Exploiting the network for securing personal devices

Chris Dalton(*), Antonio Lioy(+), Diego Lopez($), Fulvio Risso(+), and Roberto Sassu(+)

(*) HP Laboratories, Bristol, United Kingdom
(+) Politecnico di Torino, Dip. Automatica e Informatica, Torino, Italy
($) Teléfonica I+D, Madrid, Spain

Abstract. Personal devices (such as smartphones and laptops) often experience incoherent levels of security due to the different protection applications available on the various devices. This paper presents a novel approach that consists in offloading security applications from personal devices and relocating them inside the network; this will be achieved by enriching network devices with the appropriate computational capabilities to execute generic security applications. This approach is fostered by the SECURED project, which will define the architecture, data and protocols needed to turn this vision into reality.

Keywords: Network-based personal security, personal security protection, remote attestation, network functions virtualization

1 Introduction

The recent years have witnessed an increasing number of user terminals (such as laptops and smartphones) being connected to the Internet and we foresee an even more exciting growth in the coming years, due to new functions such as car infotainment systems, smart Internet-of-Things (IoT) devices, and more. This scenario encompasses a high number of devices with very different capabilities and hence poses significant challenges in terms of security, particularly with respect to protection from external threats.

First, many devices have limited resources, particularly embedded and mobile devices, and are often further constrained by severe limitations in terms of power consumption. As a consequence, complex protection applications (like anti-virus or VPN client with strong encryption) may not be executed on all devices.

Second, users can access the network from anywhere, hence they experience different levels of protection depending on the network they are connected to. For example, a user is typically exposed to more threats when connecting from a public hotspot than when connecting from the corporate network (as it usually includes a sophisticated border firewall).

Last but not least, the level of protection depends upon the security applications available for a specific terminal. For example, a laptop can be equipped with a powerful parental control, while the same software may not be available when browsing the Internet from a smart TV, hence leaving kids unprotected.

This paper proposes a possible solution to the above problems, based on a *network application offloading* approach [7]. In a nutshell, we move protection from the user terminal to the (closest) network edge device (NED), which can be represented by an access point, switch or router, augmented with the computing capabilities required to run the offloaded security applications. According to this approach, users will configure the desired security countermeasures (applications and policies) only once, then they will be applied automatically by all NEDs regardless of the user terminal and network connection.

The main advantage of this approach consists in transforming protection from device- or network-based into a new user-centric paradigm, hence delivering personalized protection independent from the user's device and location. In addition, this would no longer require to install specific software on each terminal, which simplifies management and reduces power consumption, hence offering to devices with limited capabilities the same level of protection of more complex platforms. This approach is fostered by the project SECURED[1], which is currently designing the technical framework to turn this vision into reality.

2 Requirements

Running personal security applications into the network is a sensitive action and several requirements must be met by an architecture aiming to reach this target.

2.1 Security requirements

Trust. Since applications would be executed at a node 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 to be an original device and not one 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 could manipulate the traffic of the victim at his will.

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 we do not exclude the possibility to accept other sources of trust. For instance, the user might be satisfied with trust originating from non-technical considerations, such as having the physical control of his home gateway or a contractual agreement (and corresponding liability) with his ISP.

[1] http://www.secured-fp7.eu/

Channel protection. If the user trusts a NED, he must also create a protected channel with it, so that attackers cannot manipulate the traffic. In addition, he must ensure that the other channel endpoint is the same entity that presented the trust proofs, otherwise an attacker could perform a man-in-the-middle attack by relaying the proofs 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 selected by that user. 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.

2.2 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, although the NED could use information exchanged during that 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.

Standardized platform. Since a security application could run on an arbitrary NED (e.g. home gateway or corporate switch, depending on the location a user connect 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).

Standardized 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.

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.

3 The SECURED infrastructure

3.1 NED deployment 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.

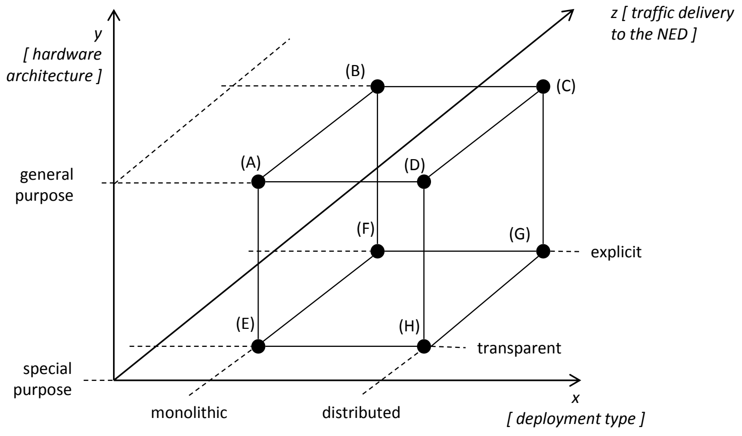

Fig. 1. Dimensions for the possible NED deployment scenarios.

The first dimension considers two possible hardware options: components engineered for data plane processing (e.g. network processors, hierarchical memory architectures, hardware accelerators) versus standard 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 cloud-like infrastructure.

Concerning the second dimension, we distinguish the NED as a monolithic component that implements all the core functions from the case in which the NED functions are distributed across multiple elements. For example, a traditional router without advanced computing capabilities might redirect the user traffic (e.g. through OpenFlow [5]) 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 has to handle a single box), while the distributed model can guarantee better scalability and is more oriented to cloud-like environments.

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), we foresee also the case of a user connecting to an untrusted or legacy network. In this case we provide a small agent operating on the user device to establish a secure tunnel to a remote NED and delivers all the user traffic to it ("explicit" traffic steering).

The Cartesian product of these three dimensions (with two options each) generates the eight points in Figure 1, corresponding to possible deployment scenarios. Among the different possibilities, labelled (A) – (H) in the figure, we discuss now those that we consider most promising (Figure 2).

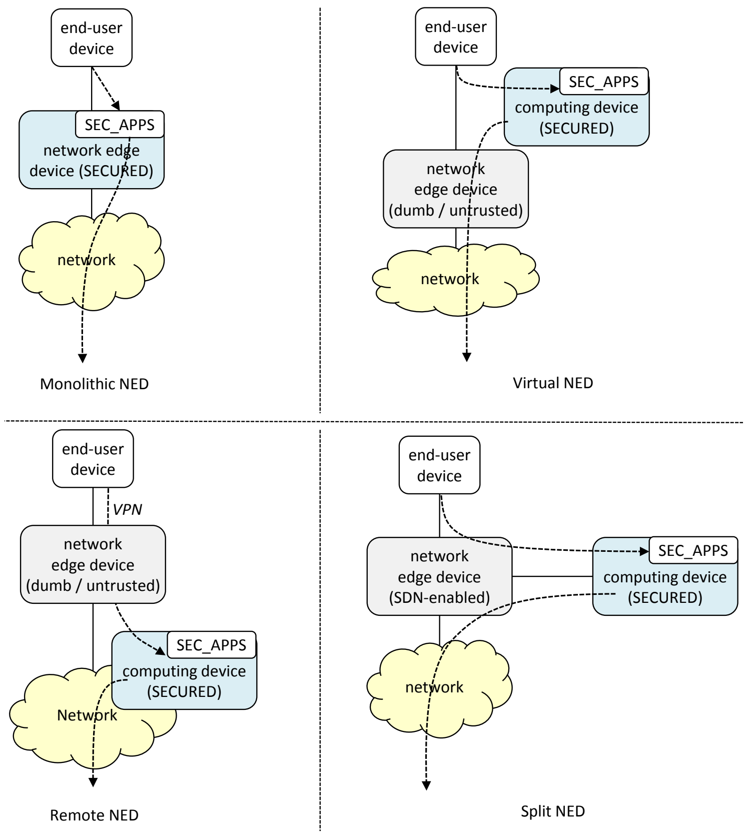
end-user device
SEC_APPS
network edge device (SECURED)
network
Monolithic NED
end-user device
SEC_APPS
computing device (SECURED)
network edge device (dumb / untrusted)
network
Virtual NED
end-user device
VPN
network edge device (dumb / untrusted)
SEC_APPS
computing device (SECURED)
Network
Remote NED
end-user device
network edge device (SDN-enabled)
SEC_APPS
computing device (SECURED)
network
Split NED

Fig. 2. Some SECURED deployment scenarios.

Monolithic NED (case E). This is the case of a high performance appliance (e.g. HP 3800 series) directly connected with the user device and containing a network router with a custom computational unit in the same hardware box.

Split NED (case D). This represents a traditional access router directly connected with the user device and redirecting the traffic to a general purpose server (e.g. via SDN technologies such as OpenFlow), which executes the security applications. This model could work also on legacy networks, when traditional routers are coupled with a companion server that takes care of the processing.

Virtual NED (case C). This is the case where a local compute node, under user control (e.g. a home desktop), is equipped with the NED software and acts as a communication gateway for all user's devices. User terminals have to connect directly to the virtual NED (via the local network, if trusted by the user, or by means of a secure channel) by explicitly redirecting their traffic to this box.

Remote NED (case B). This point represents the case in which the user terminal explicitly connects to a remote NED through a secure channel (e.g. a traditional virtual private network): in this case we would depart from our philosophy of not requiring any modification to the client as we need to install a custom application at the user terminal. This approach would incur penalties both in management (necessity of a VPN client) and performance (additional computations performed at the terminal and non optimized routing through the remote NED). However we consider this case as a form of "last resort" option if the user connects to a legacy network without SECURED capabilities: this case should be rare as many modern routers already support some protocols that enable the implementation at least of the split NED option.

3.2 Providing trust

Regardless of how the infrastructure is implemented, the most important aspect from the user's perspective is that the NED must be able to process the traffic as expected and must prove this to the user. The problem is how to guarantee to a user that, when he connects to a network, the traffic will be processed by a SECURED device. Indeed, a user may connect to a legacy network (without NEDs) or to a NED that has been previously compromised: in these cases users are exposed to possible threats. To avoid this situation, SECURED exploits the Trusted Computing technology, in particular the *remote attestation* procedure.

The Trusted Computing Group (TCG) defined the specifications of a cryptographic chip, the Trusted Platform Module (TPM) [4], which uniquely identifies a Trusted Platform (TP). The TPM contains the necessary primitives to record measurements (i.e. fingerprints) of hardware and software components, to protect measurements integrity while they are stored at the TP, and to securely transmit them to a verifier. The latter can evaluate, from received data, if the TP will perform the requested tasks as expected: this procedure is known as remote attestation.

If the user remotely attests a NED before sending network traffic to it, this prevents the threats described above:

- in case of a legacy network, a device cannot prove to the user that it belongs to a SECURED infrastructure since this proof requires the use of an asymmetric key (Attestation Identity Key), which belongs to a unique TPM and whose private part is never exposed outside this chip;
- in case of a compromised NED, the TCG methodology ensures that the user can reliably detect if the accessed device will not properly process his traffic.

However, the sole attestation of the NED is not enough to protect users against attacks from other users connected to the same network. Indeed, attackers may try to intercept or modify the communication between the user and the attested NED. Although a secure channel is appropriate to overcome this problem, this does not ensure that the endpoint contacted by the user is the attested NED: as pointed out in [2], the endpoint may be a device controlled by an attacker relying the attestation to a NED. To fight this threat, SECURED employs a trusted channel between the user terminal and the NED and investigates which solution is the best fit for this goal (e.g. [1] or [8]).

3.3 Security policies

Secured allows to describe user security requirements via a High-level Security Policy Language (HSPL). HSPL is a user-oriented language suitable for expressing concepts related to end-point protection, which represents a departure from current languages that are either related to network filters (for border firewalls) or to access control (for database and applications). This language is appropriate for capturing the user requirements but cannot be directly implemented by security controls. As a consequence, we translate HSPL into a medium-level security policy language (MSPL) which conveys the same information in an application-independent format suitable for configuring security controls, typically an ordered sequence of permit and deny actions related to matching packets or payloads. A final translation step is needed from the MSPL to the application-dependent languages that are needed to configure the actual security controls (e.g. the Linux iptables firewall or the Snort intrusion detection system).

The Security Policy Management service (SPM) allows users to create, delete, edit, view, store and save their security policies. Each user may have more than one set of policies (associated to different personae) to differentiate the level of protection according to the security level required for a certain type of work. The SPM is also the main user interface to select security applications (or Personal Security Applications, PSA in short), either directly (in case of an expert user that prefers an application-driven security configuration) or indirectly (in case of a user preferring a policy-driven security configuration and thus selecting applications among those that offer the capabilities needed by his policy).

Once the policy has been specified with MSPL statements and PSAs with the required capabilities have been selected, we still need to create the configuration files for the PSAs. This is done by invoking the Medium-to-Low level (M2L) translation service associated to each PSA: it transforms a policy expressed in MSPL in the configuration format required by the specific application.

4 The Secured architecture

Figure 3 displays the Secured architecture. We now proceed to explain how the application offloading can be realized, examining first modifications at the user terminals (to recognize if it is attached to a Secured network) and then introducing the main components inside the NED. Finally, we describe at high level the steps required for a user to setup a network connection with Secured.

4.1 User terminal

If the user may access either a Secured infrastructure or a legacy network, he must install on his devices a small monitoring application, the Secured app. This application is activated each time the device attaches to a new network to check that the connection is to a trusted and secure NED. In case this condition

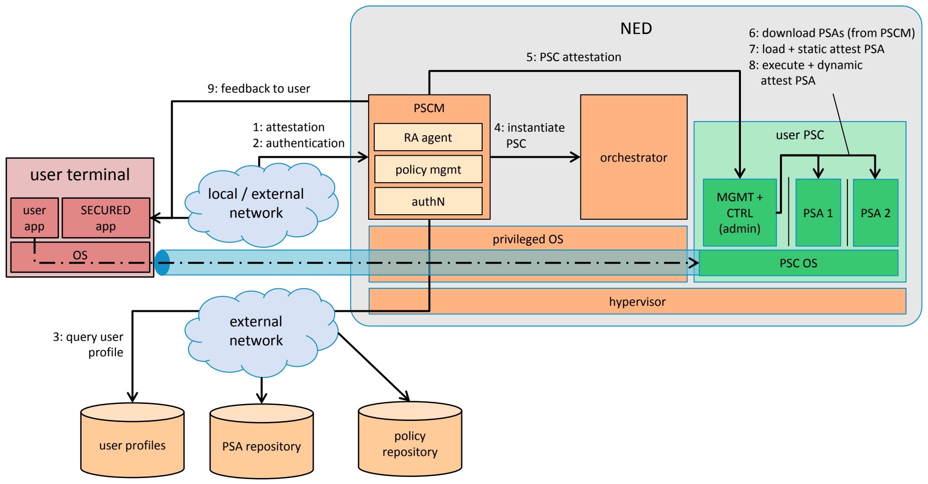

Fig. 3. Overview of the SECURED architecture.

is not verified, the SECURED app establishes a remote connection (e.g. VPN) to a trusted NED and redirects all the traffic of the user terminal to the remote network node, hence guaranteeing the expected level of protection although with a higher latency.

It is worth nothing that the above application is not mandatory. For instance, we foresee the case of devices which cannot install this application. These devices are still compatible with the SECURED model, although they may not have access to additional features such as the possibility to trust the NED or to automatically connect to a remote NED in case of a legacy network.

4.2 The NED

Within the NED, each user is provided with a Personal Security Controller (PSC), a logical container of execution environments (e.g. virtual machines) that will coordinate the execution of his security applications into the network.

The PSC can run either directly on the network device (monolithic NED) or on a separate computational unit (split NED) When a user connects to the network, the NED will create a new PSC and download on the created container the security applications (PSAs). When ready, the new PSC will operate on the sole traffic of the user.

Two main NED components are involved in configuring the PSC: the Personal Security Controller Management service (PSCM) and an orchestrator.

The PSCM is the component contacted by users to setup a connection with SECURED and contains three main modules. A *Remote Attestation Agent* is in charge of executing the remote attestation protocol with the user and reporting the integrity status of a NED. An *Authentication Module* requests to a connecting user a proof of his identity to retrieve the user profile (policies and applications). The *Policy Management* component performs harmonization and conflict resolution on the policies extracted from the user profile.

After the PSCM determines the configuration of the user connection, it contacts the orchestrator to start a new PSC. The orchestrator determines, depending on the requirement of the PSAs chosen by the user, the number of virtual machines that must be created for that PSC to process the user traffic. Also, its role is to configure the network paths inside the NED (to connect together the virtual machines forming a PSC) and inside each virtual machine (to send the traffic from a PSA to another). Finally, the orchestrator monitors the integrity of the PSC, detects whether there are communication problems between virtual machines of the same PSC and if a PSA inside a PSC crashed; if one of these events occurs, the orchestrator may request the hypervisor to restart the virtual machine causing the problem.

4.3 Connection set-up

When a device connects to a NED the following steps are performed to create a protected network connection.

1. Front-end attestation. The user terminal has to perform a remote attestation pass toward the NED to verify that is connected to a trusted device running the expected software.

2. User authentication. This step aims at discovering the identity of the user connecting to the network, which is needed to retrieve his personal security profile. This could be integrated with existing authentication mechanisms that are already active for network access, such as the 802.1x protocol[2] or SIM-based authentication in mobile networks. This way the user would perform a single authentication, both for network access (not requested by SECURED) and to retrieve the user profile.

3. Retrieval of the user profile. Upon successful identification of the user, the PSCM fetches from a server the user security profile, which contains the list of PSAs to be executed and their calling order. Then the PSCM contacts the PSA repository to retrieve the application characteristics, such as their execution model (e.g. full fledged virtual machine, Java virtual machine, Linux container) and hardware requirements (e.g. CPU and memory). This information is needed to create a precise view of the computing/networking primitives to be set up, which includes the execution environment themselves, the PSAs, and the network connections between the previous components to satisfy the desired service order.

4. Setup of the user PSC. Giving the execution graph created in the previous step, the orchestrator issues the proper commands to create the required computing resources and properly connect them. These resources are grouped under the term Personal Security Controller (PSC), which may include different execution environments based on the requirements of the PSAs. In this step no

[2] While the 802.1x protocol was originally intended to perform device authentication (e.g. based on the MAC address of the user terminal), recent extensions allow to perform this step based on user-defined credentials, such as username and password.

PSAs are installed, as the user has to perform an additional verification step to make sure that his PSC has been set up properly.

5. Attestation of the PSC. The user completes a remote attestation phase to verify the correctness of the PSC (albeit limited to computing and networking resources), making sure that the execution environments are trusted and that traffic will traverse those components in the expected order.

6. Download and install applications and policies. PSAs are downloaded from the repository and installed in the execution environment. Furthermore, policies are retrieved from the user profile and applied to the applications.

7. Loading and attestation of the PSAs. PSAs are loaded in memory and are statically attested to verify the correctness of the applications themselves.

8. PSA execution. PSAs are launched and operate on the user traffic. Possibly, a dynamic attestation step can be carried out on the whole PSC (execution environments, network connections, PSAs) to detect run-time attacks.

9. Feedback to the user. Finally, the user is notified that all steps have been successfully completed and the user PSAs are operating properly. A dynamic feedback is optional but strongly desirable to notify users about possible changes (e.g. when moving from a network to another, hence the PSC moves to a different NED, or in case of any problem such as a crashed PSA or network issues) .

Note that the user is required to complete the setup of his profile before being able to connect to a NED. This requires the user registration in the profile server with a valid account, selection of the proper PSAs and definition of the desired policies (following either the policy-driven or application-driven approach).

5 Evaluation and conclusions

As evident from the discussion above, the execution model chosen for our network application offloading schema is compatible with the service model proposed in ETSI by the Network Functions Virtualization (NFV) group. This is a recent framework for the provision of network services by virtualization techniques [6] and many operators are looking at it with increasing interest. As such, it is of high interest also to SECURED as a target environment for its implementation. NFV is based on the availability of a homogeneous infrastructure, supporting the deployment, replication and mobility of software-based implementations of the different network functions, named VNF (Virtual Network Function). Network services are built by composing VNFs and deployed by the NFV Orchestrator upon the virtualized infrastructure.

NFV can support an additional SECURED model, the **Distributed NED**, which can be seen as the generalization of the Split one. In this case the NED is composed by several distinct processing components deployed in different locations, such as a dedicated server in the enterprise domain, the edge point-of-presence of the network operator, and/or a centralized datacenter. Each critical component (PSC, PSCM) is mapped to a separate VNF, while PSAs are mapped onto VNF elements, the so-called VNFCs (VNF components).

There are mutual benefits in a relationship between SECURED and NFV. First, the NED faces a scalability problem as it may have to cope with hundreds of simultaneous users, but this is not an issue for NFV as new VNF can easily be deployed as needed. Second, since VNF is a technology being currently adopted by telecom and network providers, its mapping with SECURED implies an easy implementation path for those parties wishing to offer SECURED services. Last but not least, as SECURED pays special attention to the trust and security aspects of the NED, there are several techniques (such as remote attestation for distributed systems) that could be adopted to improve the NFV framework.

In addition to NFV, the adoption of "industry standard" components, such as OpenFlow (for networking) and KVM/OpenStack (for the computation part), enables our solution to be integrated in cloud-oriented platforms, hence guaranteeing synergies between different services of a network operator. Moreover, this allows a NED to offload part of its workload to other machines, such as servers operating in a datacenter, which can guarantee almost unlimited computational power in addition to cost savings (even when the NFV approach is not taken). Our architecture does not mandate the use of a single option, but leaves freedom to choose the most appropriate technology depending on the deployment scenario: a single NED may be appropriate for a home network or a small company infrastructure, while a cloud/NFV architecture may be used by a mobile operator to handle the network traffic of its customers.

Offloading applications to the network gives important advantages. In many cases, our approach ensures better performance in terms of responsiveness and throughput because of the limited resources available at the user terminal. Second, it saves resources at the user terminal, that may be dedicated to other purposes (entertainment, work) or to save power. Third, it provides *personal* security protection, independent from the physical terminal in use. Finally, our approach breaks the paradigm that the highest security standards are available only on high-end platforms: a user could have many and heavy applications operating on his traffic even if his terminal does not satisfy the technical requirements (e.g. CPU frequency, amount of memory) for those applications.

Among the costs that need to be paid for our solution, we mention the increased amount of time needed for connecting (securely) to the network, in addition to the overhead generated by exchanging additional data between the user terminal and the NED. For instance, the trusted channel between the user terminal and the NED, one of the key elements described in Section 4, requires either performing encryption/decryption of network packets at each channel side[3] and repeatedly fetching and evaluating the integrity measurements performed by the NED. It is worth noting that the above overhead does not apply in all scenarios; for example, the trusted channel can be avoided if the user trusts the network he is connected to (i.e. other entities are not considered as adversaries or the user is directly connected to the NED with a cable). In this case, the network performance would be the same as if applications are run at the user's terminal.

[3] This step could be avoided in case the access network already uses encryption, such as a WPA-protected WiFi hotspot.

Another possible drawback of our solution is the difficulty, for PSAs running in the NED, to access the information available inside the user terminal, such as the application that generated a given packet, in order to implement per-application security policies. While currently we are not addressing this issue, we are confident that a solution can be envisioned based on [3], which requires an additional software in the user terminal that monitors the traffic and transfers the <*network session ID - process ID*> pairs to a PSA running in the NED.

We think that the results of this preliminary evaluation are promising. The proposed architecture opens an interesting opportunity to offer user-centric protection (as opposed to the current device- and network-centric approaches) and enables also new business models, such as a marketplace for security applications (PSAs) and ISP contracts including PSA execution.

Acknowledgement

The research described in this paper is part of the SECURED project, co-funded by the European Commission (FP7 grant agreement no. 611458).

References

1. Armknecht, F., Gasmi, Y., Sadeghi, A.R., Stewin, P., Unger, M., Ramunno, G., Vernizzi, D.: An efficient implementation of trusted channels based on OpenSSL. In: ACM Workshop on Scalable Trusted Computing. pp. 41–50 (2008)
2. Goldman, K., Perez, R., Sailer, R.: Linking Remote Attestation to Secure Tunnel Endpoints. In: ACM Workshop on Scalable Trusted Computing. pp. 21–24 (2006)
3. Gringoli, F., Salgarelli, L., Dusi, M., Cascarano, N., Risso, F., Claffy, K.: Gt: Picking up the truth from the ground for internet traffic. ACM SIGCOMM Comput. Commun. Rev. 39(5), 12–18 (Oct 2009)
4. Trusted Computing Group: TPM Main Specification, Version 1.2, Revision 103. https://www.trustedcomputinggroup.org (2007)
5. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38(2), 69–74 (Mar 2008)
6. Network Functions Virtualisation Industry Specification Group (NFV ISG): Network Functions Virtualisation - update white paper (Oct 2013), http://portal.etsi.org/NFV/NFV_White_Paper2.pdf
7. Risso, F., Cerrato, I.: Customizing data-plane processing in edge routers. In: European Workshop on Software Defined Networks. pp. 114–120 (2012)
8. Sadeghi, A.R., Schulz, S.: Extending IPsec for efficient remote attestation. In: FC'10: Int. Conf. on Financial Cryptography and Data Security. pp. 150–165 (2010)