D2.3.2

Specification of the SECURED architecture (beta version)

Project number	611458
Project acronym	SECURED
Project title	SECURity at the network EDge
Project duration	36 months (1/10/2013–30/9/2016)
Programme	FP7 (Collaborative Project)

Deliverable type	**R** - Report
Deliverable number	D2.3.2
Version (date)	v1.1 (30/09/2015)
Work package(s)	WP2
Due date	30/09/2015 – M24

Responsible organisation	HPLB
Editor	Adrian L. Shaw
Dissemination level	**PU** - Public

Abstract	This document describes the alpha specification of the SECURED architecture. It provides full descriptions of the functional requirements, component descriptions, and the required interactions with the SECURED infrastructure.
Keywords	architecture, security, requirements, definitions, PSA, VNF

Editor
Adrian L. Shaw (HPLB)

Reviewers
Diego R. Lopez (TID)
Michael Georgiades (PTL)
Antonio Lioy (POLITO) – quality control

Contributors
Adrian L. Shaw (HPLB)
Ludovic Jacquin (HPLB)
Antonio Lioy (POLITO)
Christian Pitscheider (POLITO)
Cataldo Basile (POLITO)
Fulvio Risso (POLITO)
Roberto Bonafiglia (POLITO)
Francesco Ciacca (BSC)
Mario Nemirovsky (BSC)
Jarkko Kuusijärvi (VTT)
Diego Montero (UPC)
René Serral-Gracià (UPC)
Francesca Bosco (UNICRI)

Acknowledgement
This work was partially supported by the European Commission (EC) through the FP7-ICT programme under project SECURED (grant agreement no. 611458).

Change Log

Version	Date	Note	Author
v1.0	25.09.2015	First version	A.Shaw (HPLB)
v1.1	05.11.2015	Quality control	A.Lioy (POLITO)

Executive Summary

This document provides an early preview of the basic SECURED architecture. As such, it is to be considered as work in progress and therefore contains some elements which are subject to change.

The document provides a general architectural framework for application offload from user terminals into the network, combined with support for security policies, and lists the functional requirements of all required components.

The overview is presented in a technology-agnostic fashion. All technology-dependent and implementation related details are addressed in deliverables provided by the following work-package in SECURED: WP3 (NED architecture), WP4 (Policies) and WP5 (Applications and Services).

With respect to the alpha version of this specification, the current beta version has added three main elements:

- the introduction of on-line and off-line workflow managers (sections 4.3.1 and 4.3.2);
- various optimisations to better support seamless switching in a mobility scenario (section 8);
- a discussion about keys and certificates in Appendix C – Keys and certificates life-cycle.

Contents


SECURED

1 Introduction

The protection of Internet-connected devices, such as mobile phones and laptops, has typically been achieved through installing appropriate specific tools on each device (e.g. a personal firewall, malware analyser, parental control application). However, this raises several issues: privileged access on the device is often required, tools may use a large amount of computational resources, platform capabilities vary, and appropriate tools may be unavailable on particular systems. Moreover users need to configure each security control on each of their devices, resulting in an unsolvable maze for non-technically savvy users. This results in ineffective or inconsistent protection for users who use different devices across different networks (e.g. border firewalls are available in corporate Wi-Fi environments, but the user is not protected over a 3G network). In addition to this problem, there is an escalating number of embedded devices connecting to the internet, of which many contain little to no security installed. This includes sensors, controllers and appliances which fall under the broad definition of the Internet of Things (IOT). Furthermore, the inconsistent set of security controls for a large number of devices also creates a policy management and enforcement nightmare in an age where Bring Your Own Device (BYOD) is becoming increasingly popular in the workplace. The SECURED project aims to provide an innovative architecture to achieve device protection from Internet threats by offloading execution of common security applications away from user devices and into a programmable device at the edge of the network. This Network Edge Device (NED) hosts a common set of a security applications, providing a uniform security control point for all connected devices. These secure and trusted NEDs can act in several different scenarios, such as in home gateways, enterprise routers, mobile or public access points. The architecture allows for different users (e.g. individual users, corporate ICT managers and network providers) to install on-demand and execute Personal Security Applications (PSA) within their own trusted and virtualised execution environment. The NED securely hosts and isolates the user environments, and it also isolates the traffic flows of different users. Users are able to monitor their PSAs through their own trusted Personal Security Controller (PSC) on the NED. Thus, the security architecture of the NED must be carefully designed to securely address the different requirements.

A management framework is also required for securely deploying and configuring PSA applications based on well-defined policies defined by the end user. Traditional policy problems typically range from confusing definitions to enforcement ambiguities. The architecture supports scenarios for multi-tier policy definition languages, each aimed at different sets of stakeholders, e.g. one aimed for non-technical end users and one for technically knowledgable developers and users. Reconciliation of potential policy conflicts is also addressed as part of SECURED.

1.1 NED models

The project anticipates different flavours of a NED, which can cover many different network scenarios. Figure 1 presents the possible implementation options of the NED according to three orthogonal dimensions, namely the hardware architecture, the type of deployment, and how the user traffic is delivered to the NED.

The first dimension considers two possible hardware options: special purpose components specifically engineered for data plane processing (e.g. network processors, hierarchical memory architectures, hardware accelerators) versus general purpose standard high-volume components (e.g. general purpose processors, mainstream memories). The former is more appropriate for high speed processing, while the latter offers a better price/performance ratio and looks more appropriate to integrate the NED in a Network Functions Virtualisation (NFV) cloud-like infrastructure. A complete software version will

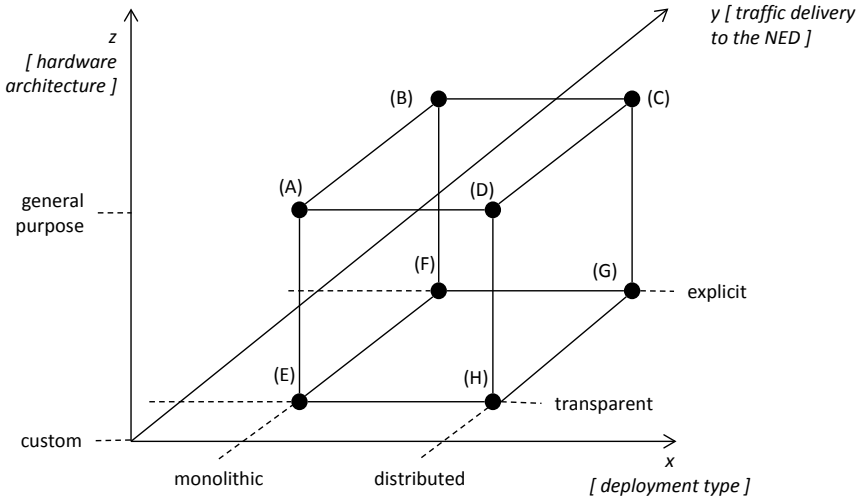

Figure 1: Dimensions for the possible NED deployment scenarios.

also be available, which can be deployed on conventional computer hardware, such as a server or a spare personal computer which has sufficient computational ability; located in the same network as the protected devices, the user only diverts their traffic to it before it is processed and forwarded to the internet. This is referred to as a Virtual NED (vNED). Overall, whether through discrete hardware or virtual, these NEDs share the same logical architecture and specification.

Concerning the second dimension, the NED is distinguished as a monolithic component (e.g. a switch or a server) that implements all the core functions from the case in which the NED functions are distributed across multiple components. For example, a traditional router that does not have advanced computing capabilities might redirect the user traffic (e.g. based on a protocol like OpenFlow [1]) to a server that takes care of the required processing. The monolithic flavour looks simpler to deploy and manage (e.g. the procedure to verify the hardware/software integrity must handle a single component). Instead, the distributed model can guarantee better scalability and is oriented towards a cloud-like or NFV environment.

The third dimension refers to the way the user traffic is redirected to the NED. While the preferred incarnation of this project assumes that the network is SECURED-aware and hence the traffic is automatically handled by the (first) network device encountered ("transparent" traffic steering), it is foreseen also the case of a user connecting over an untrusted or not SECURED-aware network. In this particular case it is foreseen that there would be a need for a small agent operating on the user device in order to establish a secure tunnel to a remote NED and subsequently deliver all the user traffic to it (using "explicit" traffic steering).

1.2 Use cases

In this subsection, use cases are presented to understand and define trust zones, user expectations and likely technological scenarios. Each describes the likely user devices, the type and number of users, their technical skills, and foreseen networking and computing capabilities. A distinction is made between a network provider (such as an ISP) and the SECURED provider. As a minimum, two business cases are supported:

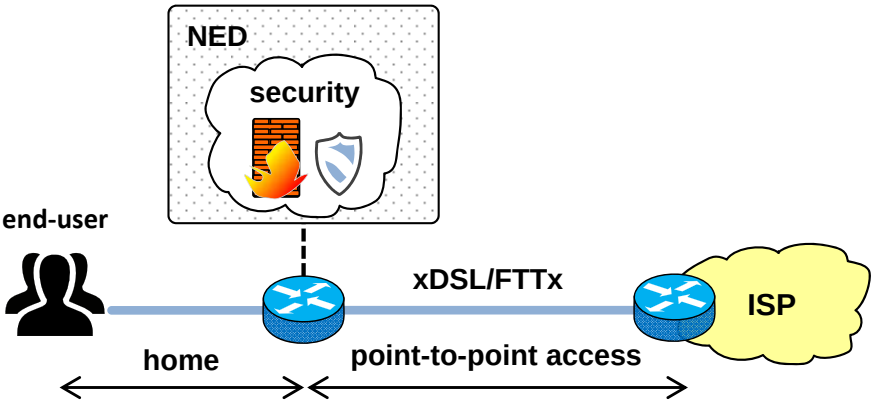

Figure 2: Security applications hosted at home-gateway NED.

1. the primary case is that the SECURED provider is also the ISP (e.g. Telefonica offering SECURED services to customers on their own network);
2. the secondary case is where the ISP is not involved in providing the SECURED service. An example would be a small business which may have an existing network provider but may either be providing services itself on a local NED infrastructure or by outsourcing managed NEDs to a third-party services company).

Another assumption is that the mobile provider is the same as the Internet provider (at home or enterprise). Other cases are considered later in the project.

1.2.1 Home network

Home users who access the Internet via xDSL, using a local network via wireless or cable, with several devices (desktop, laptop, tablet, smartphone, smart TV, ...). There are few and known users (e.g. family members and visitors), low technical skills, and limited local networking and computing capabilities.

Home gateway. This subscenario considers a NED that is placed within the home of the end user, which has either been specifically purchased by the home user or it has been provided for free by the ISP. The user is the NED owner, and thus is the administrator of the NED. Figure 2 shows the NED acting as the local access point within the home environment.

ISP hosting. Since many home users have direct, fixed xDSL connections to a place in the ISP network, there are opportunities for network providers to offer hosting of SECURED security services without the need for a powerful NED gateway like in the Home gateway scenario. This is illustrated in Figure 3.

1.2.2 Enterprise

Employees access a corporate LAN via cable or wireless using multiple devices, of which are primarily workstations. The enterprise may contain a high number of users (hundreds to thousands, including

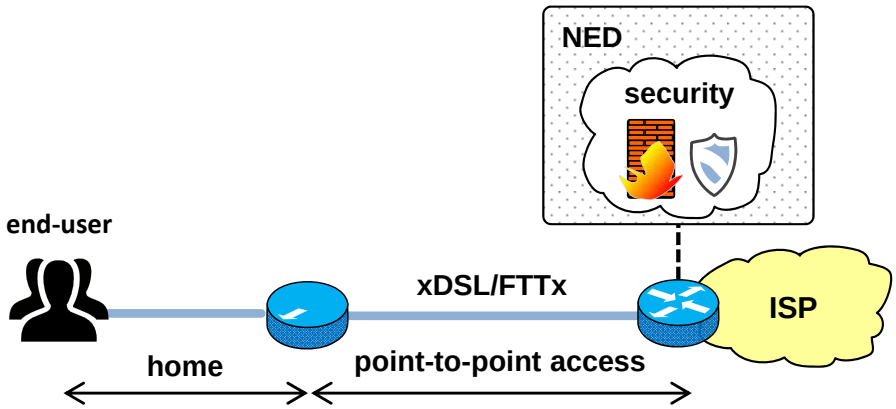

Figure 3: Security applications hosted at ISP NED.

visitors and contractors, of which most are known and registered). It is expected in this scenario that there is a sufficient number of available system administrators are available who have highly technical skills, and the enterprise has powerful networking and computing capabilities. NEDs are placed either locally to the users or near border gateways, depending on the security and trust model of the organisation. These differences are depicted in Figures 4 and 5.

Hosted on site. Protects all employee traffic through appropriately placed NEDs, or redirect all traffic flows to local servers with large computational capability.

Outsourced security. Hosted by an external entity, such as an ISP. Particularly fitted for small and medium businesses that do not own a capable IT infrastructure, yet are looking for a unique BYOD security solution for highly mobile employees (phone, tablet, laptop). This may be achieved through a contractual agreement between a business and a mobile network provider.

1.2.3 Public hotspot

Citizens access the Internet via WiFi access-point (e.g. at a cafeteria, railway station, airport, ...), where there are many other users (unknown and changing frequently), few devices (just laptop, tablet and smartphone), low local technical skill and networking/computing facilities (but ISP may provide additional skills and facilities). This scenario considers that NED will be part of the access-point infrastructure (the NED acting as the access point) or traffic is forwarded to a nearby vNED.

1.2.4 Mobile

Customers access the Internet via 3G/4G, using few devices (just laptop, tablet and smartphone), a single known and registered user per device (but many known and registered users in the mobile network), no local technical skill, no local network, no local computing facility. Everything is provided by the telco/ISP, including the NED devices. The mobile access network is totally controlled by the mobile provider, and there is a strict contractual relationship between user an provider. In functional terms, it is equivalent to the enterprise or home networked environments.

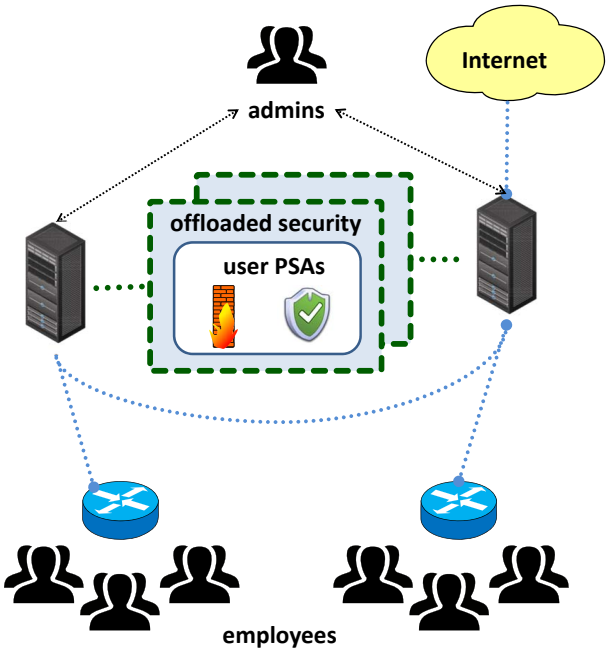

Figure 4: Traffic forwarded to vNEDs.

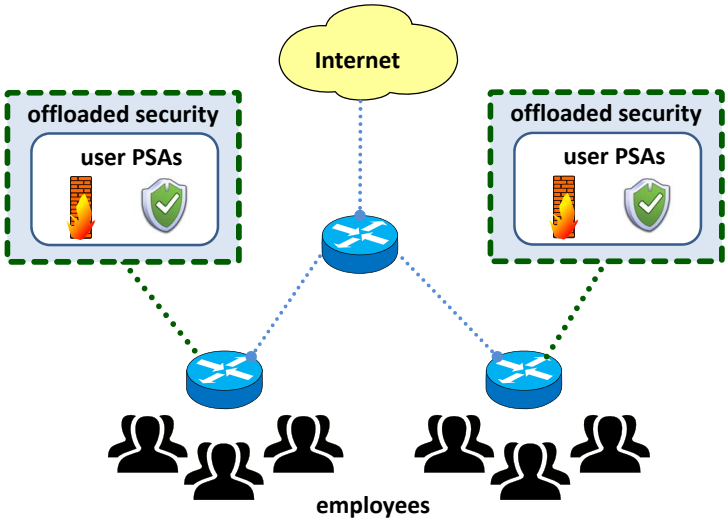

Figure 5: Monolithic NEDs placed in the enterprise.

1.2.5 Internet of Things (IoT)

In addition to the primary intended use-cases listed above, SECURED may also consider an extended use-case for special purpose internet connected devices, for instance sensors and embedded consumer products. These devices typically access the internet over a wireless network (Wi-Fi, 3G, ZigBee, ...), but are not very configurable, and have extremely limited compute capabilities and features. These are usually managed by a local administrator with some degree of technical knowledge. Furthermore, it may be impossible to install anything locally on these devices (e.g. for adding VPN or additional protocol support), so this case must not rely on anything local to the user terminal.

1.3 Application- and policy-driven configuration

When it comes to policy configuration, two approaches are considered for the architectural design:

Application-driven scenario. The user selects applications and either defines a high-level policy which can be applied everywhere or the user can manually create medium-level policies for each personal security application.

Policy-driven scenario. The user creates a high-level policy and applications are automatically assigned to fit the scenario. Since this scenario is fairly advanced, it is not covered within the alpha version of this specification.

2 Threat model

This section describes the primary threats that may arise when security is offloaded from the user device and into a network edge. Threats are evaluated from three distinct perspectives: legal, privacy, and traffic. The legal analysis highlights possible legal issues regarding the application of such a model, the privacy analysis highlights possible threats for the user privacy, and the technical analysis highlights possible security related threats. The list of identified threats have been used to specify the requirements presented in Section 3.

2.1 Legal and Privacy

The legal and privacy analysis at this stage of the project takes into consideration some general issues related to the legal matters rising from application of the model and the possible threats for the user's privacy. A proper threat model analysis from the legal and privacy perspective will be properly conducted considering the different use cases defined in Section 1.2, once the security review is carried out. Moreover, adequate requirements, in parallel with the technical and security requirements mentioned in the following Section 3, will be developed.

2.2 Technical

The technical analysis focuses on threats regarding traffic generated by the user and traffic directed to him. From the technical point of view this threats can be divided into three categories: traffic processing, traffic interception/manipulation and traffic generation.

2.2.1 Traffic processing

The user has no direct access to the NED and can not verify that their traffic is processed according to their specification. A malicious NED, which does not apply any security, may pose as a legitimate one and therefore the requested security features may not be applied. SECURED users need a method to verify that their traffic is processed correctly.

SECURED allows third parties to write security specifications for users. This could be a threat in case the other actors are malicious. Therefore the users must be notified when somebody overwrites their security requests.

2.2.2 Traffic interception and manipulation

Traffic interception and manipulation is the most serious threat in the case the user wants to offload the creation of secure channels to the NED.

During the transmission from the user terminal to the NED the traffic could be intercepted and manipulated by a malicious user. Therefore the connection from the user terminal to the NED must be sufficiently secure. Depending on the connection type the security requirements may be different.

Since each NED processes the traffic of multiple users the threat of traffic interception and manipulation exists also inside the NED itself. Therefore within the NED the traffic of different users must be isolated.

2.2.3 Traffic generation

A malicious applications on the user terminal may establish an alternative connection to the Internet or an other network in general. This alternative connection can be used to send and receive traffic which has not been processed by SECURED.

An alternative scenario may be that an attacker bypasses the NED and contacts the user terminal. Also in this case the traffic is not processed by the NED and the requested security may not be applied. Another problematic scenario would be an external entity generating traffic to overload the the user terminal, such as a distributed denial-of-service (DDoS), which would affect the NED's service.

Therefore the SECURED architecture must ensure that all traffic from a remote device to the user terminal is processed by the NED, and that the NED has the capacity to manage the traffic.

3 Requirements

This section defines the security and technical requirements for the SECURED architecture, which addresses many points raised from the requirements analysis in D2.1 [2]. Some of the requirements can be weakened if other conditions hold. The architecture will have some components or features which can be made optional given certain assumptions about security and trust of some components and environments.

3.1 Security requirements

Completeness. All the traffic generated by, or headed to, the user terminal (including the LAN traffic) must be processed by the NED.

Trust. Since applications would be executed within a node that is not under the control of the end user, a verification mechanism is needed to provide evidence that the NED can be trusted to run the applications. In particular, a NED should provide the following guarantees.

First, it must prove that it is an original device and not a node simulating the SECURED behaviour (for example by reproducing the same output upon a request); the consequence of trusting a fake device could be that its owner or an intruder could manipulate the traffic of the victim at their will. The device can be evaluated using a technique like remote attestation.

Second, a NED must prove that the traffic of a given user is processed by the applications he requested and not by some malicious software (that could, for example, forward all user's traffic to an attacker's favourite location).

Note that trust should come from the evaluation of these guarantees, but SECURED does not exclude the possibility to accept other sources of trust. For instance, end users might be satisfied with trust originating from non-technical considerations, such as having the physical control of their home gateways or a contractual agreement (and corresponding liability) with their ISP.

Channel protection. If the user trusts a NED, they must also create a protected channel with it, so that attackers cannot manipulate the traffic between them. In addition, they must ensure that the other endpoint of the protected channel is the same entity that presented the trust proofs, otherwise an attacker could perform a man-in-the-middle attack by relaying the proof requests and replies to a trusted device.

Isolation. As a NED could be multi-tenant (e.g. many users connected at the same time to a public WiFi access point), it must ensure the proper separation of traffic of the different users and must bind each flow only to the applications that originated it. Since applications could misbehave (e.g. due to a bug or a vulnerability exploited through a malformed packet), a NED must properly confine each application so that a misbehaving one does not affect the others.

Authentication, Authorisation and Accounting. The architecture will include many different external services, some of which may be managed by different stakeholders or are held in different domains. Each repository and external service will require authentication. Hence, a common authentication system between systems would be an ideal requirement. However, as an example, if policies and application binaries are held in different domains with different authentication credentials, then there must be a way of transparently supporting different credentials in an interoperable manner.

3.2 Technical requirements

User authentication. To deliver protection to the right user, a NED must have the capability to recognize who is currently connecting to it with a standard authentication procedure (e.g. a username/password pair). It is worth noting that this is not a mechanism for network access control[1], although the NED could use information exchanged during this phase. Rather authentication is needed to retrieve the user's profile (applications and policies) so that a NED knows how the traffic of this user must be processed.

[1] For connecting to the network, the user might have already performed and authentication step (e.g. as part of a 802.1X access control procedure).

Standardised platform. Since a security application could run on an arbitrary NED (e.g. home gateway or corporate switch, depending on the location a user connects from), it must be designed to support different environments. This requirement could be met by designing applications in a platform-independent way (e.g. as Java byte-code) or ensuring that a NED could run the environment required by an application (e.g. through virtualization).

Standardised policies. Typically applications that accomplish similar tasks for different platforms offer different configuration options, thus increasing complexity for a user to obtain the same behaviour. To overcome this problem, a user should have the possibility to express how his traffic must be processed with an application-independent policy language. The user policy should be used transparently by different SECURED providers (may be limited by semantic problems or specific policy extension of a provider). The PSA(s) should run everywhere but may be limited by business, trust, or technical impediments of a specific SECURED provider.

Interoperability. Interoperability must be considered for the three major stakeholders: users (have PSA running everywhere, have policy accepted everywhere), developers (create a PSA that can run everywhere, support policy with a well defined specification), and providers (support policy and PSA with a well defined specification).

Scalability. Since the NED is primarily a networking device (although augmented with computational capabilities) supporting a massive number of concurrent tenants connected to it, all the NED components executing user applications should be as lightweight as possible, with fast primitive operations oriented to network processing, such as packet filtering and segment/payload reassembling.

Availability. The architecture must be able to support the availability needs of enterprises, service providers and network operators. These stakeholder typically have large distributed infrastructures, where providing high availability helps to avoid service faults and downtime. This is also related to the *Scalability* requirement.

Deployment scenario. The SECURED architecture must support different deployment scenarios. We should at least support the four deployment scenarios mentioned in the "use cases" section. Additionally, the SECURED architecture should support cloud-like environments, so that the user traffic is processed by using Network Function Virtualization technologies.

4 Architecture overview

The high level architecture contains the functional specifications of all the SECURED components and their exposed interfaces. The main workflows and connection setup for SECURED are described in Section 5.1.

4.1 SECURED application

The SECURED application runs on all user terminals and takes security decisions on behalf of the user. It is very lightweight and has minimal impact on the user terminal, designed to provide support for protocols such as NED discovery, remote attestation, network configuration (such as VPN for accessing remote NEDs over untrusted networks) and for mobility handover. In addition to these capabilities there

will be a way to receive notifications from the user PSAs deployed on the NED, which will initially be developed as part of the SECURED application. The application can be activated and deactivated by the user. When the application is activated all the network traffic is processed by the connected NED, otherwise the user terminal acts and reacts like a device without SECURED.

Whilst the SECURED application is strongly recommended for better user experience and security, it is considered an optional component in the architecture due to use case scenarios where applications cannot be installed on the local user terminal, such as with the IoT use case.

4.1.1 NED discovery

In the absence of a trusted local NED, the NED discovery module is responsible for identifying suitable remote NEDs. The discovery process includes NEDs specified by the network configuration of the user terminal, a user-defined NED list and public NED lists. After the module has identified possible NEDs, the best NED is selected and the connection is established. The most suitable NED will be a trade–off between latency and NED capabilities. For instance, a small and localised datacenter servicing a residential area may result in better throughput for an Intrusion Detection System (IDS) than a local gateway. If the NED is not the default gateway, the nearest NED can be presented to the client either through Dynamic Host Configuration Protocol (DHCP) information or through a Zeroconf implementation.

4.1.2 Remote verifier

The Remote Attestation (RA) verifier is an optional local or remote service which assesses the NED measurements during the trust establishment phase. The integrity analysis of the report returned by the NED can be performed either locally on the user terminal or through a remote trusted third party verifier. The RA protocol includes the verification that the NED is genuine and the integrity analysis of the NED measurements. Furthermore the integrity analysis can be customized by the user by adding or removing integrity requirements. Since computation is limited on the local device, only a small digest list of trusted software should be checked. This can be particularly appropriate for fixed connection scenarios. For more diverse NED software stacks, a remote verifier would be more computationally equipped to handle a diverse range of measurements. This in turn would be more appropriate where users connect to many different NEDs, such as in the mobile scenario. The location of the remote verifier can be anywhere: either on the local network or as a completely independent third party on the internet (e.g. an example domain would be `https://verifier.secured-fp7.eu`).

4.1.3 Network configuration

After the NED has been attested, the network configuration module establishes a secure and trusted channel with the NED. Furthermore the network configuration module ensures that all traffic generated by the user terminal is processed by the NED and no other communication channel outside the secure channel is established.

4.2 User Profile Repository (UPR)

The UPR is an independent network service which stores the user profiles (including the policies) of SECURED users. It is dependent on a general purpose authentication system, which is described

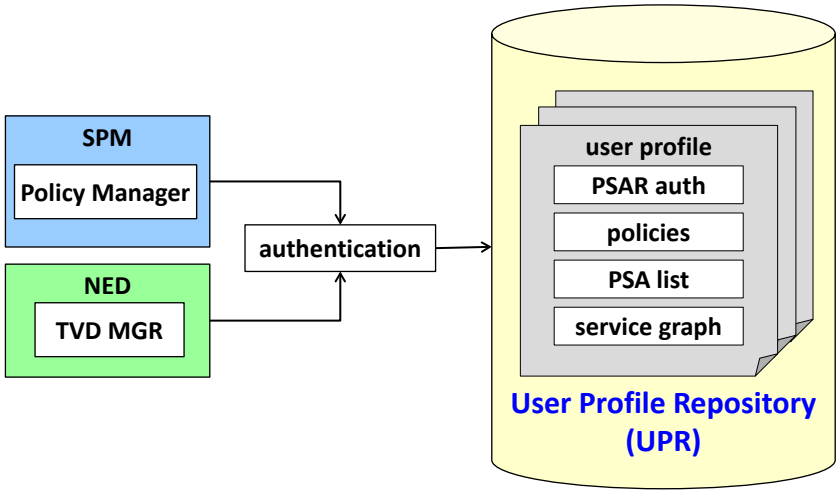

Figure 6: Overview of the User Profile Repository (UPR).

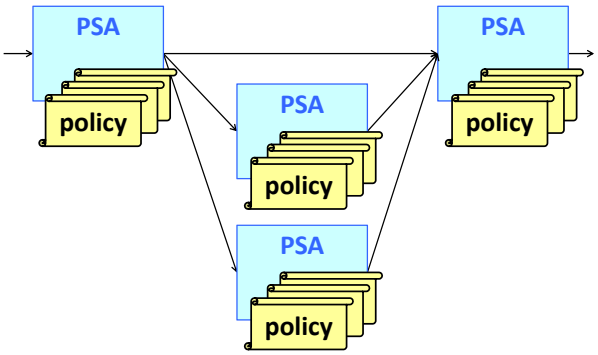

Figure 7: Service graph.

in Section 4.9. Apart from the user policies, it also contains the latest service graph, which is the composition and configuration of PSAs inside the user's TVD. An abstract overview of the UPR is shown in Figure 6.

User profile
A user profile contains a list of the authentication credentials, a list of all the PSAs in possession of the user, the security policies, the service graph, and the address of the PSAR location.

Service graph
The Service Graph is the formalism that describes the service requested by the user and it is, in its simplest form, a set of cascading PSA that are active on the user's traffic. The service graph can be either specified by the user in case of the application-driven service model (e.g. an experienced user that selects the PSA that have to operate on their traffic and their service order) or calculated automatically (e.g. in case of a standard user). The latest service graph of the user is always stored in the UPR, for which the graph can cover one of two cases: *policy-driven* (where PSAs are selected based on an abstract optimisation-profile) and *application-driven* (where the user has selected particular preference and ordering of PSAs).

Formally, the service graph is defined as the superposition of two directed graphs in which nodes represent the PSAs and the arcs represent the flow of the traffic between the two nodes. Arcs can be labeled with a set of packet filtering rules (e.g. OpenFlow Flowmods) that select which traffic has to flow through that arc in that direction. The initial/final arc of the service graph terminates on the Service Access Points, which at the first sight can be considered (i) the user terminal and (ii) the Internet. In fact, service graphs are composable in order to allow the traffic to traverse the service graphs of different actors (e.g. the end user, the service providers, etc); the actual service experienced by the user results from the concatenation of all the service graphs of the involved actors. The composition of the service graphs is carried out by merging the IN/OUT service access point, according to the order given by the priorities of the different actors. The service graph is intended to describe the service, not how the service will be provided. As a consequence, no physical information is present such as physical/virtual ports, application/VNF (Virtual Network Function) placement, topology, and other. The complete service graph will consist of the user service graph concatenated with other service graphs which are forced by the administrative domain. For instance, a user service graph will subsequently enter the employer's service graph (which has it's own set of PSAs and policies). An example of a possible service graph is shown in Figure 7 and its textual representation can be seen in Appendix Appendix B – Abstract service graph. The service graph is represented by two lists: one of the PSAs and another one with the flow-rules that the traffic must follow (the most selective rule has the priority in case of multiple matching).

4.3 Security Policy Manager (SPM)

The SPM is a remote service for users to create, delete, edit, view, and export their security policies. End users define a single HSPL policy or multiple MSPL policies. For more information on policies, see Section 6. Once the policy is defined, the policies and service graph can be exported to the User Profile Repository, defined in Section 4.2. The interaction between the SPM and the User Profile Repository is illustrated in Figure 8. This policy support tool offers two different interfaces:

- a web-based management system for end-users to define their policies, create profiles for quick enforcement (and sharing), find appropriate PSAs to implement the policy, or match a policy with the available PSAs;
- a programmatic interface (likely based on the web-service paradigm, but the actual model will be decided during the project implementation) offering various policy management services to the other software components of the project, such as policy retrieval for a specific user or PSA, translation from HSPL to MSPL, conflict detection and resolution among different policies (for example when the user policy must be combined with the employer policy) and identification of security capabilities needed to implement a policy.

In addition to providing a policy editor and translation services, the SPM will contain additional logic to handle the policy-driven scenario, which includes automatic selection of appropriate PSAs given a set of constraints or criteria. The definition of the SPM services will be provided in Deliverable D4.4 [9]. Information about the policy refinement service is described in Section 6.1.

Given that SECURED is being designed with mobility in mind, it is important to reduce the time it takes to establish a SECURED session wherever possible. For instance, it may not be necessary to perform policy translation and configuration generation on each new SECURED connection. This can be accomplished through cacheing of policies and configurations for fast instantation. However, it is

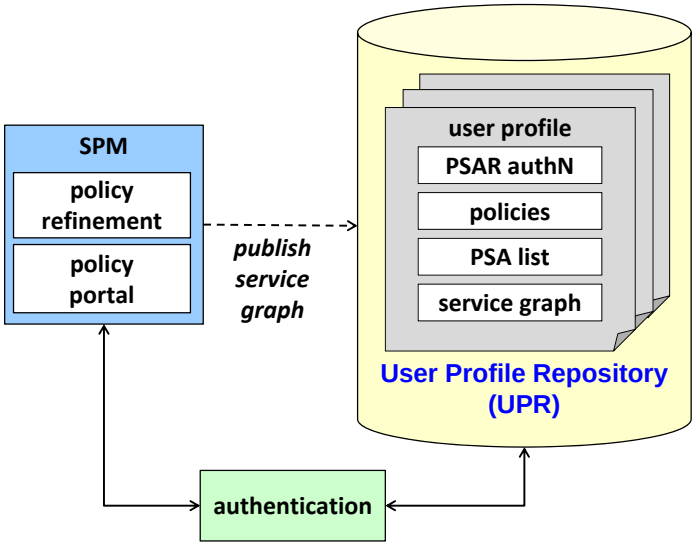

Figure 8: SPM creates service graph and publishes to the user profile repository.

extremely vital that when policies do change that the policy translation services are invoked. Hence, the architecture introduces a logical component to take care of two workflows that occur in the infrastructure: one which occurs during the policy definition phase and one when the affected user connects to a NED. These are respectively known as *offline* and *online* workflows, and must involve the SPM for the specialised services as well as the UPR for storing the user's own policies and configurations. An overview of the relationships of the workflow manager concept is illustrated in Figure 10. The next two subsections describe the architectural details of the workflow manager.

4.3.1 Offline WorkFlow Manager (OffWFM)

The offline workflow manager must take care of the phase where the user is specifying policies and must store the results of the policy translation service in the User Profile Repository (UPR). The workflow responsibilities include:

- validating that the high level policies are enforceable (via an SPM service);
- storing the high level policies in the user's profile (on the UPR);
- invoking translation of the high level policies into medium level representation (using the H2M SPM service), and storing the result in the UPR.

If the high-level policies are not enforceable then the user must be notified during the creation. Once the workflow has successfully completed, medium-level policies will be stored in the UPR.

4.3.2 Online WorkFlow Manager (OnWFM)

The online workflow manager must take care of validating whether changes or conflicts have occurred in the policy stack and must also take into consideration relevant situational data from the infrastructure

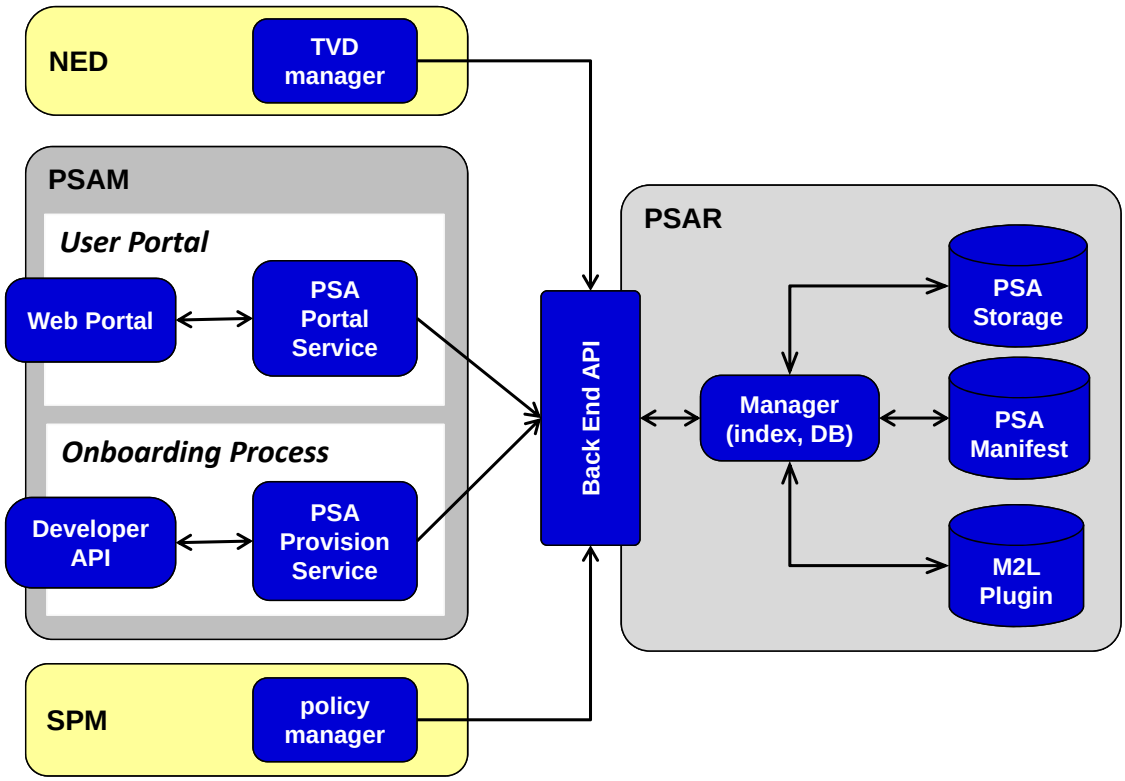

Figure 9: PSAM front-end and PSAR back-end.

layers. More information about the policy stack is given in Section 6. As an example, such situational awareness may include factors such as temporal or geographical data.

When the user authenticates with the NED, the online workflow manager (inside the PSCM) must also simultaneously send a request to see if policies have changed. If there are no changes in the policies of the policy stack members and if there are no unresolved conflicts, the instantiation procedures work as normal. However, if policies have changed, then the workflow manager must invoke the M2L service of the SPM to refine and translate the latest medium level policies to low level configurations. It is currently planned that identified conflicts will cause a report to be generated in the user's profile at the UPR, where they can resolve it through a UI of their choosing.

More details of the workflow manager's interactions, including the H2M and M2L refinement processes, can be found in [10].

4.4 PSA Manager (PSAM)

The PSAM contains two modules: the user portal and the developer service. Each module is composed of an interface (i.e. a web page for the users and an API for the developers) and an underlying service that performs the checking logic and interacts with the others SECURED infrastructure components (PSAR, SPM, NED, etc.).

The user portal allows a user to configure their user profile (i.e. service graph, PSAs and policies) while the developer service is used by a developer to make their PSA available to the user.

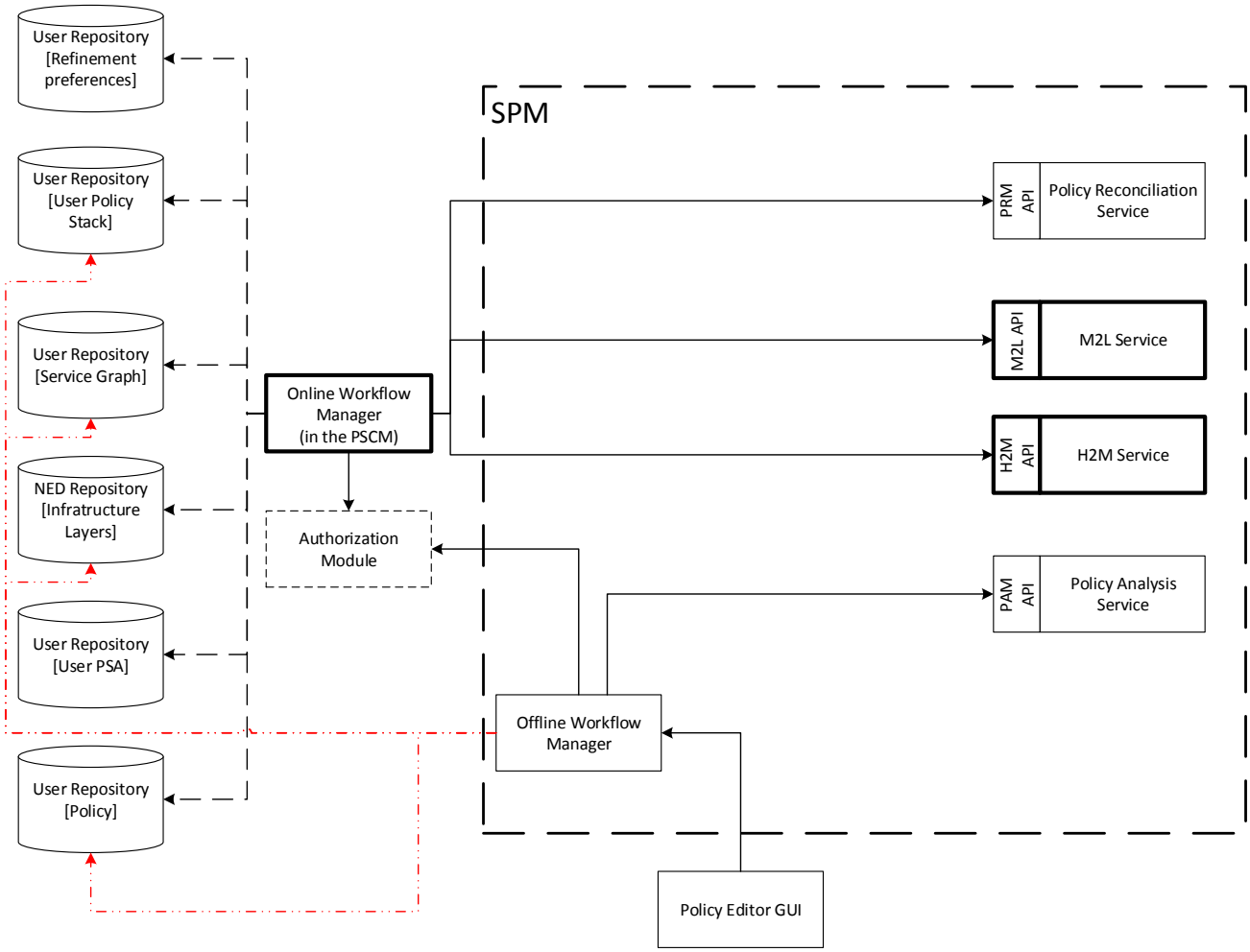

Figure 10: Online and offline workflow managers in the SECURED architecture

4.5 PSA Repository (PSAR)

The PSAR is a storage service, addressable through an API, for the PSAs and their associated manifest and policy translation plugin. The PSAR API can be accessed by the PSAM (for a developer to put their PSA or a user to choose a PSA), the SPM and the NED (to retrieve the PSA or the plugin). The relationship between the PSAM and the PSAR, and the NED is shown in Figure 9.

4.6 PSAM–PSAR integration

SECURED envisions two main ways of deploying the PSAM–PSAR (with regards to the NED): the open and closed ecosystems. In the open ecosystem the PSAM and PSAR are publicly accessible by any NED, while in the closed ecosystem there is a on a set of NEDs that can use the PSAM and PSAR. For the open ecosystem, the user should be able to authorise access to the PSAM, PSAR and UPR using a third party service (e.g. a decentralised protocol and identity provider such as OpenID).

The open ecosystem aims to allow the creation of FOSS PSAs repositories or PSA marketplaces where

a user can buy some PSAs to be executed on every NED they will connect to. The closed model is more suitable for the enterprise scenario where the PSAM and PSAR are part of the enterprise network, and the PSAs can be audited before their deployment inside the company.

4.7 Connection to the NED

Depending on the scenario, the connection to the NED can be established through different access mediums (i.e. layer 2 level links) that may be shared or open to public. Nevertheless, whatever the medium type, the user device–NED channel must be secure; in this context security can be separated in three properties: confidentiality, integrity and authentication.

Table 1 summarises some widespread features that enforces the required properties for the primary SECURED use cases with the three most relevant connection media: Ethernet, WiFi and 3G/4G. This list is proposed to show examples of the enforcement of the channel's security, any user device–NED channel proposing the required properties can be used to connect to the NED; in particular, any tunnel over an insecure medium that provides the three security properties can be used.

use case	medium	secure channel enforcement
Home	Ethernet	Implicit (physical–based) through the direct connection.
	WiFi	Confidentiality and integrity through WPA2, authentication with SSID (weak)
	xDSL/FTTX	Confidentiality and authentication using Radius over PPPoE for ISP-hosted NEDs (Section 1.2.1)
Enterprise	Ethernet	Implicit confidentiality and integrity through direct connection, explicit authentication with IEEE 802.1X
	WiFi	Explicit secure channel through WPA2 EAP-PEAP
Public hotspot	WiFi	Potentially nonexistent, can be created with the SECURED app
Mobile ISP	3G/4G	Security of the channel built into the protocol (confidentiality, integrity, mutual authentication provided through 3GPP AKA)
Virtual/Remote NED	IPsec	Security of the channel built into the protocol.

Table 1: User device–NED secure channel enforcement.

As an optional feature to the connection establishment, an implementation can also tie the initial attestation (of the user–independent part of the NED) with the secure channel handshake. However, regardless of the implementation and as stressed in section 3.1, the identity used for the trust attestation of the NED must be strongly bound to the authentication of the secure channel[2]. More information on this is provided in [3]. Whilst the standard architectural models allow the user to authenticate via the NED, it is also recommended for particular scenarios, such as in the enterprise scenario, to provide *mutual authentication* between the user terminal and the NED. Some different implementations examples for the secure channel establishment are described in Deliverable D3.1.1 [4].

[2]Implicit authentication of the secure channel weakens this binding.

4.8 NED components

The following subsection provides definitions for all the key components required for all types of NED, including both virtual and monolithic. The NED architecture employs a hierarchical model with the principle of least privilege, where user PSAs are controlled by a personal controller (PSC), which in turn must be able to interact with the NED's global TVD Manager for privileged operations to occur. The subsection is then followed by a brief overview of the interactions between the components. More details of the components and how they can be implemented are described in deliverable D3.1.1 [4].

4.8.1 Personal Security Controller Manager (PSCM)

The PSCM is the NED font-end, it is contacted by users to setup a connection with SECURED and contains two modules: the Remote Attestation Agent and Authentication Module. The *Remote Attestation Agent* is in charge of executing the remote attestation protocol with the user and reporting the integrity status of a NED. The *Authentication Module* requests to a connecting user a proof of their identity to retrieve the user profile (policies and applications).

4.8.2 Trusted Virtual Domain (TVD)

SECURED defines a Trusted Virtual Domain (TVD) as a logical container (or sandbox) for the PSC and PSAs of one user. This implies that each user has one personal TVD associated to him. The main properties of TVD are: isolation, trust and simplification. The isolation property ensures that a PSA can communicate only with other PSAs as well as with the PSC of the same user TVD. The trust property ensures that a PSA can only execute inside a TVD if it has an appropriate level of trust. The simplification property allows to uniformly apply security policies to all PSAs of the same TVD.

Trusted Execution Environments (TEE) The isolation at the application level in SECURED imposes the requirement to introduce the concept of Trusted Execution Environments (TEE). The TEE is defined as the virtual execution environment for executing one or more PSAs. It includes two parts: the privileged area and the unprivileged area. It is administered and configured by a user's Personal Security Controller (PSC) through a *Control and Management* interface (see Section 4.8.4).

The privileged part contains the control and management modules of the TEE which enforce the communication with its PSC. Furthermore, it controls the execution of the different PSAs acting as a proxy between them and the PSC. On the other hand, the unprivileged part contains one or more PSAs. If there are multiple PSAs in an TEE, then the TEE must isolate PSAs from interfering with the execution of other applications or the network configuration.

4.8.3 TVD Manager (TVD MGR)

The TVD manager is the component that determines, creates, controls and destroys the TVDs of connected users. This component is split in two subcomponents: the NED Orchestrator and the NED Management. The former is a technology-independent module, and the latter is the technology-dependent module. The relationship between the PSCM and the TVD Manager is shown in Figure 11.

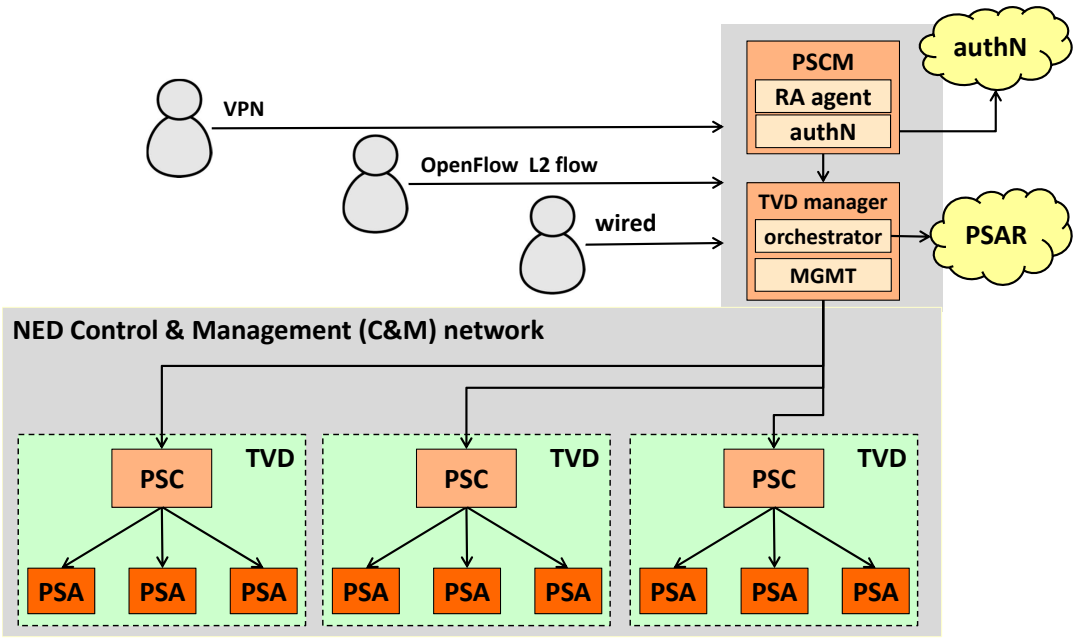

Figure 11: Initialisation: NED authentication and user TVD instantiation.

NED Orchestrator

The orchestrator is the decision–maker component that allocates resources to a user's TVD and creates the connections of their corresponding TEEs to the data and control/management networks. An important remark: the orchestrator stores only the user session tokens and does keep any user context information. This module is aimed to coordinate and manage the NED TEEs instantiated and the internal network topology.

Whilst this component is technology-independent, it relies upon the NED Management module to enforce its decisions and uses the control plane network to retrieve the needed user informations from the SPM using the user session token. The user information useful to the orchestrator is stored inside the user profile, but the orchestrator will also need to locally load the user's PSC instance, fetch the PSAs and policies, but must not perform any action on them (apart from transferring them to the appropriate component).

NED Management

The management component is the technology-dependent part of the platform that is privileged enough to enforce the decisions of the Orchestrator. The two main functionalities are: management of TEEs and configuration of the virtual networks within a user TVD.

4.8.4 Personal Security Controller (PSC)

The Personal Security Controller (PSC) is the component which controls and manages the TVD environment, TEEs, user applications, configurations and network interfaces on behalf of a single user. Each user gets their own instance of a PSC inside their TVD, which captures all the user state and context for their session. Since the PSC is privileged to load applications and enforce configurations, it is by design kept isolated from the applications, and all applications related to the user must communicate

to their PSC through a restricted Control & Management interface. However, changes to the overall TVD (such as creating a new TEE) require sending a request to the TVD Manager, which is an external privileged entity outside the user TVD. The role of the PSC and the datapath for the user traffic flows are depicted in Figure 12.

The main roles of the PSC component are described next:

- it keeps the abstract service graph (PSAs and interconnections) of the user;
- it determines the TVD topology (realization of the service graph) from PSAs requirements (it may change depending on NED resources);
- it sends the TVD topology to the TVDM orchestrator (the latter contains all TVD topologies and information on how TVDs are chained);
- it monitors the status of the TVD (e.g. detecting if a TEE has crashed and requires restarting, or the allocation of a new one);
- it receives the manifests of all the PSAs and assigns each PSA to a TEE.

The PSC is internally divided into two main parts: *Control* and *Management*, each with two different interfaces respectively.

PSC Management

The PSC management tasks include:

- controlling the execution state of the PSAs (e.g. start, stop, restart);
- it pushes the initial configuration of the PSAs;
- it receives the low level configurations for the PSAs from the TVD Manager;
- it retrieves and enforces the PSAs chaining topology.

PSC Control

The PSC Control tasks include:

- monitoring information relay for listing the execution state of the PSAs;
- static measurement of the PSAs;
- PSAs exception handling which can include execution exception (crash, failures) but also flow-related information (malware detected, intrusion detected, illegal access).

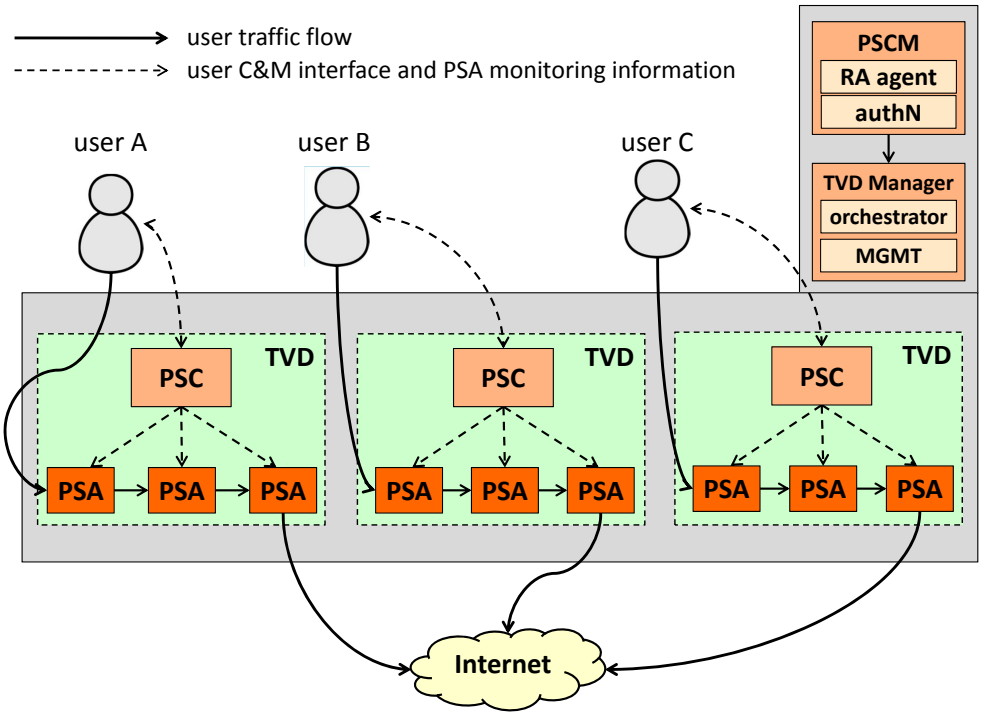

Figure 12: Established datapath between the users, their PSAs, and the Internet.

4.8.5 Personal Security Application (PSA)

A PSA is a component that implements the security policies selected by the user and actually processes the user's network traffic. The PSA is composed of a set of security controls that implements one specific security capability (defined in Deliverable D5.1 [5]), e.g. network monitoring or access control. The user can select the PSAs by hand and configure them separately (application-driven scenario) or define a set of policies and let the SECURED platform select and configure the PSAs according to the defined policy (policy-driven scenario).

A typical PSA contains only a single security control and is called a simple PSA. A PSA that contains more than one security control is called a complex PSA. The benefit of a simple PSA is that its functionality is well-defined and it can be easily used to chain multiple PSAs together to enforce specific parts of a policy. On the other hand, the performance of a complex PSA can be better when enforcing a policy that requires using multiple security controls together, since there is no overhead of passing messages between different PSA execution environments.

The primary PSA execution method supported by SECURED will be the traditional Virtual Machine PSA model. SECURED will support a standard TEE for the VM, in which PSAs are configured and executed. By default, each VM contains a single PSA and service chaining will be used to link PSA traffic inside the user TVD such that it complies to the user's service graph definition.

Other PSA execution environments being considered for the architecture of the SECURED platform are the Container PSA model and Application PSA model.

In the Virtual Machine PSA model the PSA runs within an isolated execution environment that is facilitated by a virtualised OS instance, e.g. Debian or Windows. The benefit of this model is that SECURED can support various security controls that run in different operating systems, allowing developers to use

existing development tool-chains and also to providesome support for network applications on legacy operating systems.

The primary disadvantage of this model is that the PSAs contain a larger footprint than that of the latter models and therefore starting/migrating PSAs will be slower. The Container PSA model enables multiple isolated user space instances using a single kernel thus minimizing the kernel space duplication between PSAs, but at the same time reducing the isolation among PSAs. In the Application PSA model the PSA is running inside a special execution environment, e.g. a Java application (running in JVM) which is inside the specified SECURED TEE. The PSA computation models are described in more detail in D5.1 [5].

In order to correctly enforce policies for PSAs, the PSA is accompanied with a plugin that can translate a user's policy into its own native configuration. This is known as the "M2L translation plugin", which is able to perform the translation of MSPL policies for the given low-level PSA configuration. The MSPL policy translation service reads the user PSAs and coordinates the correct M2L invocation. More details on this service are presented in D5.2. For details on MSPL policies, see Section 6.

A "PSA manifest" is associated with each PSA to describe,e.g. the PSA in general, describe the security functionalities (capabilities) the PSA provides and the requirements the PSA has in terms of the execution environment used to run the PSA. Additionally, the manifest provides information, for instance, about the performance of the PSA (network throughput, etc.) to help the user in choosing the PSAs (in application-driven scenario).

4.9 Authentication System

The authentication system is an independent system which allows users to be authenticated with their services: the UPR, the PSAR, the PSAM and the SPM. It must be able to receive at least a username and password and be able to return a unique cryptographic session token with a short lifespan, which is enough to last for the setup phase of the user TVD on the NED. The cryptographic token must be used at least to access information from the UPR, such as the service graph and the polices. In an open ecosystem with an independent identity provider, the user must pre-authorise access and verify their identity to the all the services prior to using a NED.

By default, the architecture has been designed with two tiers of authentication in mind. One set of credentials for their authentication system identity, called the SECURED credentials, and the second set of credentials in the form of long-lived authorisation "cookies" or credentials used to access other third party services. In this scenario, the user can use a single unified credential to gain access to their pre-authorised services. This allows for an open model where the user profile is stored and managed in an independent domain from the PSA provider. This is particularly suitable for an open repository model, where sensitive and private user data, such as policies and identity information, can be kept isolated from the providers of the PSAs.

In a scenario where the SECURED provider is also the network provider, or if in the enterprise scenario where all the infrastructure is owned, then the second authentication tier of cookies are optional and not required. For instance, a network provider and the PSA provider may be offered by the same company, in which casethe SECURED provider can be managed under one management domain, using a unified credential for all systems (e.g. mobile network access will automatically be authenticated to access the PSAR).

Figure 13 illustrates the role of the authentication system between the NED, the user profile repository and the PSAR. Existing authentication systems can be used to suit the deployments of the use cases. For

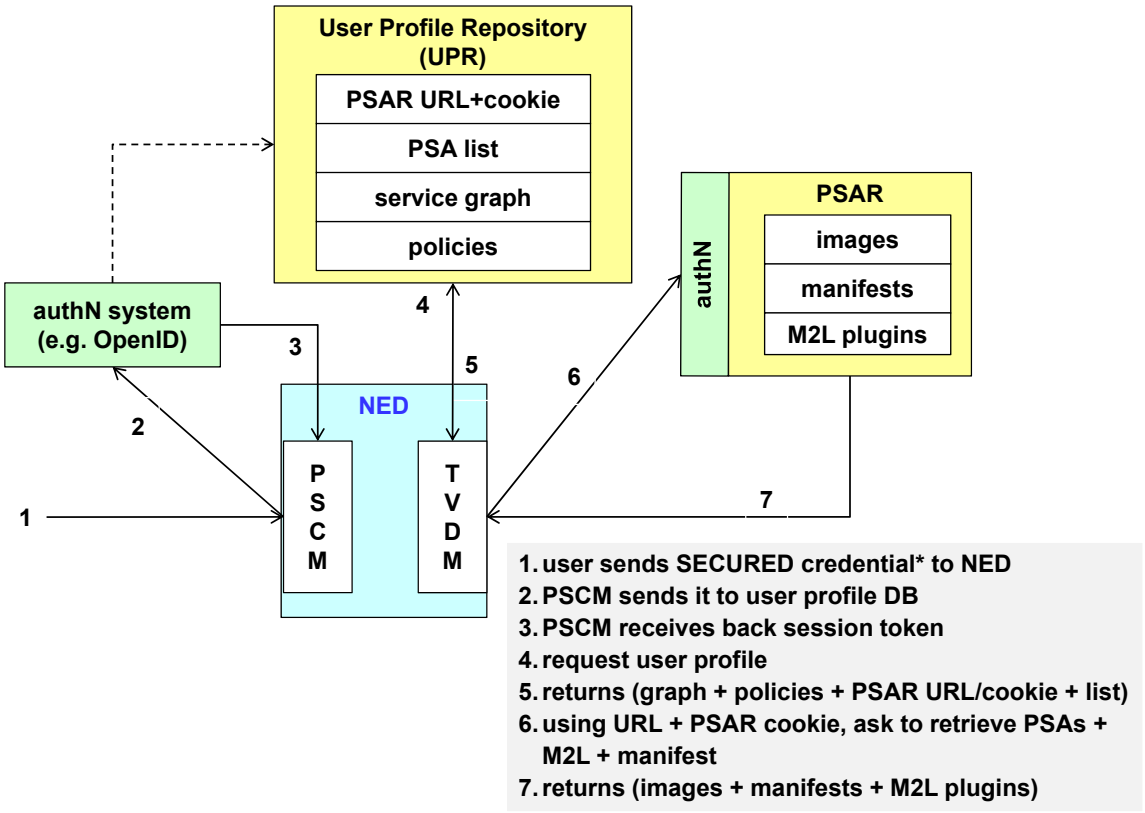

Figure 13: Integration between authentication system, user profile and PSA repository.

the enterprise scenario, a Kerberos-based authentication system with LDAP can be used. In a mobile scenario, the OSS/BSS of the mobile network provider will be able to leverage functionality in existing AAA+ authentication systems.

4.10 Interfaces

The following is a list of functional definitions of the interfaces required between the main architectural components. It is anticipated that most software components of the SECURED architecture will communicate by means of RESTful interfaces to allow for an evolutionary API.

SECURED application : PSCM. is the interface used by the SECURED application to create a new connection with a NED and to receive status updates from the NED during a running session. All the signalling for connection establishment and authentication credentials exchange occur on this interface. Also this interface can be used to receive the remote attestation measurements from the NED so that the user terminal can compare them to the golden measurements. Alternatively, through this interface, a compatible RA verifier can be selected. The SECURED application is able to monitor the user's TVD through this interface (security alerts, running status, . . .). In case there is no need for protecting the channel through VPN establishment (as in the case where a user is going to use a remote NED), then the NED must be the first point of contact (single-hop) for the user terminal to setup the SECURED service.

User terminal : NED. This is the interface used to send traffic to the NED. The data plane interface is established over a secure channel (VPN tunnel, WPA encrypted channel, ...) and all the ingress and egress traffic of the user's device flows through this interface.

PSCM : Authentication system. This is the interface used to authenticate a newly connected user. User's credentials are sent over this interface to authenticate the user and retrieve the authentication token. The PSCM will then pass this token internally to the TVD MGR so that it can proceed contacting all the other infrastructure components to retrieve the user's profile, PSAs, and policies.

Authentication system : UPR. This is the interface between the SECURED authentication system and the User Repository. It allows for the User Repository to verify the authentication token passed from the TVD MGR when it is asking for user profile retrieval.

PSCM : TVD MGR. This is the interface used by the PSCM to request the creation of a new TVD. Once the user is authenticated the PSCM passes to the TVD MGR the authentication token over this interface. The TVD MGR also notifies the PSCM about the completion of the user's TVD instantiation.

TVD MGR : UPR. This is the interface used by the TVD MGR to request the user's profile on behalf of the user's PSC. The profile includes information about the user's policy, and complete service graph. Through this interface the MSPL or the low-level configurations for the PSAs are retrieved from the SPM.

TVD MGR : PSC. This is the interface used by the TVD MGR to control a PSC; the PSC is also able to reach the external SECURED infrastructure (PSAR, SPM) through this interface, invoking the TVD MGR. The messages exchanged on this interface accomplish the following tasks:

- the user profile is uploaded to the PSC;
- the PSAs are fetch;
- the PSC can request the TVD MGR for the TVD expansion (PSAs instantiation);
- the PSC retrieves the PSAs' low level configurations;
- the TVD MGR can monitor the PSC status; this can be accomplished in two ways: polling approach (the Orchestrator pings the user's PSC to verify if it is alive and running) and heartbeat approach (PSC sends heartbeat to the Orchestrator to confirm it is alive);
- the PSC can request any TEE management operation (e.g. reboot, shutdown, or migration).

TVD MGR : PSA Repository. This is the service and interface that allows the TVD MGR to retrieve PSAs on behalf of the user's PSC. If the UPR and the PSAR are under two different administrative domains then this interface points towards an optional PSAR authentication system (the PSAR authentication credentials are retrieved from the user profile). PSAs tarballs, manifests, and M2L plugins are retrieved over this interface.

PSC : PSA. This is the interface used by the PSC to monitor, configure, and manage the execution state of a PSA. Low-level configuration scripts are passed from the PSC to the PSA through this interface. A PSA can also proactively send notification about any security threat recognized to the PSC.

SPM : UPR. This is the interface between the SPM and the UPR. Through this interface the user's policies are stored/retrieved/updated. The policies are stored into the user profile in the Policy Repository premises.

PSAM : PSA Repository. This is the interface between the PSAM and the PSA repository. The PSAM is actually made up of two different components serving different users: the developers' portal and the users' portal. Both can retrieve and upload to/from the PSAR different kind of information. A few are listed here (see D5.2 for more details [6]):

- list of PSAs;
- PSA manifest;
- M2L plugin;
- request PSA download, upload, block, and remove.

SPM : PSAR. This is the interface between the SPM and the PSA repository. The SPM requires contact with the user's PSAR to retrieve the PSA manifests in order to identify PSAs that are suitable for a policy-driven service graph.

5 Workflows

The SECURED infrastructure has three primary stakeholders: the PSA developers, the SECURED providers and the users. The workflows focus on the public facing, whilst management workflows for providers are described in D5.2 [6].

5.1 Developer workflow

The main developer workflow consists of making their PSA available to the users. This workflow will be detailed in a later version of this document (to fully support an open–market model), but for the alpha version of the architecture the PSAR is assumed to be on the same managed infrastructure domain as the PSAM and the NEDs.

Given that restriction, the PSAM should export a developer interface that allows the following interactions:

- developer registration;
- developer authentication;
- PSA publication (with the corresponding manifest and M2L plugin);
- PSA update;
- PSA removal.

Once the PSAM has performed the required check (the PSA, manifest and M2L plugin are correctly formed, the developer has the right to update or remove the PSA, etc.), the PSAM stores the PSA, manifest and M2L plugin in the PSAR.

5.2 User workflows

There are two main workflows for the users: the off–line workflow that allows a user to configure their security controls (policies and eventually PSAs) and the on–line workflow that consists of assessing the trust in the NED before it enforces the user's security controls.

5.2.1 Security controls configuration

The SECURED components involved are: the user profile repository, the PSAR, the PSAM, the SPM and, optionally, an authentication system. The user interface is part of the PSAM and the common actions are:

- **user registration**: the PSAM creates an empty user profile and if required the user PSAM credentials [3];
- **user authentication**;
- **user profile update**: change the PSAM credential, or fill the profile with the some third–party credentials (to automatically authenticate the user to a PSAR) for example.

Next, the user needs to create their service graph: SECURED supports two ways for the user to configure their security control - the application–driven and the policy–driven approach.

Policy-driven configuration.
For the alpha revision of this specification, only the application-driven configuration is supported.

Application-driven configuration.
The users create and modify their service graph by building the chain of PSAs they want to be applied to their network traffic; for each of the PSA involved, they also specify the policy to apply using the PSA-independent language MPSL[4] described in section 6.

Once the user profile has been correctly completed, especially the service graph and required credentials to retrieve the PSAs, the SECURED infrastructure – mainly the NED – can enforce the user security controls to their traffic.

5.2.2 Security controls enforcement

The PSAs and corresponding policies are applied to the user traffic at the NED level[5], which requires the following steps before granting the enforcement to the user:

1. **Secure channel establishment and NED attestation**: the user starts any interaction with a NED by verifying that its state and identity is "as expected" (he may rely on a third party verifier for that);

[3] The SECURED infrastructure may use an existing authentication system such as OpenID or BSS for example.
[4] The developer must provide a M2L plugin with their PSA to create the PSA configuration from the MSPL.
[5] It is a prerequisite that a secure channel exists between the user device and the NED.

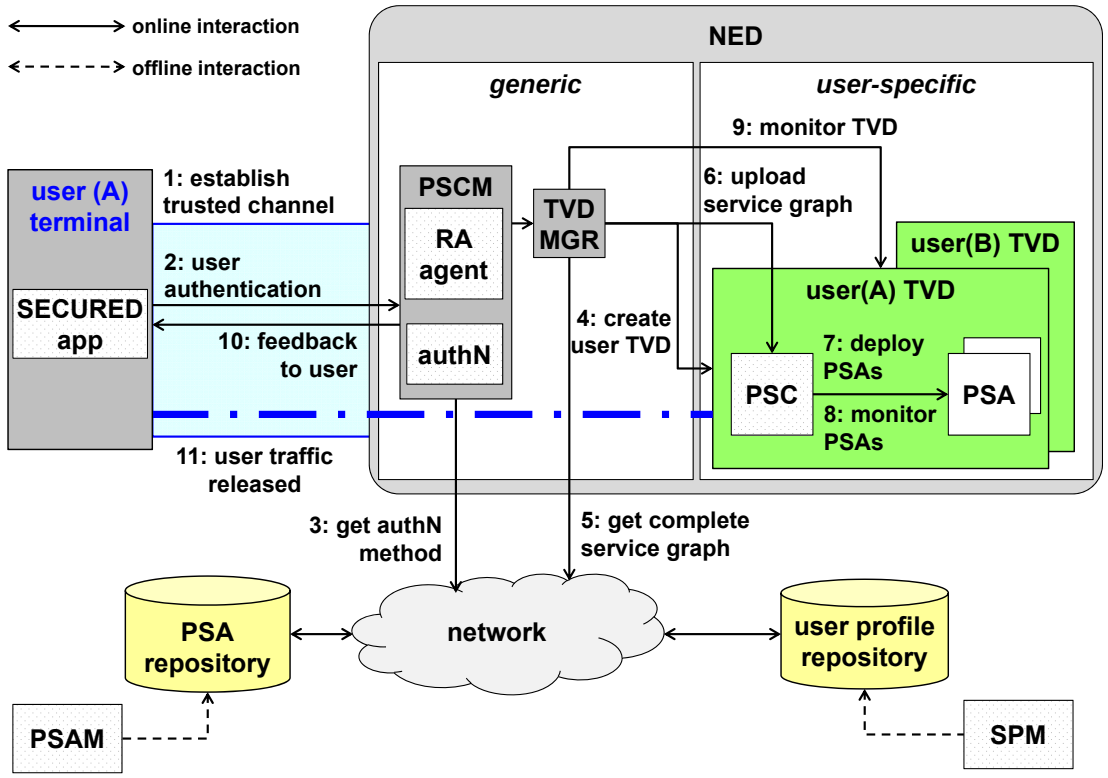

Figure 14: SECURED architecture overview

2. **User authentication and TVD creation**: once trusted, the NED can authenticate the user, retrieve their service graph, create their TVD and populates it with the user's PSC and PSAs[6];

3. **User Notification**: once the TVD has been instantiated and measured, a notification is sent back to the user about the status of the TVD (the PSAs launched) and the matching policy enforced.

Once these three steps have passed, the NED enforces the security controls on the user traffic.

4 **TVD monitoring** (optional): some NED implementations will also monitor the TVD environment and report changes to the user. These integrity reports are intended to be received by the *SECURED app*.

Each of these steps implies a trust establishment, either explicit or implicit, between the user and the NED: trust in the NED base components for 1, in the TVD for 2 or in the TVD execution for 3. Therefore, a SECURED infrastructure must focus on providing a comprehensive feedback to the user in case of failure. The interface used can be part of the SECURED application or a captive portal for application-less client for example.

An abstract overview of the NED component interactions is illustrated in Figure 14.

When a new HSPL needs to be enforced whilst there are existing connections to the NED, then one of the following basic actions should be taken:

[6]This step particularly requires a NED to retrieve the user PSAs (and their M2L plugin) from the PSAR; this is when SECURED compare the PSAs requirement (from the manifest) and the NED capabilities.

- all current sessions associated to the old HPSL are disconnected and the initialisation sequence begins again with the new HSPL;
- if the new HSPL does not alter the existing service graph (including the PSAs and the TVD topology), then new configurations will be sent to the PSAs;
- the final alternative is to not modify existing NED sessions and only apply the HSPL to newly created sessions. This would however mean that long lasting sessions will not be covered by the new policy.

6 Policies

The user policy stored in the user profile is divided in two sections according to the language used to represent it: the HSPL section and the MSPL section.

The HSPL is the language used to represent high-level directives about end-point protection. HSPL is currently under development and will be reported in D4.1[8]. Currently, HSPL allows the definition of security policies in the form of close-to-natural-language sentences that follow the syntax described below:

```
<subject> <action> <object> [<(field_type=values)>, ...,
                             <(field_type=values)>]
```

where `subject`, `action`, and `object` values are selected from a set of predefined values that will be provided by SECURED. Users (which must not be necessarily experts) can extend the base vocabulary and can associate `fields_types/value pairs`. HSPL can be used to specify the following network access control policies requirements:

- my son is allowed to access Facebook from 18:30 to 20:00
- all users are allowed to access all web content during lunch time
- network administrators are allowed to access SSH on servers in the server-subnet

where the semantics of the direct and indirect complements is defined by a set of attributes. For instance, 'Facebook' is a placeholder that links the official Facebook web site, and 'web content' and 'during lunch time' complements are defined as `(traffic type = web)` and `(time = 13:00-14:00)`.

Additionally, HSPL is used to enable specific security controls (that do not allow much configuration freedom). For instance, it is possible to specify the following HSPL policies:

- enable basic parental control
- scan email for malware detection
- remove tracking data from Facebook

Even in this case the semantics is implicitly defined by a set of predefined fields/values associations; for example `(check type = malware)` is used to select the detection type of the email scanner.

HSPL policies cannot be directly enforced by PSAs that are configured by means of low level settings. Therefore, SECURED has specified the MSPL that conveys the information needed to actually configure PSAs. MSPL is independent of vendors, products, opens source software, etc. that is, the information needed to configure security controls is structured according to a SECURED-defined data model. This data model builds on the PoSecCo abstract configuration language [13].

MSPL is structured in sub-models. The General Model describes the basic policy-related features, including rules, conditions, actions, etc. independently of the security control type. The access control, filtering, data protection sub-models, by subclassing and extending the general model, permit the instantiation of policies for specific categories of security controls. Other sub-models will be developed during the next years. MSPL will be reported in D4.1 [8].

6.1 HSPL policy refinement

HSPL policy refinement is one of the services performed by SPM. It takes as input the name of the link to the user repository and returns a refinement status notification. The refinement service accesses the user repository, reads the HSPL policy and refines it into the (semantically equivalent) MSPL policy for the PSAs the user has in their user profile.

The policy refinement is a sophisticated task whose feasibility depends on the user-specified HSPL and on the PSAs the user has in their user profile. Two ways are foreseen to perform the HSPL policy refinement: application-driven and policy-driven.

In the *application-driven approach*, the user selects the set of security applications (e.g. parental control application) and defines the HSPL policy. In this case, HSPL policies could be non–refinable. That is, it is not ensured that there exists a set of MSPL rules for the user PSAs that can enforce the specified HSPL. The refinement makes use of information from the PSA manifest that describes (among other things) the capabilities and other refinement-specific properties (see D5.1). A very simple example of non–refinable policy occurs when a user wants to filter blacklisted web sites but it does not selected any PSA with application layer filtering capabilities. Other more subtle forms of non–enforceability are possible, for instance, a user that wants to filter communications looking inside an application layer protocols but the PSA does not inspect the selected protocol, or a user that wants to filter communications based on the value of some fields that are not watched by the PSAs.

In the *policy-driven approach*, the user only specifies the HSPL, and then the PSAs are semi-automatically selected by the refinement service (from a catalogue of available applications) based on their capabilities matching those required by the policies. The selection can be straightforward (when only one PSA is available with a required capability) or may be based on various criteria (such as cost, performance, reliability or reputation) when multiple PSAs offer the required capability. For example, a user may prefer to adopt open-source applications (typically free of charge); choose applications with low network latency (e.g. to match QoS requirements); adopt applications that are reliable to faults or that have a better reputation (e.g. according to expert reviews). A trade-off amongst different criteria (e.g. cost vs. reputation) is interesting and useful to users. Therefore, in the case where the policy-driven approach is used, the refinement process also takes as input the type of optimisation the policy refinement process must perform and builds on optimisation techniques to select the best set of PSAs for the user.

It is expected that technology-savvy users and security experts would be more interested in the application-driven approach than normal users, while normal users should find it simpler to define their policy and then enforce it through the policy-driven approach.

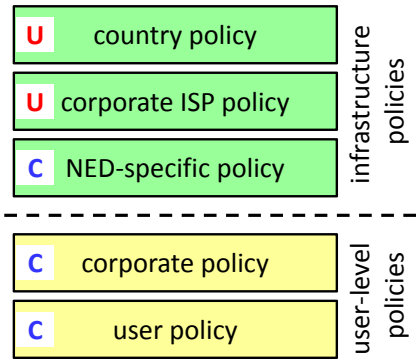

Figure 15: An example of a user policy stack

6.2 The policy stack

The security policy applied to a user is the combination of the security policies defined by different actors. For example, a user can specify his own security policy when he is at home. Similarly a company can specify security policies that will best serve its security needs. In this case, all the company's employees will be subject to these policies. Therefore, when the user connects from his personal smartphone or the business smartphone, he experience different security policies.

On top of these user/company-defined policies, there are other policies that depend on the network provider. For instance, the user's ISP may apply restrictions to the user traffic depending on the contract (e.g. low price contracts may not include VoIP traffic) while the company will certainly have no restrictions on the Internet connection. On the other hand, a company may define NED-specific policies (e.g. the NEDs accessible to guests may have a more restrictive policy than the ones only available to internal employees). Furthermore, country laws generate the need for other security policies (e.g. filter illegal gambling sites).

To model this scenario, SECURED uses the concept of a *policy stack*[7]. The policy stack is the ordered set of policies that apply on the user. At the lowest levels there are the user-level policies, that are specified by the user or other entities in the user's context (e.g. a layered company policy for employees, or the parents' policy for children). At the highest level, there are the infrastructure policies that depend on the entity that provides the network connection. As an example, Figure 15 presents an example of a policy stack from the corporate scenario explained before.

It is worth noting that, the user-level policies depend on the users and apply to them in every case, while the infrastructure policies are only known when the users connect to a NED

Policies in the policy stack are categorised in "cooperative policies", which are disclosed to the user by a *reconciler* which can access them for several activities (like inter-conflict analysis and reconciliation as described in next sections), and "uncooperative policies", which are not disclosed to the user by the reconciler. These two policy types are labelled respectively C and U in Figure 15. More details will be provided in later WP4 deliverables.

6.3 MSPL analysis

MSPL policies may contain *anomalies*. Anomalies are defined in literature as the occurrence of contradicting rules or never activatable rules (redundant rules) [14]. Anomalies may appear because the user

[7] Theoretically, in the most general case, policies can be prioritised to form a semi-lattice.

created them during MSPL policy specification, or because of the merging of MSPL policy statements obtained by refining the HSPL policy with the ones explicitly written by the user. Showing anomalies to the user is important as anomalies may be the evidence of specification errors. However, not all the contradicting rules are actually errors (as contradictions are an easy way to define the exceptions like 'allow access to all the internal websites but ws_1). Thus MSPL policies need to be examined by the users.

Therefore, the SPM provides the *MSPL analysis service*, which analyses user's MSPL policy for anomalies. This anomaly analysis is named in literature intra-policy analysis, as it only works within one policy. This service takes as input the link to the user's repository, reads the MSPL policy and produces as output a list of identified anomalies that are shown to the user via the GUI[8]. For each anomaly, the MSPL analysis service produces a set of standard anomaly resolution actions that can be selected by the user. Moreover, the user can also manually resolve the anomalies by editing the MSPL policy (e.g. in case of big mistakes). In case of contradictive anomalies, one possible resolution action is to simply accept the anomaly.

Unfortunately, anomalies may occur also among different policies in the policy stack. In this case, the process is named inter-policy analysis. The MSPL analysis service provides support for inter-policy analysis. To perform the inter-policy analysis, together with the user profile link, the analysis service receives as input a set of links to other policies in the policy stack. Clearly, all the policies involved must be cooperative. Even in this case, the MSPL analysis service outputs a report including all the identified anomalies and a list of resolution actions. One difference with intra-policy analysis is that the only resolution actions allowed for this case are the ones that (1) imply accepting the anomalies or (2) operating changes into the user policy (other cooperative policies in the higher stack layers are not modifiable by the user).

6.4 Reconciliation of cooperative policies

When the policy will be actually enforced by the NED, contributions from different actors must be reconciled to obtain a single policy to be processed by the user PSAs. In such a case, there are two possible situations, either the end user's MSPL policy is not in disagreement with other policies in the policy stack or there is a difference in policies. SECURED foresees a process to automatically resolve all the differences. This process is named policy reconciliation, and it is performed by another service provided by the SPM, the *policy reconciliation service*. This service takes as input the link to the user profile and the user policy stack, and produces as output a new MSPL policy[9] where all the contradictions are solved (by means of reconciliation functions that take into account policy priorities and other). All policies to be reconciled need to be cooperative.

6.5 MSPL policy translation

The reconciled MSPL, obtained as output of the policy reconciliation service, needs to be translated to be actually used by the user PSAs (the ones selected by the user in the application-driven scenario, selected by the HSPL in the policy-driven scenario). To this purpose, the SPM provides another service, the *MSPL translation service* that accesses the user's repository, reads the reconciled MSPL policy

[8] It is currently under evaluation whether the report must be stored in the user profile or just produced and consumed by the user (e.g. through the GUI).

[9] For optimisation purposes, the reconciled policy can be stored to avoid to be recomputed as it is expected that users connect to a limited number of NEDs.

and outputs a set of actual configurations for all the user PSAs. Every PSA is provided with a M2L translation component, which is able to perform the translation of MSPL policies for the given PSA. The MSPL policy translation service reads the user PSAs and coordinates the correct M2L invocation.

6.6 Chaining of uncooperative policies

Reconciliation is only possible with cooperative policies, as full access is needed for a policy to be processed by the reconciliation service. In scenarios with different administrative domains where reconciliation cannot be done (e.g. enterprise and government domains), an alternative approach is used to enforce the policies in the user policy stack, the *policy chaining*.

Chaining consists of redirecting the output of one set of PSAs in one administration domain (e.g. user TVD) to another set of PSAs in another administration domain (e.g. the ISP TVD). The user must not necessarily own the PSAs in other domains when chaining is performed. This is useful when more sophisticated controls are required by the entities that specify the higher policies in the stack.

The user is informed about other uncooperative policies in its policy stack. The user also receives a report indicating the MSPL rules that are affected (partly or completely overwritten) by chained uncooperative policies. This can be delivered either through a web portal provided by the PSCM during the authentication phase or alternatively through a notification sent to the SECURED app.

7 Design considerations for mobility

One of the aims of SECURED is to provide security services to mobile devices when they move across different types of access point, offloading these computational tasks away from the endpoints. Targeting mobile devices security implies that mobility is a key concept to be addressed in the project. Therefore, the design considerations that mobility introduces to the SECURED architecture have to be carefully analysed with the clear objective of having a general and flexible architecture, able to adapt to different mobile scenarios. This analysis of mobility is particularly applicable for certain BYOD scenarios, such as when users move from their office to their home, or from their office to a 4G network.

The security service provided by SECURED will be an on-demand service, which implies that all the security defined by a user has to be setup, deployed and enforced when the user connects. Likewise, if the user disconnects all the configuration regarding security needs to be removed. Furthermore, if a user moves, his security has to remain close to him which may result in migrating the security state across different NEDs.

Depending on the mobility patterns, SECURED considers two types of mobile users: nomadic and live-mobility users. These two classes are:

Nomadic mobile user. A user is considered nomadic when every time he connects to SECURED, being this connection the first time connection or caused by turning on the device in a different location, where the security is completely setup and instantiated from scratch. For example, a user working with a laptop at home idling the device while on the move, then restarting it once he reaches the office. In this case, it is not necessary to keep the connections alive. Even if it can happen that the state of some applications need to be restarted, but without requiring seamless mobility.

Live-mobility user. In this case a user requests seamless mobility, i.e. the security moves along with the user keeping its state, as well as the user data connections. For example, a live mobile user

may be a user working with a smartphone on the move. This implies the possibility of changing the access medium, i.e. from a WiFi AP to another, most probably implying a change in the NED providing the SECURED service. Then, in the live mobility scenario, two main tasks have to be addressed: handling the endpoint handover (user perspective), and the migration of the user security state (SECURED infrastructure perspective).

Apart from the user classification defined above, two different mobile security deployments models are foreseen, which will allow addressing mobility in different ways. According to the location of the SECURED NED, two scenarios are foreseen (depicted in Figure 16):

Intra-NED user migration. A user migrates from a network access point to another, but both points are managed by the same NED (e.g. a NED managed by the ISP located at the edge of the core-network, or in the case of a split-NED). The network access technology can be the same or even change during the endpoint handover process, but maintaining the same NED. From a mobility perspective, this is a less demanding scenario because the user's security execution environment does not have to be migrated.

Inter-NED user migration. A user moves between two NEDs and his security requires to be migrated. In this case, not only the network access changes but also the NED where the user security is enforced. This scenario is more challenging and raises both the need to migrate the user's PSAs states and manage the endpoint handover, which imply multiple additional steps at the destination NED (e.g. policy reconciliation, as later described), all without loss of the ongoing connections on the end-device.

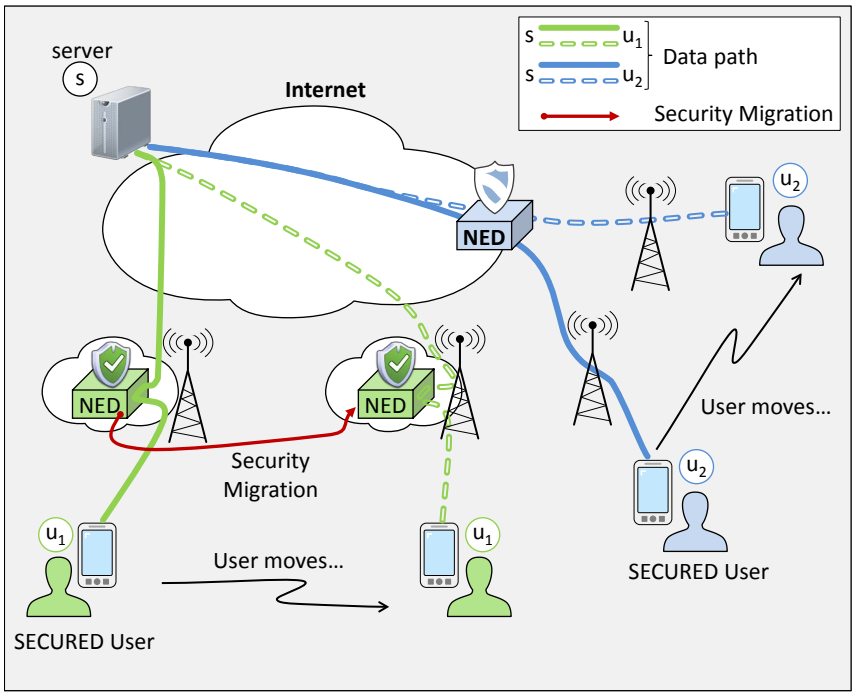

Figure 16: Mobile scenarios: inter/intra-NED migration.

As an initial definition of the mobility case in SECURED, it is considered that the mobile service will be offered by the same administrative domain, e.g. the same ISP or company is the one providing

the security service when the user moves. The scenario becomes more complicated for the case of providing SECURED through different administrative domains in a seamless and transparent manner. This implies that different domains require to perform a federation process to allow a roaming user to consume SECURED services when it connects to their infrastructure. At this early stage of research, the mobility part of this specification assumes that a user is always under the same administrative domain.

Within the monolithic NED implementation and the mobility case that considers switching the NED, the specification will need to be updated to directly address the inter-NED migration scenario described previously. Migration of security applications can be addressed in three ways:

Restart. When the user migrates, the destination NED recognizes him as new and triggers he normal authentication/attestation procedure. No state migration actually happens, and previous connections are lost. This case is related with the nomadic user mobility scenario.

Reactive migration. The user's state is migrated from the source NED to the target NED once the physical connection migration has already happened, i.e. the handover process already took place (late-migration). This can cause user's connections drop and service discontinuity because of the latency introduced by the environment startup and state migration. In this case the user will probably face a delay in the service provisioning.

Pro-active migration. The migration is somehow anticipated and the target NED can start setting up the execution environment before the actual handover takes place. This approach shorten the migration latency and allows to achieve seamless mobility.

SECURED considers that the pro-active migration is a more suitable approach compared with the reactive one. Furthermore, different mechanisms to trigger the migration process require further research. The next section presents some initial ideas regarding these mechanisms.

7.1 Pro-active approach

In the pro-active migration, the user's device or the infrastructure is able to recognize in advance the proximity of a handover point and trigger the migration to a designated best-NED target. Two main approaches are foreseen depending from which side the migration is triggered: either from the endpoint or from the SECURED infrastructure.

Device-triggered proactive migration. The user's mobile device is in charge of signalling to the target-NED that it is going to migrate. This implies an active role for the mobile device that will then have a context-aware agent/application running on it to communicate the mobility status to the infrastructure. The handover is not transparent for the user's device.

Infrastructure-triggered proactive migration. The source NED is able to recognize the upcoming user's migration based on location information provided by the network and/or the device and trigger the migration toward the target NED. The migration is transparent for the user, however, the endpoint is involved in the handover processes.

In both cases once the migration event is triggered, the source and target NEDs will communicate so that the target NED can retrieve the user's credentials and PSAs' status. Then, it will pre-allocate the

resources to host the user's environment and start it once the user actually migrates. It is possible that the anticipated migration will never happen. To avoid resource wasting, a roll-back operation will be provided, where the pre-allocated environment on the target NED may be destroyed upon the migration failure using timeouts.

7.2 Migration-aware applications

Due to the diversity and size of the applications, SECURED foresees the need of an application to optimize the migration process by declaring themselves as "migration-aware". Particularly, applications will be able to signal this feature inside their manifest. Then, these PSAs will be able to expose in a compact format all the current user's state, enabling a fast and efficient migration. This mechanism will be made available through the compliance to a specific SECURED API, which will be specified within WP5. As a fallback solution, non migration-aware applications will be really challenging from a mobility perspective because the only way to migrate the user's state will be to move the whole memory footprint of the application. This could not be feasible under certain circumstances (e.g. really huge states, specific virtual network configurations) causing the loss of the user state in case of migration.

7.3 Context aware policy reconciliation and mobility

SECURED allows the definition of policies at different domain levels in a hierarchical manner, i.e. the policies can be defined by governments, ISPs, companies and users. Therefore, a policy reconciliation process is required at each NED to resolve policy conflicts such as policies overriding. According to the context and location of the NED, the user policies and the context policies (joint referred as the policy stack) will have to be conciliated and as a result, a unique set of non-overriding policies will be obtained. These policies will be enforced on the NED user execution environment. However, once the inter-NED mobility case is included, the policy reconciliation may impact in the security state migration as described below.

The first and most simple scenario is where the inter-NED migration does not require a policy reconciliation at the destination NED. This case implies that when a user moves from one NED to another, the destination NED is under the same administrative domain and the policies obtained from the policy reconciliation are the same in both NEDs. Thus, the security can be completely migrated without the requirement of adding or removing applications in the user's security. However, the second scenario is when the policy reconciliation at the destination NED results in a different set of user's policies due to a change of context (e.g. moving from the home to a corporate network). This different set of policies may be enforced by a different group of applications (total or partially different), and the way that they are connected. Thus, it is important to further investigate how the migration of the user's security state will be solved in the latter case. Different approaches can be considered, e.g. a complete migration of the user's security state by migrating his personal execution environment, or a per application state migration. The latter aims to maintain the state of the applications that are re-instantiated at the destination NED. These two cases provide a different level of granularity which impose the complexity at the SECURED infrastructure for the case of complete migration, or in the developers for the case of a per application state migration.

8 Seamless mobility

In this context we understand mobility as the action of changing the network attachment point (either wireless or wired). Particularly, we focus on the idea of performing this change minimizing the disruption on the network service.

Then, handling seamless mobility may be regarded as a multi-layer issue, where all the different layers of the TCP/IP stack are involved, i.e. from the physical layer up to the applications all the different layers are directly or indirectly involved.

8.1 Physical and link layers

Starting from the physical and link layer levels, when dealing with mobile scenarios we can easily identify two different kinds of mobility, first, when the change of the network attachment point between two access points using the same technology, for example, from a 802.11 network to other 802.11, and a second when there is a change in the underlying technology, e.g. from 4G to 802.11.

In the first case, considering state of the art hardware, a mobile node must handover the connection from one attachment point to another, but, due to lack of hardware support, nowadays it is generally not possible to handover the connection without first closing the connection to the origin Access Point, and later spending some time negotiating the connection on the destination, i.e. a WiFi card is not able to associate itself to two different access points simultaneously. As a consequence, it is not possible to perform a handover without a transient, albeit short, disruption on the service. The good news though, is that hopefully such disruption will be alleviated by the higher layers.

In the second case, the handover can be smoothly performed at link layer, but with the associated change of IP network, the system must rely the seamless handover also to higher layers It is also worth noticing that some cell-phone vendors forbid within the firmware that a mobile device uses 4G and 802.11 at the same time, in which case we find a similar situation as the case above.

8.2 Network layer

As we just stated in the previous section, besides the connectivity disruption associated with the handover between two access points, another issue present in such environments is the disruption present when a change in the underlying IP addressing takes place. A consequence of that in the general case is the flush of all TCP connections when the physical link associated with them goes down. Within SECURED we may overcome this issue using two different approaches:

1. Reusing the virtual tap interface present in the IPsec tunnel between the client and the NED, while automatically recreating the tunnel on the destination using the same IP inside the tunnel. This may be accomplished by using a bridge, as we discuss later.

2. Use LISPMob during the handover, so the internal IP address of the whole procedure is maintained. This provides the added benefit of having a global mapping system that can assist in the persistence of the different connections using multiple providers, as long as LISP is supported on the different networks.

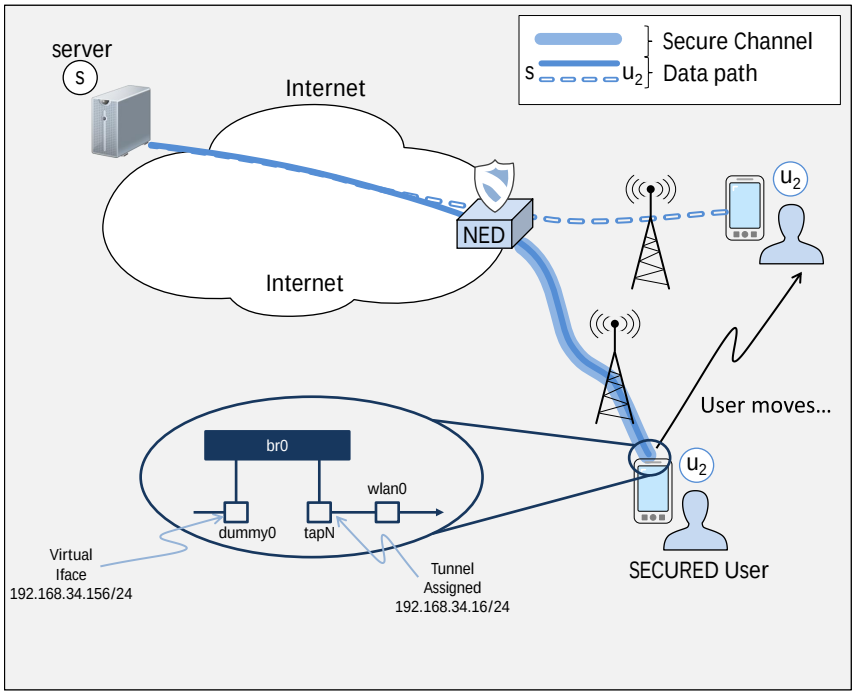

Figure 17: Bridged interface abstraction

8.2.1 Bridge approach

The idea behind this approach towards seamless handover is twofold, on the one hand, setting up a LispMob environment in a broad network is often complex and error prone. While on the other hand, in some scenarios where having a change in IP address can be avoided. Then, all the added complexity of LispMob is not necessary.

The technical detail behind this approach is shown in Fig. 17. The most important treat necessary of this solution is that it is necessary to provide a mechanism to avoid trashing all TCP connections when an interface goes down. This can be achieved by the presence of a bridged virtual interface, where there is a dummy virtual interface holding a fixed IP address, while the tap interface coming from the IPsec tunnel is renegotiated during the handover.

Then, the bridge allows to just forward all the traffic going and coming from the dummy interface through the tunnel, effectively allowing a neat solution for the mobility scenario.

In parallel to this, some other considerations must be taken. First is the assignment of this fixed IP address, which to simplify all the setup, should be unique to the NED and belong to the same subnet as the inner IPs of the tunnel, to allow the transparent forwarding without added static routes.

8.2.2 LispMob approach

LISPMob [11] is a protocol on top of LISP [12] aimed at the provisioning of seamless mobility. Mainly, it provides a framework to obtain independence of the actual IP addresses present on the physical interfaces through a smart tunnelling of all the traffic from the Mobile Node to the rest of the Internet.

The LISP Mapping System [12] is the main concept that uses the LispMob architecture. LISP is a protocol that separates the Location of a node in the network (RLOC) with its unique identification (EID).

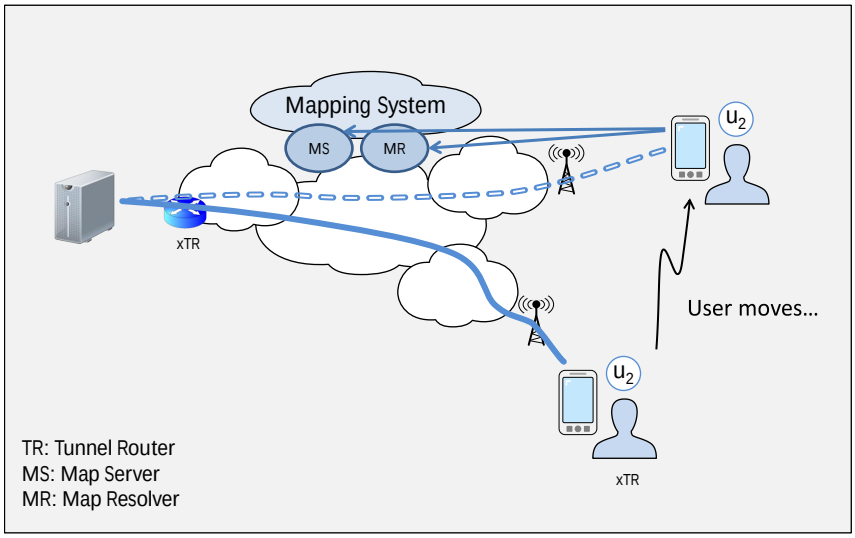

Figure 18: LispMob overview

Hence, the LISP mapping system is a publicly accessible service that publishes location information associated with EIDs (EID-to-RLOC mappings), it can be seen as a DNS-like service, where its main elements are Map Servers (MS) and Map Resolvers (MR). Both MSs and MRs are located in the border of the network providing an interface to the mapping system. EID-to-RLOC mappings are stored in Map Servers, and each Map Server is associated with a partition of the EID namespace. Moreover, Map Servers store the location information for those EID prefixes. Therefore, each LISP mobile node is associated with a specific Map Server where it registers its EID-to-RLOC mapping, and updates it according to its actual location. In this context, Map Servers have assigned a set of prefixes (EIDs) and delegate them to either LISP tunnel routers or mobile nodes (Fig. 18).

As it can be observed, besides the mapping system, LispMob relies on tunnels to obtain its location independence. The entity in change of such tunnel management are xTR. xTR is the Ingress (or Egress) Tunnel Router, where all the connections are forwarded. To simplify the LISP model, in LispMob the mobile node acts both as the client and the Tunnel Router, directly managing the addresses from the same device.

LispMob is a promising technology. However, the main issue present in this case is the fact that to provide support for mobility for connections to the outside world it is necessary to have a LISP enabled network, or the usage of proxies within the company's domain.

8.3 Transport layer

The transport layer may be non-connection oriented, i.e. UDP, or connection oriented, i.e. TCP. In the mobility environment, when the network layer address doesn't change, using non connection oriented protocols the handover will not cause any disruption in the transport layer of the service as the communication may continue as before. So, in practice, if the proper techniques of network layer are in place the connectivity will be maintained.

In the case of connection oriented protocols and no change in network layer address the perduration of the connections is dependent on the operating system implementation, e.g. Android cell phones generally drop all TCP connections when there is a network disruption, implying that is the role of the

application to regenerate them and resume the work. Opposed to that, in the case of Linux operating systems, the TCP sockets are subject to timeout before being expired by the kernel itself. Therefore, if the handover is quick enough the sockets may be refreshed as long as the netwkork layer address does not change.

In the case that there is a change in the underlying network layer address, by default all the connections will get dropped, as all the sockets become invalid, in fact that's the whole idea about having LispMob solutions in such environments, avoiding the change in IP addresses. To overcome this hard restriction in mobility scenarios, there are alternative solutions using transport protocols such as MPTCP [7], that are devised to further detach the transport layer sockets from the underlying Network Layer Address. The drawback of this solution is the required support from the server side. However, we plan to further investigate this solution during the third year of the project.

8.4 Application layer

Finally, mobility may also be handled by the same applications making use of it. This is specially true in devices such as Android, where network connectivity is tied to the internal event system of the operating system, i.e. `Intents`, allowing the application to be aware of the network status. Hence deciding when to refresh existing connections or initiate new ones to recover the application status.

The approach of leaving the converging work to the application has some benefits:

- *Application awareness* – The application itself is aware about its status and can alleviate the migration by caching its contents. Examples of this is the Google Docs Suite, where the user session is automatically refreshed by the browser itself, and during the convergence, the system caches all keypresses and presents an offline view of the document being edited.
- *Simplifies the overall approach* – Leaving the burden to the applications alleviates the rest of the system, as in lower layers in the hierarchy tend to be more generic, thus requiring more complex solutions to solve the problem.

Even with the above, just using applications to overcome this problem generates another set of drawbacks:

- *Mandatory application support* – Without support from each application the user experience will require to restart the applications (or manually refresh their status), manually configuring the security, and as a consequence having to wait for long periods of time for all to converge.
- *Poor security support* – In environments such as SECURED, where a secure channel is present, when switching to another access point, besides the association to it, it is required to renegotiate the channel, and of course, in the case of having associated PSA the migration to the new attachment point.

The conclusion of all this is that a mixed approach is necessary and we follow this solution within SECURED.

9 Interoperability and architectural extensions

This report presents an early specification for the SECURED architecture, which will be revised with subsequent future versions, and should be treated as a preview.

9.1 Interoperability standards

In the SECURED context, interoperability means that any device can use a NED and that, idealy, any user security control can be enforced by a NED. The connectivity interoperability is achieved through the transparent traffic delivery and the interoperability of the underlying network stack (e.g. TCP/IP, 3G/LTE/WiFi). SECURED leverages three features to address the enforcement of interoperability:

- the policy–driven configuration allows the user security controls to be executed on any NED (functional interoperability);
- the model–driven approach avoids the need to define an API which could end up not being future–proof (future interoperability);
- the PSA integration model defines a limited API to interact with the PSA and uses a network abstraction for handling the user traffic (execution interoperability).

The first two solution proposed are research areas that will be explored during the project, and therefore not part of this alpha version.

The execution interoperability feature aims at providing a way for a NED to execute any PSA[10].

9.2 Multi-domain support

With regards to the ecosystem (i.e. the way NEDs, PSAM/PSAR, SPM, etc. are deployed in a SECURED domain), this version of the document is limited to a unified ecosystem, where every component is part of the same managed domain. Opening the ecosystem to encompass marketplaces and FOSS repositories will require improved interoperability definitions and is one of the main extensions for the architecture.

9.3 Distributed model

The second main extension for SECURED focuses on the NED itself. In the initial vision, the NED is a single computational node which can be moved depending on the network topology and the scenario; it can be the edge device itself (monolithic NED) providing standard routing services and capabilities, nearby (split NED) e.g. using a software-defined network, or in a specific place known by the user device (virtual/remote NED).

As more enterprises and network providers move towards managed private cloud and miniature data-center installations, the more capable businesses will be in deploying computational NEDs. However, keeping resources confined to an individual node creates scalability challenges for performance and administration. SECURED will aim to address an additional paradigm where the NED itself can be decomposed into a distributed system across multiple nodes, in which it will be able to scale to service a larger numbers of users. This extension addresses big enterprise network use cases, servicing for small and medium businesses, and large residential areas in an ISP setting. This provides consolidated compute in big collections of managed computing nodes in a central place, rather than transforming

[10] SECURED does not aim at solving the problem of ubiquitous execution of an application: if a PSA is designed to be executed in a given environment (e.g. GNU/Linux), a NED without this environment will not be able to enforce the corresponding security control, hence, SECURED is likely to promote a template TEE for a common PSA platform.

each edge device into a separate split-NED. From the user's perspective there is no difference between a physical NED hosted by the ISP and an NFV-based distributed NED.

The current foreseen model retains the fundamental architecture with the addition that physical boundaries and connections can become virtualised. The design of the SECURED architecture is flexible enough that it easily aligns with the current ETSI NFV architecture[15], illustrated in Figures 20 and 19. However, as physical boundaries and connections become virtual, a number of security challenges arise relating to topics such as remote attestation and authentication, which will be explored in the relevant work packages.

In the current architecture iteration the PSC is a single, simple and static controller for the user's TVD. SECURED will investigate an evolved PSC which can provide network programmability and act as an SDN controller for the user's TVD, with a constraint of Northbound and Southbound interfaces (e.g. a software TVD vSwitch in the current models, or a physical switch in the distributed model) that can still satisfy security and isolation requirements. This has the potential for future vendors or open-source developers to provide additional innovation to the SECURED platform. Furthermore, inter-process communication frameworks for co-operating PSAs (with potential performance improvements) may be a likely research area that requires investigation for subsequent architecture versions.

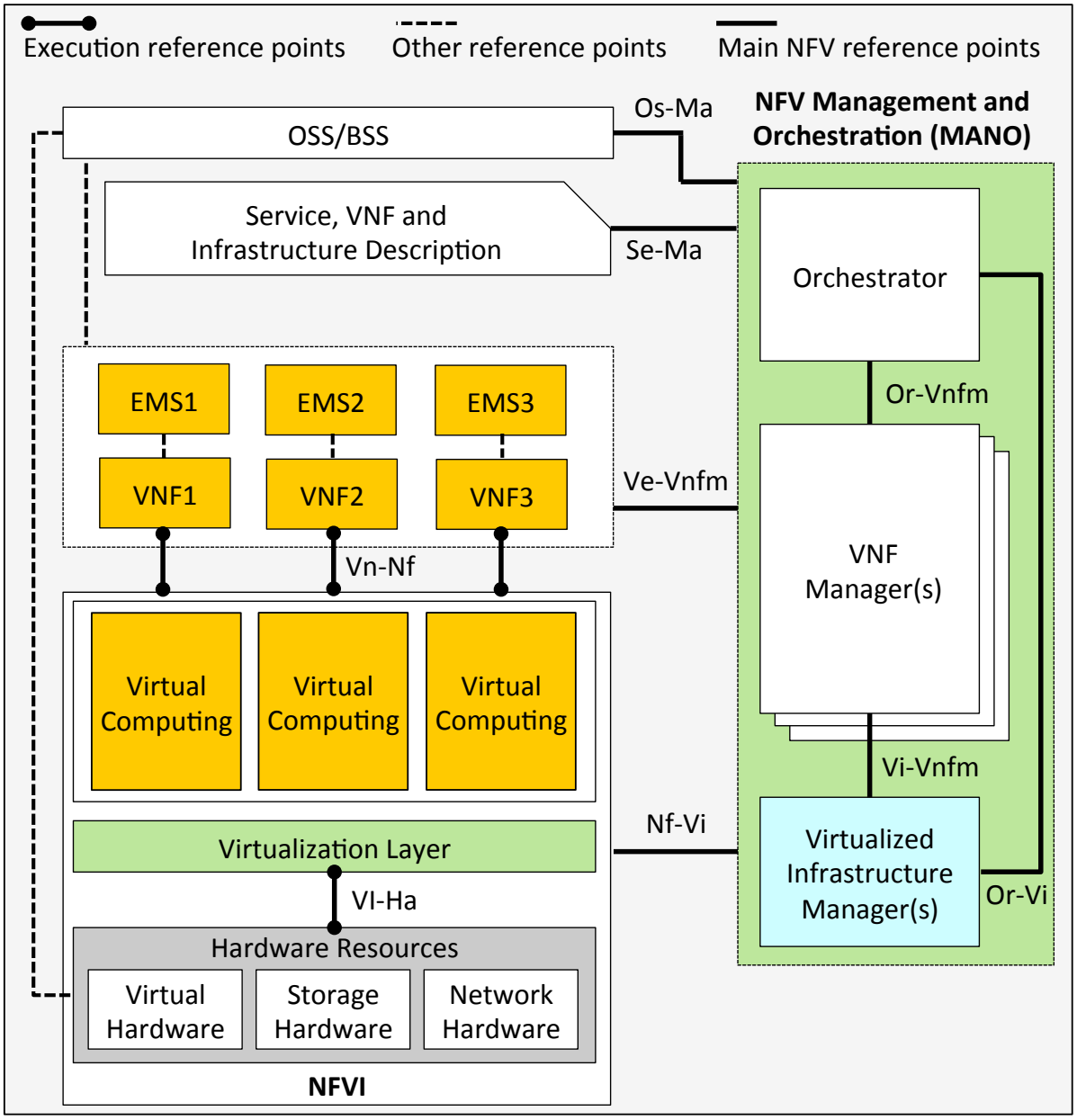

Figure 19: The ETSI NFV architecture framework.

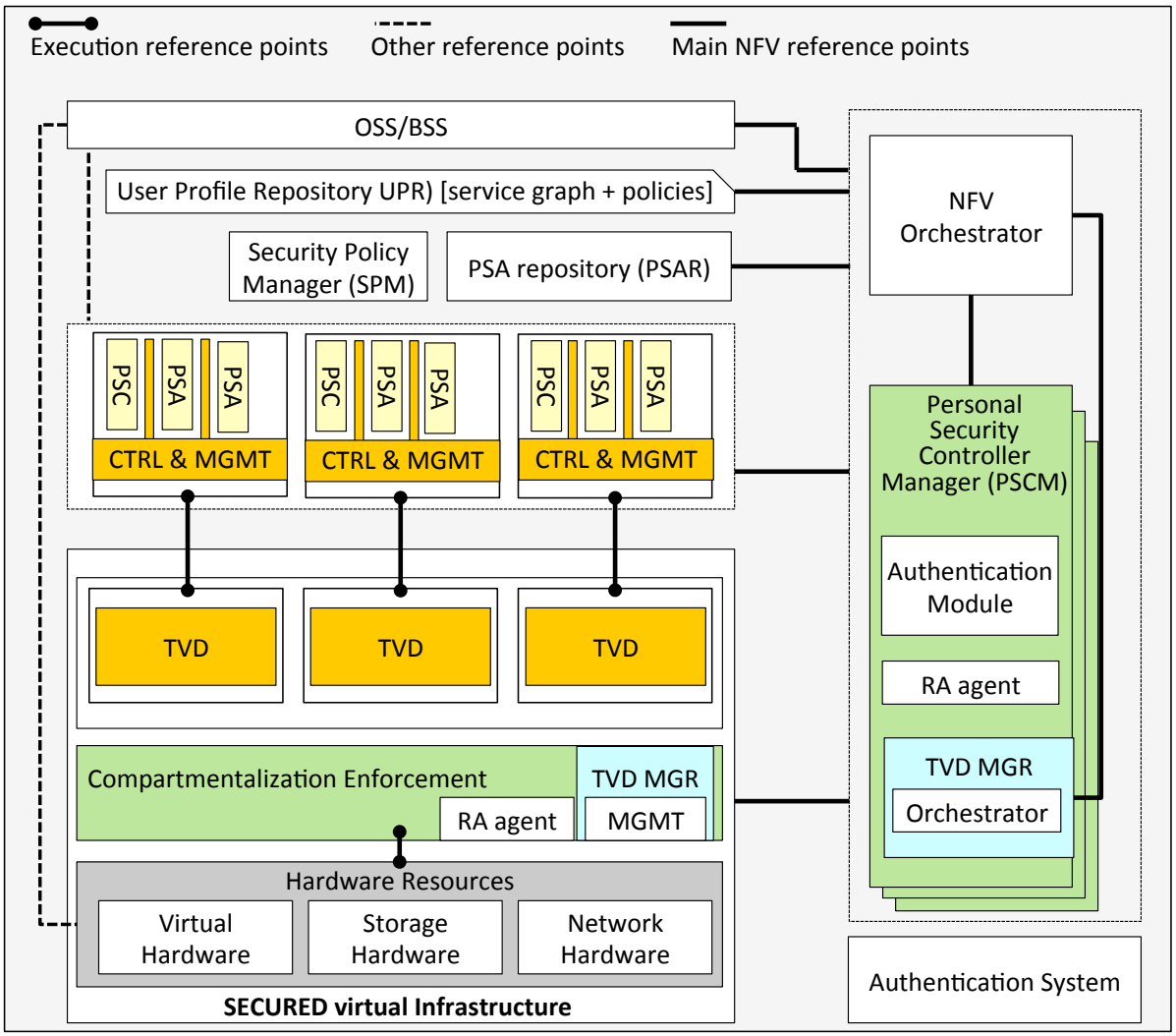

Figure 20: SECURED architecture alignment with the ETSI NFV architecture.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, March 2008, DOI 10.1145/1355734.1355746

[2] SECURED project, "D2.1 – User requirements," January 2014

[3] N. Williams, "On the use of channel bindings to secure channels," RFC-5056, November 2007

[4] SECURED project, "D3.1.1 – NED specification (alpha)," July 2014

[5] SECURED project, "D5.1 – PSA specification," July 2014

[6] SECURED project, "D5.2 – Specification of PSA management service and repository," October 2014

[7] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses", RFC-6824, January 2013

[8] SECURED project, "D4.1 – Policy specification," April 2015

[9] SECURED project, "D4.4 – Policy support tools," June 2016

[10] SECURED project, "D4.2 – Policy transformation and optimization techniques", September 2015

[11] A. Cabellos, A. Rodriguez Natal, L. Jakab, V. Ermagan, P. Natarajan, and F. Maino, "LISPmob: mobile networking through LISP", lispmob.org whitepaper, December 2011, http://downloads.lispmob.org/LISPmob_Whitepaper.pdf

[12] D. Farinacci, D. Lewis, D. Meyer, and V. Fuller, "The Locator/ID Separation Protocol (LISP)", RFC-6830, January 2013

[13] POSECCO project, "D3.3 - Configuration meta-model," http://posecco.eu/fileadmin/POSECCO/user_upload/deliverables/D3.3_Configuration_Meta-Model.pdf

[14] E. Al-Shaer and H. Hamed, "Discovery of policy anomalies in distributed firewalls," *INFOCOM 2004*, Hong Kong (China), March 7–11, 2004, pp. 2605–2616, DOI 10.1109/INFCOM.2004.1354680

[15] ETSI, "Network Functions Virtualisation," http://www.etsi.org/technologies-clusters/technologies/nfv

Appendix A – Abbreviations

AAA Authentication, Authorisation and Accounting

CA Certification Authority

HSPL High-level Security Policy Language

ISP Internet Service Provider

M2L Medium-to-Low level translation

MSPL Medium-level Security Policy Language

NED Network Edge Device

NFV Network Functions Virtualisation

OSS/BSS Operations Support System / Business Support System

PKC Public-Key Certificate

PSA Personal Security Application

PSAM PSA Manager

PSAR PSA Repository

PSC Personal Security Controller

PSCM PSC Manager

RA Remote Attestation

SDN Software-Defined Networking

SECURED SECURity at the Network EDge

SPM Security Policy Manager

TEE Trusted Execution Environment

TVD Trusted Virtual Domain

TVDM (MGR) TVD Manager

UPR User Profile Repository

VM Virtual Machine

VPN Virtual Private Network

vNED virtual NED (software-based)

xDSL Digital Subscriber Line

Appendix B – Abstract service graph

This appendix contains an example of a service graph.

```
"USR-SG" : {
  "id" : "user1_123",
  "PSAs": [
    {
      "type" : "firewall"
    },
    {
      "type" : "web_cache"
    },
    {
      "type" : "antivirus"
    },
  ],
  "flow-rules" : [
    {
      "flow-space" : {
        "port" : "eth0",
      },
      "action" : "firewall:1"
    },
    {
      "flow-space" : {
        "port" : "firewall:2",
        "app_proto" : "web"
      },
      "action" : "web_cache:1"
    },
    {
      "flow-space" : {
        "port" : "web_cache:2"
      },
      "action" : "antivirus:1"
    },
    {
      "flow-space" : {
        "port" : "firewall:2"
      },
      "action" : "antivirus:1"
    },
  ]
}
```

Appendix C – Keys and certificates life-cycle

In this section we describe the necessary certificates life-cycle management, based on their usage: infrastructure specific, user-facing certificates and NED certificates.

C.1 Infrastructure certificates

For infrastructure's internal repositories (such as the UPR, PSAR, . . .) the main purpose of their PKC is to secure through TLS the corresponding REST services APIs. As such their management is traditional and depends on the kind of infrastructure.

In case of a closed one (NED and support infrastructure owned and managed by a single entity) all certificates can be issued by the same CA, which can be a public or private one.

In case the various components are owned or managed by different entities, then in principle their certificates could be issued by different CAs but this would cause the usual problem of mutual trust recognition. The idea of a single root CA to issue all the PKCs needed for the NEDs and the elements in the support infrastructure would be appealing because would simplify trust relationships and strengthen the architecture against various threats. However this would come to a price: need for a global agreement, which may have also commercial implications (possibly moderated by creating dedicated intermediate CAs based on a per organization or per functionality partition).

C.2 User-facing certificates

There are three main component that a user can directly access:

- the PSAM web portal, for registering and configuring her profile;
- the NED used to secure her Internet access;
- the verifier which attests the NED.

The PSAM portal is the most relevant one for the end user: as it will be accessed with a standard browser, it should use a PKC issued by one of the CAs commonly installed in all browsers.

Towards the user, the NED will expose its PKC in two cases: when hosting the web-based interface of the PSC and when a trusted and secure channel is created (e.g. IPsec with PKC authentication of the NED). The former case requires a NED certificate issued by one of the CAs commonly installed in all browsers (as would happen to the verifier if directly contacted by the end user) The latter case is a bit more complex. For the end user it's not a concern as the channel will be created by a specialized application (not a normal VPN client as it must verify the result of remote attestation too) which will be equipped with the right trust anchor (that should be immutable or contain an update mechanism to periodically download an updated list of trust anchors). However the SECURED application must also be able to trust the answers provided by the verifier, so the simplest solution is to have both the NED and the verifier certified under the same trust anchor (directly or indirectly).

C.3 NED keys and certificates

The NED must possess the keys and certificates needed for remote attestation. As this will be performed only against the verifier(s), it's just this component that must enter in a mutual trust relationship with the NED (as the NED may not by willing to share its integrity status to unknown or untrusted verifiers).

With respect to NED authentication towards the infrastructure, it could use the same certificate used towards the verifier, as the infrastructure may need anyway to attest the integrity status of the NED before accepting it inside its network (especially if the NED and the infrastructure are managed by different entities).

NED certificates towards the end user have been discussed in the previous section.