Virtualized Security at the Network Edge: A User-centric Approach

This is the author's version of an article published in the IEEE Communications magazine (April 2015, pp.176-186).
Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/MCOM.2015.7081092

Virtualized Security at the Network Edge: A User-centric Approach

D. Montero*, M. Yannuzzi*, A. Shaw†, L. Jacquin†, A. Pastor‡, R. Serral-Gracià*, A. Lioy§, F. Risso§, C. Basile§, R. Sassu§, M. Nemirovsky‡‡, F. Ciaccia¶, M. Georgiades‖, S. Charalambides‖ J. Kuusijärvi **, F. Bosco††
*Technical University of Catalonia (UPC), Spain
†Hewlett-Packard Laboratories, United Kingdom
‡Telefónica I+D, Spain
§Politecnico di Torino, Dip. Automatica e Informatica, Italy
¶Barcelona Supercomputing Center (BSC), Spain
‡‡ICREA Researcher Professor at BSC, Spain
‖PrimeTel PLC, Cyprus
**VTT Technical Research Centre of Finland Ltd, Finland
††United Nations Interregional Crime and justice Research Institute, Italy

***Abstract*—The current device-centric protection model against security threats has serious limitations. On the one hand, the proliferation of user terminals such as smart-phones, tablets, notebooks, smart TVs, game consoles and desktop computers makes it extremely difficult to achieve the same level of protection regardless of the device used. On the other hand, when various users share devices (e.g., parents and kids using the same devices at home), the set up of distinct security profiles, policies, and protection rules for the different users of a terminal is far from trivial. In light of this, this paper advocates for a paradigm shift in user protection. In our model, the protection is decoupled from the users' terminals, and it is provided by the access network through a Trusted Virtual Domain (TVD). Each TVD provides unified and homogeneous security for a single user, irrespective of the terminal employed. We describe a user-centric model, where non-technically savvy users can define their own profiles and protection rules in an intuitive way. We show that our model can harness from the virtualization power offered by next-generation access networks, especially, from Network Functions Virtualization (NFV) in the Points of Presence (POPs) at the edge of Telecom operators. We also analyze the distinctive features of our model, and the challenges faced based on the experience gained in the development of a proof-of-concept.**

***Index Terms*—Security, virtualization, offloading, NFV.**

I. Introduction

The protection of users' terminals against Internet threats is largely dominated by a device-centric model. This basically consists of installing a set of security applications on each terminal, such as anti-virus software, a personal firewall, etc. An average user nowadays has multiple terminals, including a smart-phone, a smart TV, and a notebook, and in many cases, also a tablet, a desktop computer and even a games console. These devices usually have different architectures (e.g., Intel or ARM) as well as different capabilities and operating systems (e.g., Android, Windows, or Linux), so the appropriate protection tools may not be available for all platforms. As a result, the most common practice is to install different security applications in the various terminals—or simply rely on the default protection means provided by the operating systems. Let us assume for a moment that users would like to have the same security policy and exactly the same protection level enforced on all of their devices. In the context of this paper, we will call this the "uniform security aim". To achieve this goal, the user typically needs to understand the configuration details of each device, which typically involves the setup of different security applications on different platforms. For non-technically savvy people, this turns out to be an impossible hurdle to overcome. As a result, most Internet users suffer from wide variations in their protection levels, and this problem is exacerbated as the number of devices per user grows.

In this paper, we propose a paradigm shift from device-centric protection to a user-centric model. The latter specifically addresses the two main drawbacks of the former, that is: i) the need for dissimilar installations of security applications in different devices due to their different platforms; and ii) the problem of non-uniform protection due to the difficulties in the configurations needed.

To cope with the first problem, we propose a model in which the protection and security policies are now unified and remain homogeneous for each user, independently of the terminal used. This is achieved by means of a user-specific Trusted Virtual Domain (TVD), which is dynamically instantiated at a secure place in the network edge. As we shall show, the TVD can be instantiated either on the user's side (e.g., on a home gateway), or on the provider's side (e.g., on a next-generation broadband access server handling the users' connections).

To cope with the second problem identified above, we propose a user-defined security model that aims at ease of use by design. We discuss the importance of exposing the selection of high-level protection policies to the average user, and the necessity to enforce the configurations required transparently to the latter. This simple strategy detaches the definition of the protection policies from their corresponding configurations, thus allowing tailored protection even by non-technically savvy users. It is worth highlighting that the virtualized security model described in this paper can be applied both to

residential and corporate scenarios. We describe its application in the form of a multi-tenant platform, considering the main stakeholders involved, i.e., service providers, infrastructure providers, security application developers, and the users.

The remainder of the paper is structured as follows. First, we outline the essentials of the paradigm proposed, including the new protection model and the security policy approach. Next, we introduce the general architecture and its main components. After that, we analyze the distinctive factors of our model, and outline some of the main conclusions that can be drawn from our prototype implementation. Finally, we conclude the paper.

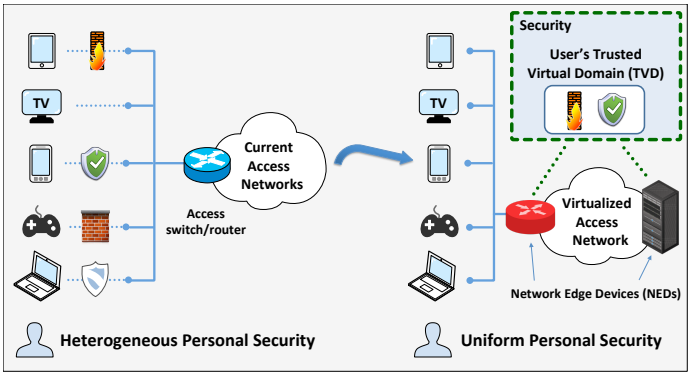
II. Toward a New Protection Paradigm

Figure 1a depicts the basic concepts, showing the evolution from device-specific security, to a common security framework for all devices hosted in the access network. In our model, security applications that are commonplace today, such as anti-virus, firewalls, content inspection tools, etc., shall be called Personal Security Applications (PSAs). Observe that, under the current protection model, the heterogeneity of devices and platforms requires the installation of various PSAs with similar roles and functions—actually four PSAs are required in the example shown on the left hand-side of Fig. 1a. Also observe that some devices may remain completely unprotected, such as the case of smart TVs.

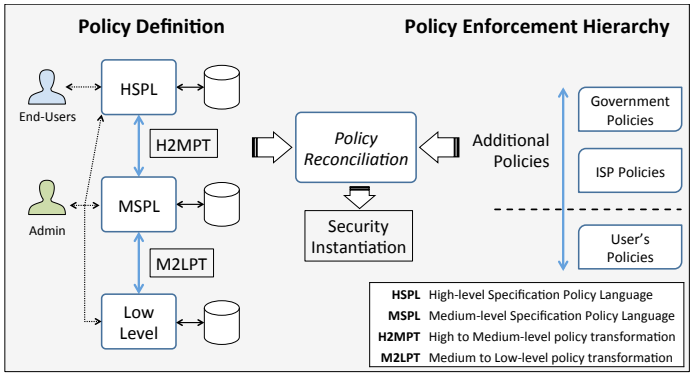
Under our paradigm, the heterogeneous set of PSAs protecting the different devices are now moved and consolidated into a TVD. Each TVD will only need to host the minimum set of complementary PSAs required by the user (e.g., an anti-virus and a firewall in the example). A TVD is a "logical container" that is instantiated *per-user*, and it is composed of the following elements: i) the execution environments hosting the user's PSAs; and ii) the required data, control, and management plane interconnectivity in order to guarantee the isolation between different users' TVDs—we will delve into this later in Section III (see Fig. 3). The right hand-side of Fig. 1a shows that a user TVD can be instantiated in either end of the access link. Indeed, as a logical container, a TVD may run entirely within a single Network Edge Device (NED), or it may run in a distributed way involving several NEDs. In our terminology, a NED is a device with virtualization capabilities that supports the instantiation of TVDs in a multi-tenant fashion. If the TVD is placed on the user's premises, then the NED could be either an enhanced home gateway or a Customer Premises Equipment (CPE). Those devices may need additional compute, storage, and networking resources, and they could be managed by the ISP. If the TVD is placed in the ISP premises—as it will be the case with the upcoming NFV-based access networks [1]—a pool of nodes belonging to the NFV infrastructure could be the NEDs devoted to host our TVDs. Note that this second deployment strategy leverages the virtualization and processing power of commodity hardware, and the unquestionable trend toward its ubiquity at the network edge—though it does not exclude the adoption of the first deployment strategy as well. It is worth highlighting that, our model has a remarkable advantage versus Cloud-based protection [2]. Whereas in the latter case the virtualized resources supporting the users' security are rarely on the path that would naturally be followed by the users' traffic, in our model, the TVD is always instantiated on the natural path. In other words, our model avoids routing detours, which would occur if the NEDs were located off the path between the user terminal and its traffic destinations (e.g., in the Cloud).

As its name indicates, the TVD must be trusted, since it will execute security applications on behalf of the user on one or more nodes that are typically owned and managed by a third-party. Appropriate techniques, such as remote attestations [3] or contractual agreements must be put in place, so as to guarantee the appropriate level of trust according to the security needs of a specific user. Also observe that the NEDs must be secure, since they will host the applications of several users that could potentially affect each other. As we shall show, the NED must be connected with a secure channel to the user terminal, because this path may be subject to attacks that could try to bypass the security controls performed at the NED.

Each PSA within a TVD implements one (or possibly more) security controls that need to be configured according to the needs of a specific user. However, the configuration of security applications is often complex and not well understood by the majority of the users. To simplify this task, we propose the model shown in Fig. 1b. The rationale behind it is that, to build a real user-centric model, it is mandatory to allow users to specify their own security requirements, i.e., their *security*



(a) Offloading security to the virtualized access network.



(b) Policy definition and the policy enforcement hierarchy.

Fig. 1. The two main objectives of our user-centric model, i.e., uniform protection and ease of configuration.

This is the author's version of an article published in the IEEE Communications magazine (April 2015, pp.176-186).
Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/MCOM.2015.7081092

policy, in a straightforward way. Our design principle aims to meet the expectations both of non-technically savvy users, and also of experts in the field, such as security administrators. For the former, the goal is to allow them to specify their security policy without needing to deal with the technicalities. For the latter, the goal is to allow them to fine tune their policies, while simplifying the configuration of the security applications under their administration.

To achieve these goals, our model is composed of three policy abstraction layers, and two translation services between them (see the left hand-side of Fig. 1b). The first abstraction layer is supplied by the High-level Security Policy Language (HSPL), a user-oriented authorization language suitable for expressing concepts related to users' protection. HSPL allows users to express general protection requirements by means of sentences that are very close to natural language, such as "do not permit access to war content", "block my son from accessing gambling sites", or "allow email scanning". In our model, HSPL policies can be selected from a set of candidate policies that can be then customized and grouped (e.g., "block access to gaming sites" + "only during week days"). The policy sentences are internally mapped to a *subject-verb-object-attribute* authorization language that is currently under definition as an XACML profile [4]. For instance, the policy "block my son from accessing gambling sites" is interpreted as "block" (verb) "my son" (subject) "from accessing gambling sites" (the object). Predefined lists of subjects, verbs and objects are made ready to the users, so they can easily compose their own sentences. Available attributes depend on the verb-object pair. Moreover, users can extend the predefined fields without being experts. The specific details of HSPL are out of the scope of this paper, so for additional information the reader is referred to [5].

The lowest layer in the policy abstraction stack is what we call the "Low Level" in Fig. 1b, as it is the one that deals with the configuration details of the PSAs. This configuration procedure is clearly application-specific, and hence is not under our control.

With the aim of abstracting the specific configuration procedures while meeting the experts' needs, we have created an intermediate abstraction layer that allows the specification of PSA configurations using a PSA-independent format. The security policies in this abstraction layer are specified by means of the Medium-level Security Policy Language (MSPL). The effort in the definition of the MSPL is not trivial. Indeed, depending on the heterogeneity of the different security control languages, the mappings can be arbitrarily complex. We address this complexity by means of an MSPL model that defines the main concepts (like policies, rules, conditions, and actions), and it is organized by *capabilities*. In this context, capabilities are defined as basic features that can be configured to enforce a security policy (e.g., channel protection, filtering, anti-virus, parental control, etc.). Our approach also allows to group families of languages with similar concepts (e.g., attributes, actions, or condition types), which can be captured by specific sub-models built by analyzing several languages of controls sharing the same capability. For instance, through MSPL, it is possible to write the configuration of a general packet filter, or to configure the options of a general anti-virus. An illustrative example of MSPL outlining the translation from HSPL up to Low Level configurations will be sketched later in Section II-A (cf. Fig. 2).

Overall, writing policies in MSPL demands the same security awareness and level of expertise as for specifying the configurations directly in the PSAs. The advantage, however, is that MSPL avoids experts the burden to master several semantically equivalent security controls and syntaxes. Observe that PSA developers will need to provide their plug-ins jointly with their PSAs, in the form of a Medium to Low-level Policy Translation service (M2LPT) (cf. Fig. 2). Also note that the complexity mainly resides in the language definition, so these translators fundamentally perform syntax adaptation. Thanks to this approach, a security policy written in MSPL can be embodied by different PSAs, provided that the candidate PSAs offer the capabilities required by the user. In addition, the PSAs can be replaced without impacting on the security policy specified by the user (e.g., replacing a Cisco packet filtering application by one provided by Checkpoint). For further details on MSPL, the reader is referred to [5].

As shown in Fig. 1b, the binding between HSPL and MSPL is supplied by the High to Medium-level Policy Translation service (H2MPT). Differently from the M2LPT translation, which is provided by the PSA developer, the H2MPT represents a translation service that is natively provided by our architecture. H2MPT uses formal ontologies to provide the semantics implied by the high-level policy statements. Our ontology is based on [6], and it models the high-level concepts (subjects, objects, verbs and attributes) as well as the medium-level concepts (rules, conditions, actions, resolution strategies), and the capabilities. The ontology also contains information on how predefined HSPL concepts are expanded into useful information for building MSPL rules. The translation process first identifies a set of applications that can enforce the security policies (e.g., a Web Filter and a Firewall), and then generates the MSPL for the selected applications. The HSPL verb-object pairs are used to match the capabilities needed for policy enforcement, while the capabilities per se are used to determine the PSAs and their interactions. Moreover, a *meta-model* defines how HSPL sentences are mapped into MSPL concepts, and how these concepts must be assembled to build valid rules. This meta-model is used by a set of Enrichment Modules and by a standard ontology reasoner, to gather all the information needed to create MSPL policies that enforce the HSPL policy [5], [6]. Finally, an H2MPT component combines this information into MSPL policies. This translation is done transparently for non-technically savvy users (i.e., for those users specifying their policies through HSPL). We contend that, by having a high-level policy specification language, our model provides far more flexibility and expressiveness than approaches based on profiles or templates. This is because these latter basically wrap under a common name a set of low level settings, which are basically applied for a fixed set of security controls.

In the model that we conceive, the PSAs can be selected by the users themselves or by a provider. If the user only specifies the HSPL, then the PSAs are automatically selected from a

This is the author's version of an article published in the IEEE Communications magazine (April 2015, pp.176-186).
Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/MCOM.2015.7081092

catalog of available applications, based on the PSAs that meet the functionality required by the policies. In our model, the capabilities of a PSA are specified through a "PSA manifest". In this context, the selection may be straightforward—when only one PSA is available with the required capabilities—or it may be based on various criteria if multiple PSAs could offer those capabilities (e.g., on the PSA reputation, its cost, etc.).

Another important aspect is that, according to recent studies, human mistakes are the major cause of breaches and vulnerabilities [7]. Thus, our model provides analytics that help reducing the likelihood of such mistakes. These include: a) contradictions among policies in different PSAs; b) policy contradictions within a PSA; or c) cases leading to suboptimal performance (e.g., rules that are never matched and that simply increase the processing time). Our model identifies this type of anomalies by means of state of the art techniques [8]. We represent clauses as hyper-rectangles, so that anomalies can be detected by using geometric intersections. Anomalies are classified by evaluating geometric relations among conditions (e.g., inclusion, intersecting conditions but no one includes the other), as well as relations between actions (e.g., same action, equivalent actions, conflicting actions). The resolution is dealt with formally modeled strategies, which cover a set of existing security control resolution mechanisms. Upon detection, we provide hints on how to resolve them, and notify the effects of each decision.

Moreover, the model that we envision should support multiple actors, which could simultaneously operate on the same traffic (see the right hand-side of Fig. 1b). Each of these actors may possibly impose its potentially conflicting security policy. For instance, a user can decide the level of protection that he needs, but the ISP may impose other limitations in order to guarantee the integrity of its network. In turn, the Government may impose additional restrictions. In case of conflict between the different policies in the hierarchy, our approach is to automatically resolve such anomalies, and inform the user about the issue and its outcome.

In order to resolve such conflicts, a "Reconciliation" [9] process is performed. The latter takes the policies of the different actors that must be reconciled, and obtains a single MSPL policy to be enforced by the user's PSAs. The core of this process is the resolution of contradictions among rules from different policies. Priorities and hierarchies are some of the simplest forms to resolve contradictions (i.e., rules from higher priority policies/actors prevail), and they typically map well to contractual frameworks. However, custom reconciliation strategies can be defined. The Reconciliation process copies non-conflicting rules in the reconciled policy, while each resolved contradiction generates a new rule. The latter have higher priority than the original ones, and the correct action is decided by the selected reconciliation strategy. More details on our reconciliation approach can be found in [10].

Observe that, actors may decide not to disclose their policies to other actors. In that case, reconciliation strategies that require full access to the policy set are not possible. An alternative approach is to use *policy chaining*. This consists of redirecting the output of one set of PSAs in an administration domain (e.g. the user PSAs) to a set of PSAs in another domain (e.g. the ISP PSAs). 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.

A. An Example of Policy Translation and Enforcement

To better describe our new paradigm, we present an example that illustrates the step-by-step process, starting from the definition of High Level policies up to the configurations made to guarantee their enforcement. Figure 2 depicts a simplified but complete example of the policy definition process for a non-technically savvy user. It is comprised of four basic steps. First, the user is requested to define its policies using the High Level Policy Language (HSPL). This user-oriented authorization language allows to express and customize a set of general security rules, by means of sentences that are very close to natural language, i.e., "*block phishing sites*".

Next, the HSPL policy sentences are mapped to a subject-verb-object-attribute authorization language aiming to extract the different security capabilities required by the user, (see step 2 on the figure). As a result, a service graph is built, where the nodes represent generic applications (PSAs) capable of fulfilling the security requirements. Observe that two applications are required in the example, i.e., Web Filtering and a Firewall. The selection of the PSAs is based on the manifest provided along with each PSA, which indicates its specific capabilities. Third, by using the ontology and the service graph information, the security policies are translated to MSPL, obtaining the application-independent definition of policies requested by the user (step 3 on the figure). The representation of MSPL policies will be stored and managed in XML format. Fourth, specific PSAs are selected satisfying the capabilities and requirements of the user. As mentioned above, the specific PSAs can selected either by the user or by the provider. For each PSA, the configurations are created by using an application specific translation plugin. These plugins convert the generic MSPL rules to application-specific configurations (see step 4). These configurations will be the inputs once the PSAs are instantiated and linked.

Finally, once the PSAs configurations are created, an orchestration system instantiates each PSA, and enforces its particular configuration, hence providing the security policies defined by the user.

III. The SECURED Architecture

This section introduces the envisioned architecture, which we call SECURED [5]. As explained in Section II, SECURED provides a system where users can offload their PSAs to their nearest compatible Network Edge Device (NED). The architecture is specifically devised to be heavily multi-tenanted, and flexible enough to be used in scale-out systems. From a use case point of view, it can be expanded and deployed in a variety of ways, ranging from small set-top boxes or home gateways, up to deployments at a much larger scale in a distributed environment (e.g., in localized datacenters at the edge of ISP's networks). Our focus in this section is on the main architectural components.

This is the author's version of an article published in the IEEE Communications magazine (April 2015, pp.176-186).
Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/MCOM.2015.7081092

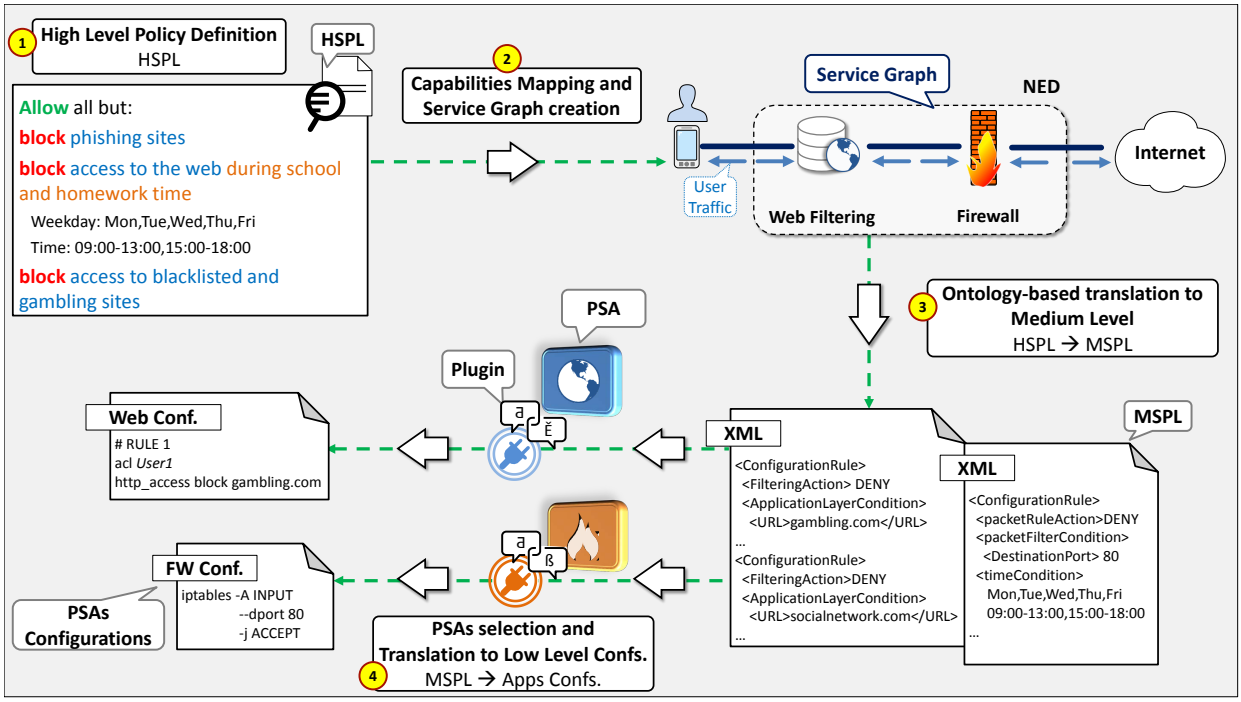


Fig. 2. Example of policy definition and enforcement, going from HSPL to MSPL and then to Low level configurations.

A. General Overview

The architecture must support the dynamic allocation and instantiation of users' security. The security functionality of each user can be comprised of different PSAs in a defined arrangement through service chaining, and these PSAs can be deployed within the same physical host or in a distributed manner. As a result, two general requirements are imposed on the architecture: i) massive multi-tenancy, which implies isolation of users, their applications, and network traffic; and ii) a secure and verifiable infrastructure and environment, which users can trust to host their security applications. A general view of the basic architecture is depicted in Fig. 3. The figure shows a generic deployment (e.g., on an NFV POP of an ISP). It is worth noting that, in simpler deployments (e.g., when the NED is a home gateway), the functionality provided by some of the systems on top of Fig. 3 could be simplified and embedded in the NED itself, or they could not be needed, such as the case of the NFV Orchestrator.

Overall, the first requirement is to guarantee complete isolation between different users. In light of this, the TVD was designed as an isolated environment which will hold the security applications of a user, and will in turn process the user traffic. A TVD is comprised of one or more Execution Environments (EEs). An EE is a lightweight and heavily controlled environment that contains and executes one or more user PSAs—each one operating on the principle of least privilege. Thus, within SECURED, two levels of isolation are defined (cf. Fig. 3): *i)* the *Compartmentalization Layer*, which is mainly responsible for the isolation between user TVDs; and *ii)* the *Containment Layer*, which handles isolation between PSAs within an EE. Thus, an EE could be either a *Compartment* or a *Containment* layer, respectively.

A derived requirement posed by multi-tenancy is network isolation. The SECURED architecture must ensure the isolation of traffic amongst different users. More precisely, each tenant will be configured with a dedicated and private virtual network. This network connects the different PSAs with the end user on one side, and the Internet on the other side. Furthermore, the architecture defines a private management network that sets up, controls and manages the different TVDs. Both the Compartmentalization and Containment Layers have a *Control and Management* component, which aims to establish a separation between the technology independent part and the implementation-dependent technology.

The second requirement is related to the establishment of trust between the end user and SECURED. This requirement is vital, since users would like to establish a certain level of trust with SECURED prior to requesting the instantiation of security applications and sending their traffic. We address this requirement by using the concept of *Remote Attestation* (RA). SECURED leverages trusted computing mechanisms to measure the system software upon component startup, where resulting measurement digests are held by a secure root of trust, e.g. a hardware device like a Trusted Platform Module (TPM) [11]. These measurements can be cryptographically signed by the device and sent to the users whenever they send an attestation request. The process of RA poses a major challenge for SECURED, and preliminary insight on a proof-of-concept implementation will be described later in Section IV-B.

B. Main components

Security Module: This module is the front end, which is contacted during the connection establishment. It is comprised of two elements, the *Attestation Agent* and the *Authentication*

This is the author's version of an article published in the IEEE Communications magazine (April 2015, pp.176-186).
Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/MCOM.2015.7081092

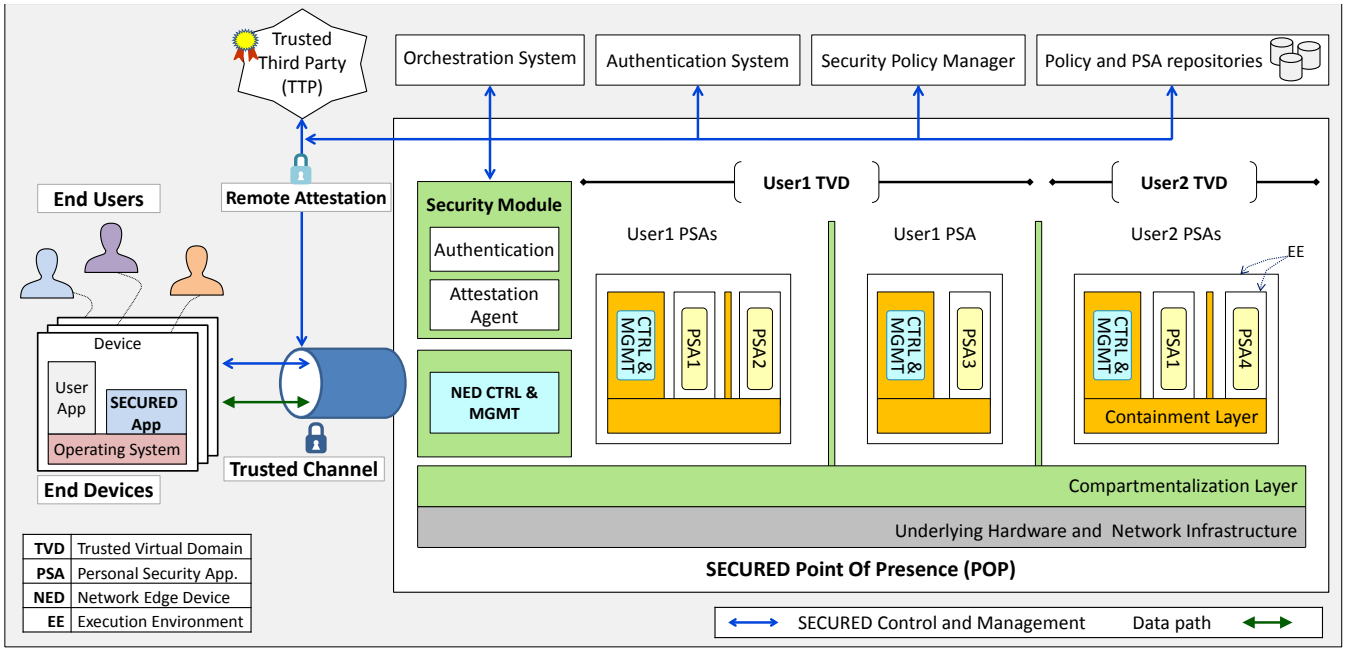


Fig. 3. The basic SECURED architecture showing a multi-tenant scheme on a Point of Presence (POP).

module. Prior to authenticating, the end user first contacts SECURED in an attempt to establish a secure connection whilst also performing the remote attestation protocol. To this end, the SECURED system receives a challenge request to perform an attestation of its software configuration. A mutually Trusted Third Party (TTP) system is involved in the attestation process. The TTP is responsible to keep a copy of known-good measurements, and provide a secure verification service to the user for verifying remote attestation responses. After a successful check, a secure channel is created and the user safely sends his credentials to the *Authentication module*.

Authentication System: The authentication of users is a key component of SECURED. This can be implemented either using a local (standalone) authentication system, or relying on an existing external authentication infrastructure (e.g., an AAA+ system). The result of the authentication process is to obtain tokens allowing the interplay between the main components within a NED, and external subsystems, such as the PSA repositories. Once the user is authenticated, the instantiation of his security must be enforced.

NED Control and Management: Once the user is authenticated, this module retrieves the user policies and metadata related to the composition of the required security applications. After that, the Control and Management module drives the instantiation of the user TVD, including its applications and the setup of the virtual network. More specifically, this module determines the resources required for the user TVD, and commands the instantiations required as well as the deployment and interconnection of the PSAs. This computation encompasses an analysis of the required compartments, containments and virtual networks to be allocated in order to instantiate the security applications. This analysis considers the PSA requirements along with the availability of resources, and the required configuration on the network (physical and virtual). In addition, this module also manages the extension of the user data path so as to connect the user's device to the newly created TVD.

Orchestration System: In the case of an NFV POP, the NED Control and Management module will be assisted by the NFV Orchestration system. However, in simpler scenarios, the former could entirely handle all the configurations required. In other words, when the NED is embodied in the home gateway of a residential user, the orchestrations needed will be handled locally without requiring any external orchestrator. In general terms, the Orchestration system should deal with the instantiations and configurations in large distributed systems (e.g., an NFV POP), and preferably, in a "technology-agnostic" way. The "technology-dependent" part could be managed by the Control and Management module embedded in the NED. In our model, the Attestation Agent keeps track of the different components during the instantiation phase (i.e., compartments, containments, and PSAs), and manages the corresponding measurements in order to present an attestation proof back to the user concerning his TVD.

Security Policy Manager: This module is in charge of handling the users' policies and the reconciliation process prior to performing the configuration of the user's PSAs.

PSA Repositories: The applications are retrieved from these repositories with their respective MSPL plugins, which then need to be loaded in one or more TVD containments.

SECURED App: This is the only application that needs to be installed in a user device. Its role is basically to support the secure communications with the NED, and handling the Remote Attestations and its outcomes.

Overall, the architecture introduced in this section allows the dynamic creation of trusted and virtualized execution environments throughout the access network. In this framework, several actors such as users, corporate ICT managers,

This is the author's version of an article published in the IEEE Communications magazine (April 2015, pp.176-186).
Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/MCOM.2015.7081092

infrastructure providers, security service providers, and software developers, can interplay and benefit from our user-centric protection model. An important remark about the proposed architecture is its alignment with the emerging NFV technology. NFV is an enabler for SECURED, and it will be essential for guaranteeing its scalability.

IV. Analysis of SECURED

The security model proposed in this paper has several distinctive factors that make it unique. To show this, we position SECURED in the current spectrum of protection techniques, and highlight its main differences with state-of-the-art solutions. In addition, we present and discuss our initial evaluations of a proof-of-concept implementation, with special focus on performance aspects related to the security, trust, and service verification offered by SECURED.

A. Positioning SECURED within the security panorama

The spectrum of solutions designed to counter security threats is really broad. The solutions available today can be reasonably categorized according to the table shown in Fig. 4. As it can be observed, there are solutions that are focused on protecting the end user device, while others propose different forms of security offloading. Moreover, current protection schemes can be classified based on whether they are user-centric, device-centric, network-centric, or corporate-centric. In a nutshell, Fig. 4 presents a high-level comparison of different security protection schemes according to two general criteria: i) the targeted protection model; and ii) where the security is enforced.

As far as our knowledge, SECURED is the only solution available nowadays that proposes a true user-centric model, which specifically addresses the need for device-independent security. As described in Section II, the user-centric approach is achieved thanks to the HSPL and MSPL languages, and the H2MPT and M2LPT translation services between the three abstraction layers involved. This allows users and even experts in the field, to focus on their security policies rather than on the configuration details of specific security applications. Another important aspect is that, conversely to many of the offloading solutions available today, which are typically deployed in the Cloud, our solution admits a rich variety of deployments on either edge of an access link. Cloud-based solutions provide compelling protection schemes while avoiding several of the overheads for end-users (e.g., for corporate customers). The downside, however, is that: a) they require routing detours; b) they are not really user-centric (at least not yet); c) they do not provide essential trust means, such as remote attestation; and d) they do not support advanced features such as anomaly verification and policy reconciliation techniques. These latter two are precisely a couple of distinctive aspects in SECURED, and therefore, are the center of our assessment and analysis at this stage. We proceed to provide insight about these aspects, and the challenges that we foresee based on a proof-of-concept implementation.

B. Remote Attestations

Trust establishment between an end user and the protection platform is a critical step towards security offloading. In our model, we use Remote Attestations (RAs) and verification techniques for the trust establishment process. Let us assume the following scenario: a user connects through an insecure channel and requests protection from SECURED. Prior to starting exchanging traffic, the user is requested to create a trusted channel toward a NED. A trusted channel is an instance of a secure channel (e.g., a VPN), where the endpoints are attested before any data exchange. In SECURED, the trusted channel protects users against a potentially compromised NED. However, enabling these security countermeasures introduces overhead. On the one hand, users may experience delay during the establishment of the connection with the NED. This is due to the integrity check needed, which is issued only once per user during the connection. On the

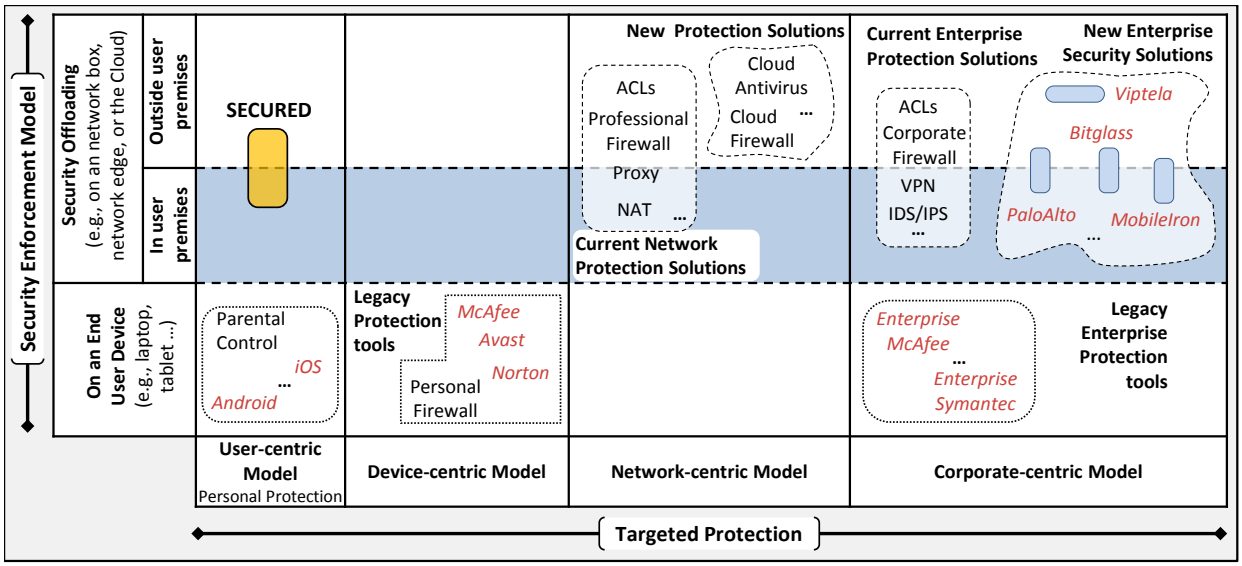


Fig. 4. Positioning SECURED considering some of the most common tools as well as some of the most recent and compelling solutions in the area.

This is the author's version of an article published in the IEEE Communications magazine (April 2015, pp.176-186).
Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/MCOM.2015.7081092

other hand, administrators may face scalability problems, since a portion of the network and the computational resources will be dedicated to the security checks as users connect. Normally, solutions offering this feature use a cryptographic chip—the Trusted Platform Module (TPM) [11]—that may pose a performance bottleneck while issuing the required verifications. SECURED overcomes this issue by introducing a Trusted Third Party (TTP) system (cf. Fig. 3). This is an entity that is trusted by users and infrastructure administrators, which asynchronously attests a set of controlled NEDs in a configurable time interval. The advantage of this approach is twofold. First, the workload for the attestation process does not increase with the number of connecting users, since the NED is common to all users. Second, end users will get a response regarding the integrity of the NED almost immediately.

We have developed a prototype that uses *strongSwan* [12] for the creation of a trusted channel with IPsec. To this end, strongSwan has been adapted to generate remote attestation requests to the TTP, and either continue or drop the connection depending on the result of the integrity verification. The TTP has been implemented with *OpenAttestation* [13], a framework for attesting large infrastructures. Our initial results show that the establishment of an IPsec connection without attestation is very fast (around 76 ms), and that the asynchronous attestation with the TTP in the same setting does not introduce noticeable delays (around 217 ms). Unlike our solution, performing synchronous remote attestation adds a significant delay on the creation time of the tunnel (around 4.119 seconds).

Another source of overhead is due to the size of the integrity reports. Figure 5 shows the size of the reports exchanged between the NED and the TTP. The results were obtained from a ten minute period, where a user repeatedly connected to the NED. While the first report generated is near 300 KB, subsequent reports are very small (between 4 and 8 KB), due to the fact that *OpenAttestation* sends only new integrity measurements—these are performed on the NED with the Integrity Measurement Architecture (IMA) [14] software. Note that the first report contains all the measurements performed at boot time. Furthermore, new reports will be generated only if new measurements are produced on the NED, i.e., when new software is executed.

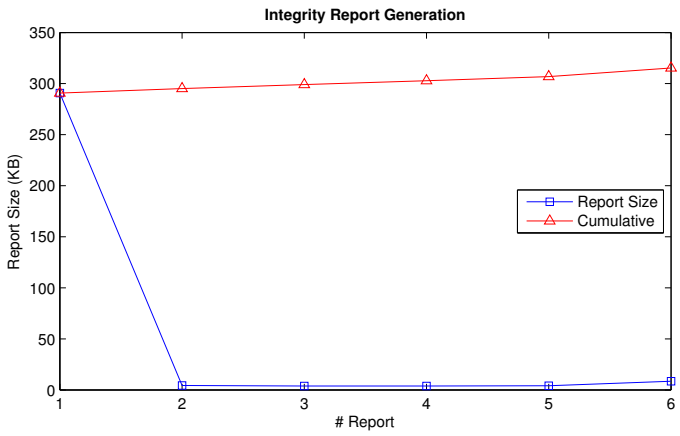


Fig. 5. Size of the integrity reports generated with OpenAttestation.

These initial results show that smartly performing RA does not incur in a noticeable overhead for the end user, as all the heavy lifting is asynchronously performed behind the scenes. The analysis also sheds light on the feasibility of enabling end users to remotely verify the status of a NED. It is worth highlighting that, the interval between two consecutive attestations can be configured, thereby offering the possibility of defining convenient trade-offs depending on the case. So far, we have seen that the RA of a single NED will introduce negligible overhead.

However, performing the RA over a distributed infrastructure poses complex challenges, and remains an open problem. These challenges increase when we also include in the picture multi-domain scenarios, or requirements such as user mobility and roaming. Furthermore, the assessment of time bounds for dynamic service deployment, as well as the appraisal of the multi-tenant isolation model will need to be deeply analyzed in the near future. We plan to develop a comprehensive prototype that will address these issues. Our research and future evaluations will prioritize the following aspects: a) security and isolation; b) easy-of-use; c) deployment and service provisioning in relatively short time scales; and, related to the latter, d) support for user mobility.

C. User-Centric Policy Framework

Our policy-based framework also needs an in-depth performance assessment to evaluate if the policy services can be actually used in real scenarios. To this purpose, we tested the performance of the reconciliation, anomaly analysis, and translation with an off-the-shelf computer equipped with an Intel processor i7-3630QM (2.4 GHz), with 16 GB of RAM, running OpenJDK RE 1.7.0 55 on top of a Linux operating system. We performed two different rule processing experiments: 1) average case with a realistic amount of rules; and 2) a higher bound worst case scenario with thousands of rules. In both cases, we have considered two types of filtering within the PSAs, namely, a *Packet Filter*, and an *L7 Filter*. During the experiments, we measured the time required to process and validate the filtering rules. As discussed in Section II, such validation is composed of three parts, anomaly analysis, reconciliation, and M2L translation.

The first tests evaluate the performance of a small/medium scenario, where the number of rules per user are on average in the range of tens or hundreds. This estimation was derived from a use case with four actors, where policies included 10 to 50 rules for each PSA, amounting to an average of 100 rules to be processed. We consider that these numbers per user are representative of a reasonable average, since in a user-centric approach, the size of the rule set will not raise to thousands—which is typically the case found on border firewalls of large companies. As reported in the first row of Table I, all the three measured policy-related tasks were completed in less than one millisecond.

The second experiment aims to assess the scalability on large scale policy scenarios. This means scenarios that, as stated on [8], statistically satisfy significant parameters of the policies that can be found in practice. This experiment provides two different results. On the one hand, we compute

This is the author's version of an article published in the IEEE Communications magazine (April 2015, pp.176-186).
Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/MCOM.2015.7081092

	Filtering level	Anomaly analysis	Reconciliation	M2L translation
Average case (time to process 100 rules)	Packet filter	<1 ms	<1 ms	<1 ms
	L7 filter	<1 ms	<1 ms	<1 ms
Worst case (time to process 5000 rules)	Packet filter	12 s	74 s	<1 s
	L7 filter	90 s	364 s	<1 s
Number of rules processed in 1 s	Packet filter	2000	1500	> 5000
	L7 filter	1000	1000	> 5000

TABLE I
RESULTS OF THE TESTS FOR POLICY-BASED TASKS.

the necessary processing time for a very large amount of rules. On the other hand, we compute the amount of rules that can be processed in 1 second—reasonable amount for interactive purposes. Both results are reported in Table I. We observe that, for the Anomaly Analysis, our prototype can process 5000 rules in 12 s for the *Packet Filtering* case. In contrast, *L7 Filtering* requires 90 s to perform the same task, due to the massive usage of regular expressions. In terms of the number of rules processed in less than a second, we obtained 2000 rules for the *Packet Filter* case, and 1000 for *L7 Filtering*. Regarding the reconciliation part, we were able to process 1500 *Packet Filter* policies and 1000 *L7 Filter* policies in less than a second. However, the worst cases for the 5000 rules considered yielded reconciliation times of 74 s, and 364 s, for the *Packet Filter*, and the *L7 Filter*, respectively. Finally, the translation of MSPL into low-level configurations is a linear problem that took approximately one second with 5000 rules both with an XSLT-based approach and with a SAX-based Java program. All these results are summarized in Table I.

Given that these computations are performed at infrastructure elements, wherein computational power can be adjusted as needed, we consider that our approach can reasonably scale in several real scenarios. For instance, the average cases are representative of residential scenarios, and all computations can be resolved online. We also consider that the processing of 5000 rules is quite representative of a corporate user case (e.g., an SME), and that the worst cases are highly unlikely that occur in practice. Anyway, the bounds found indicate that there are cases in which the reconciliation cannot be handled online, and therefore, this analysis serves as a starting point for investigating new strategies and optimizations.

V. CONCLUSION

In this paper, we have argued that for the large majority of Internet users, the current protection model against security threats is broken. Users typically have multiple devices, but achieving the same level of protection irrespective of the device used has become "mission impossible". We have proposed a paradigm shift in user protection, through a user-centric model that also decouples the security from the user terminals. The protection model that we envision is based on the setup of a Trusted Virtual Domain (TVD) per-user, placed in the access network. Our approach facilitates security policy configuration, and enables uniform protection independently of the terminal used. We have also shown that the trust and security verification mechanisms offered by a prototype implementation can be applied in many practical scenarios, such as the case of residential users.

Despite this, several of the issues addressed in this paper require significant efforts in terms of research. The list is large and includes aspects such as: remotely attesting distributed systems, multi-domain scenarios (i.e., the interplay among different ISPs), user mobility and roaming scenarios, scalability analysis, assessment of upper bounds for dynamic service deployment, isolation assessment, development of a comprehensive threat model, constraints and deeper analysis of corporate scenarios, and more.

ACKNOWLEDGMENT

The research described in this paper is part of the SECURED project [5], co-funded by the European Commission under the ICT theme of FP7 (grant agreement no. 611458).

REFERENCES

[1] ETSI, "Network Functions Virtualisation (NFV) Architectural Framework," http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf, December 2014.
[2] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2012, pp. 13–24.
[3] K. Goldman, R. Perez, and R. Sailer, "Linking Remote Attestation to Secure Tunnel Endpoints," in *Proceedings of the First ACM Workshop on Scalable Trusted Computing*, 2006, pp. 21–24.
[4] OASIS, "eXtensible Access Control Markup Language (XACML) Version 3.0," http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf, January 2013.
[5] "Security at the Network Edge (SECURED)," http://www.secured-fp7.eu/.
[6] C. Basile, A. Lioy, S. Scozzi, and M. Vallini, "Ontology-based Security Policy Translation," *Journal of Information Assurance and Security (JIAS)*, vol. 5, no. 1, pp. 437–445, 2010.
[7] IBM Global Technology Services, "IBM Security Services 2014 Cyber Security Intelligence Index," http://media.scmagazine.com/documents/82/ibm_cyber_security_intelligenc_20450.pdf, June 2014.
[8] C. Basile, A. Cappadonia, and A. Lioy, "Network-Level Access Control Policy Analysis and Transformation," *IEEE/ACM Transactions on Networking*, vol. 20, no. 4, pp. 985–998, 2012.
[9] P. McDaniel and A. Prakash, "Methods and Limitations of Security Policy Reconciliation," *ACM Transactions on Information and System Security*, vol. 9, no. 3, pp. 259–291, Aug. 2006.
[10] C. Basile, A. Lioy, C. Pitscheider, and S. Zhao, "A formal model of policy reconciliation." in *Proceedings of the 23th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2015)*, March 4–6, 2015.
[11] H. Zhang, Z. Qin, and Q. Yang, "Design and Implementation of the TPM chip J3210," in *Proceedings of the 2008 Third Asia-Pacific Trusted Infrastructure Technologies Conference (APTC '08)*, 2008, pp. 72–78.
[12] A. Steffen, "strongSwan - IPsec for Linux," https://www.strongswan.org.
[13] Intel, "OpenAttestation SDK: A SDK for Remote Attestation," https://github.com/OpenAttestation/OpenAttestation.
[14] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn, "Design and Implementation of a TCG-based Integrity Measurement Architecture," in *Proceedings of the 13th Conference on USENIX Security Symposium*, 2004, pp. 223–238.