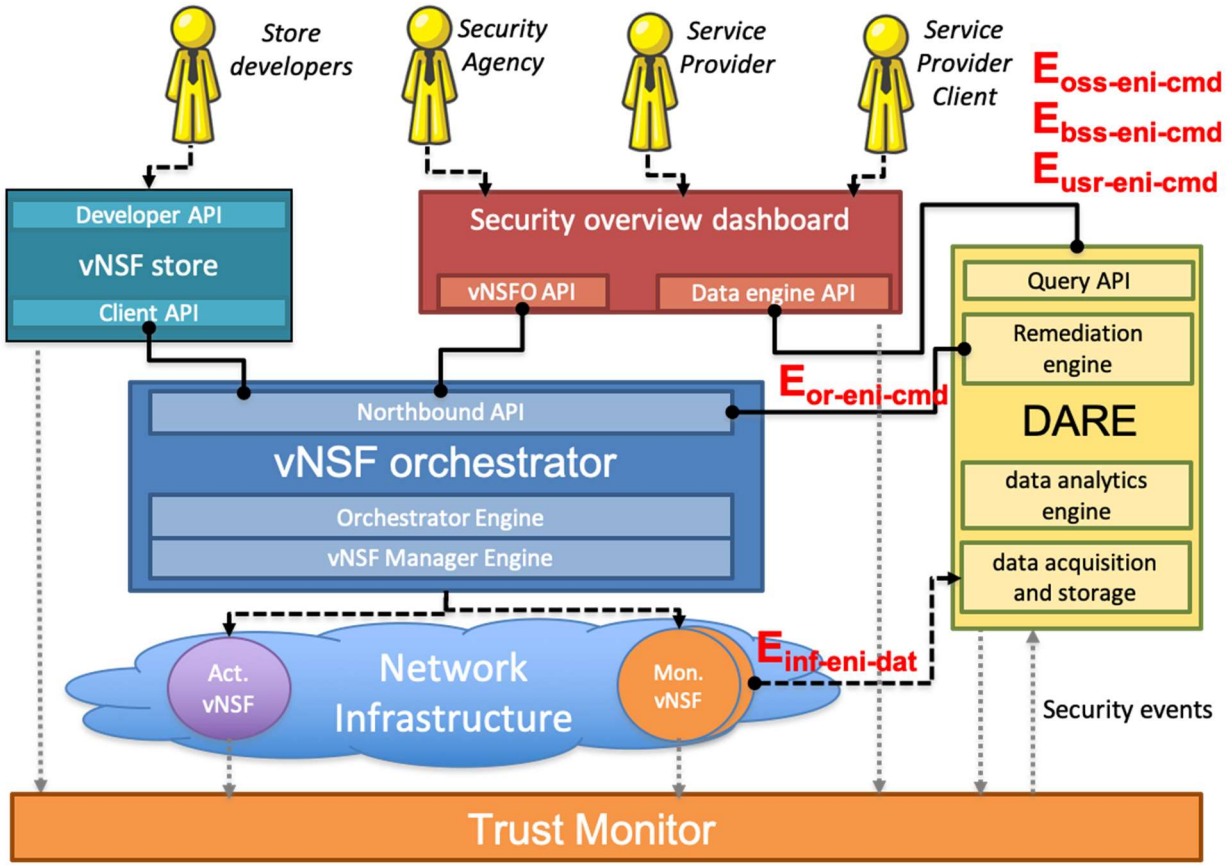

Figure 1 - SHIELD architecture overview and reference points

In a nutshell, the Network Infrastructure is the running space for the vNSFs; the DARE stores and analyses the security logs and events provided by the former to derive intent-based policies shared with the dashboard; and finally, the security dashboard presents the results to the operator. These core components are supported by: i) the vNSF store,

which holds the vNSFs images; ii) the vNSFO (NFV MANO-based orchestrator), which manages the Network Services (NSs) and their vNSFs; and iii) the Trust Monitor, which verifies if the SHIELD platform is trusted at all time.

2.2.1 Network infrastructure

The network infrastructure supports the instantiation of vNSFs. When creating a NS, the vNSFs can be considered as security appliances dynamically deployed on the network infrastructure. SHIELD identifies two main types of vNSFs:

1. **Monitoring vNSFs** are devoted to gather information about the network, generate alarms and triggers in case of ongoing attacks.
2. **Acting vNSFs** apply the necessary mitigations to pre-empt attacks and protect against known vulnerabilities and threats, or mitigate them as a security incident evolves. The proper acting vNSF is chosen depending on the kind of threat detected (if not already present).

The network infrastructure interacts with the Trust Monitor functional block in order to attest the integrity of each network functional blocks. The network infrastructure, consisting of network devices and other virtualized network infrastructure components, is orchestrated by the vNSFO (MANO-based orchestrator) with the support of a VIM (Virtual Infrastructure Manager). The vNSFO also orchestrates the vNSFs lifecycle management, such as deploy, configure, destroy, monitoring and data collection, with the help of an internal vNSF Manager (vNSFM) engine. Monitoring vNSFs inspect captured data and provide valuable information to the Data Service Engine component of the DARE. The network status is reported periodically since more complicated events (i.e. an attack using multiple vectors) could sometimes not be detectable by individual vNSFs but can be inferred by the DARE. These interactions are illustrated in Figure 1.
According to the ETSI NFV specifications, the network infrastructure layer includes the physical and virtual nodes (commodity servers, VMs, storage systems, switches, routers etc.) on which the services are deployed.

2.2.2 Virtual Network Security Function (vNSF)

The vNSF ecosystem can be highly heterogeneous. Yet, some constraints must be met in order that vNSFs can securely interact with the platform. Each vNSF has a series of logical interfaces for management: one interface is used by vNSFO for configuration and control connections; another interface is used for communication with the DARE, in order to to report incidents and metrics; finally, a third interface should be used for attestation operations only, where available, see dotted grey arrow to Monitor in Figure 1. Related to this last interface, see references to ETSI NFV and IETF documents under Table 1.2 for further details.

Examples of existing vNSF are: virtualized intrusion detection system (vIDS), Layer 3 Firewall, Layer 7 firewall, and Proxy Transport Layer Security (TLS).

2.2.3 vNSF orchestrator (vNSFO)

The vNSFO is responsible for managing the lifecycle of NSs. To that end, the vNSFO interacts with each of the other modules to obtain data from the vNSFs, to receive deployment and configuration requests, or to convey data of running nodes. Internally, the orchestration engine communicates with the vNSF Manager (vNSFM) engine to delegate the management of the vNSFs that are part of the requested NS. vNSFO & vNSF Manager use ETSI OSM (Open Source MANO) implementation and VIM is delivered through Openstack.

2.2.4 vNSF store

The vNSF store acts as a nexus between the vNSFO and the developer, which can register, manage vNSFs and make them available for later use. The following data is provided to the store: i) the network service descriptor (contains deployment requirements and any other metadata required by vNSFO and VIM), ii) the software images (contain the functionality that is instantiated), and iii) the security descriptor (contains information to validate its integrity).

The store provides two interfaces to cover this functionality:

- the Developer API provides interaction with the vNSF developer. It allows to:
  - upload a new NS or vNSF,
  - update its information and remove it,
- the Client/deployment API provides interaction with the vNSFO.

2.2.5 Trust Monitor

The Trust Monitor is the functional block in charge of monitoring the trustworthiness of the SHIELD infrastructure. This is achieved by a combination of authentication and integrity verification techniques standardized by the Trusted Computing Group (TCG). Each NFVI node, being equipped with a Trusted Platform Module (TPM) and suitable software, must be properly authenticated using hardware-based cryptographic identities from TPM. In addition, each node provides a proof of the integrity of its software stack by using the Linux Integrity Measurement Architecture (IMA).

2.2.6 Data Analysis and Remediation Engine (DARE)

The Data Analysis and Remediation Engine (DARE) is an information-driven Intrusion Detection and Prevention System (IDPS) platform that stores and analyses heterogeneous network information, previously collected via monitoring vNSFs. It features cognitive and analytical components capable of predicting specific vulnerabilities and attacks, see Table 2.1. The processing and analysis of large amounts of data is carried out by using Big Data, data analytics and machine learning techniques. By processing data and logs from monitoring vNSFs deployed at strategic locations of the network, the DARE components provide information for the development of cybersecurity topologies (mitigation recipes), meaning that in case malicious activity is detected, they implement remediation activities, by recommending policies through the means of a dashboard and accessible API.

The DARE consists of three main components: the data acquisition and storage module, the data analytics engine and the remediation engine, see Table 2.1.

The data acquisition and storage module is responsible for the ingestion of the selected datasets and their preparation for further processing. This module is composed of different types of data collectors and workers following the Apache Spot[6] architecture. It therefore supports network flow, Domain Name System (DNS) and web proxy logs collection and transformation. The data acquisition process implements a data streaming service or data bus (i.e. Apache kafka subscriber) that collects data provided by monitoring vNSFs, and storage it in a distributed storage system like Apache Hadoop Distributed File System (HDFS).

The data analytics engine module produces packet and flow analytics by using scalable machine-learning techniques. To this end, it involves the latest distributed computing technologies (Apache Spot, Spark, Hadoop Distributed File System (HDFS), Kafka, Hive). The threat detection procedure of the Cognitive module is based on the Apache Spot framework and uses a combination of machine learning tools to run scalable machine learning algorithms (e.g. Latent Dirichlet Allocation – LDA[7] for network flows fields), not only as a filter for separating malicious traffic from the benign one, but also as a way to characterize the unique behaviour of network traffic.

Finally, **the Remediation engine** uses the analysis from the data analytics modules and is fed with alerts and contextual information (by configuration or by the vNSFO) to determine a mitigation plan for the existing threats. It performs in near-real-runtime, and generates a cybersecurity topology for a detected threat. That topology is converted first into a HSPL (High-level Security Policy Language) and later into a MSPL (Medium-level Security Policy Language), which are included in a remediation recipe. The Remediation Engine's main goal is to incorporate a combination of recommendations and alerts that provide relevant threat details to all interested parties by using the dashboard and the direct application of countermeasure activities by triggering specific vNSFs configuration (Action vNSFs), via the vNSFO, upon confirmation by the user. One example is an HSPL recipe with an intent based security policy to block an infected device. The HSPL is translated to MSPL, agnostic to the technology, to add a new filter rule for an IP address. Remediation engine send this info to the Dashboard (query API) and it is confirmed. At the end the MSPL information is sent, through the vNSFO API, to a Firewall vNSF, which translates the MSPL to its specific configuration and set the new firewall rule to block the malicious IP address. This change will leverage the vNSFO API that control the vNSF.

2.2.7 Security dashboard

The SHIELD platform provides an intuitive and appealing graphical user interface. From this dashboard, operators have access to monitoring information showing an overview of the security status as well as allowing operators to take actions and react to any detected problem. It provides security-related features comprising vNSF and NS lifecycle

[6] http://spot.incubator.apache.org/

[7] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022.

management, threat detection notifications, threat mitigation actions evaluation and application, untrusted nodes alerts, service status, either by interacting with the platform itself or through REST API tailored for such purpose. (vNSFO API and Data Engine API)

2.2.8 Relationship with DGS ENI-005 Architecture and Reference points

SHIELD NFV environment (vNSF Store, vNSFO, network infrastructure and NFVI) is considered as a Class 1 Assisted System (an Assisted System that has no AI-based capabilities) from the point of view of the ENI system architecture. This is the current situation of the ETSI MANO functionality, NFVO has not embedded AI engine, but an orchestration capacity based on External Designated Entity of the Assisted System (i.e., the operator, OSS, BSS).

Regarding the Operation Mode, SHIELD PoC behavioural is considered as "recommendation mode" [MOP.1]. This mode has been carefully selected in order to allow the operator to acknowledge the mitigation actions, and avoid automatic responses specially if we consider the legal aspect related to the observation and modification of client's traffic [MOP.5.2]. A security problem detected by DARE is reported to the OSS Dashboard including a recommendation action, which contains a policy change proposal, to solve the problem [MOP.7]. The Designated Entity has the final word in terms of acceptance or rejection of that proposal.

DARE component on SHIELD PoC follows the ENI architecture. Table 2.1 shows the mapping between ENI architecture Functional Blocks and SHIELD PoC.

Table 2.1 - Mapping between ENI Functional Blocks and SHIELD PoC

ENI Functional Blocks	PoC SHIELD Components
Knowledge Representation and Management	**DARE remediation engine**. Based on predefined recipes
ENI Ingestion and Normalisation	**DARE data acquisition and storage module**. Support several ingestion types
Context-Aware Management	**DARE remediation engine**. Limited to some rules based on predefined recipes
Situational Awareness	**DARE remediation engine**. Limited to some rules based on predefined recipes
Policy Management	**DARE remediation engine**. Use Intend based language: HSPL (High-level Security Policy Language) and MSPL (Medium-level Security Policy Language)
Denormalisation and Output Generation	**DARE Query API**. HSPL & MSPL is sent directly to the Dashboard and vNSF. Denormalization is done by each vNSF
Cognition Framework	**DARE data analytics engine**
Lifecycle Management	N/A
Ancillary	N/A

Reference points used in SHIELD PoC are shown in the Figure 1. Table 2.2 lists a brief description of each one.

Table 2.2 - Description of ENI external reference points used by SHIELD PoC

ENI External Reference Points	Description
$E_{oss\text{-}eni\text{-}cmd}$ $E_{bss\text{-}eni\text{-}cmd}$ $E_{usr\text{-}eni\text{-}cmd}$	Dashboard receive MSPL recommendations for security threats. The dashboard interacts and help to accomplish multiple functionalities: - OSS: Monitoring ISP security threats - BSS: Billing system for ISP, to allow use or not of vNSFs - User: controlling Service provider Clients
$E_{inf\text{-}eni\text{-}dat}$	DARE using streaming service collect data for security monitoring

$E_{or\text{-}eni\text{-}cmd}$	DARE interacts with vNSFO to request contextual information

2.3 PoC Success Criteria

All goals are met when the described functionality is proved to be available. More specifically:

- PoC Project Goal #1: Demonstration that at least one specific cybersecurity related attack is detected using different Machine learning algorithms and their combination based on data collected from the network following the PoC architecture described.
- PoC Project Goal #2: Upon AI engine detection of an attack, availability of a couple of intent-based security policies to isolate the attack, and demonstration of how at least one is translated into a specific vNSF configuration (e.g. L3 firewall rules).
- PoC Project Goal #3: Demonstration of a remote attestation process with capability to detect a device corruption or tampering. Trust Monitor (client) attest remotely the vNSFs (servers).