Preprint of Paper at NOMS 2020

Implementing a Security Policy Management for 5G Customer Edge Nodes

Hammad Kabir, Muhammad Hassaan Bin Mohsin, Raimo Kantola
Department of Communications and Networking
Aalto University
Helsinki, Finland
{hammad.kabir, muhammad.mohsin, raimo.kantola}@aalto.fi

***Abstract*-- The upcoming 5th generation (5G) mobile networks need to support ultra-reliable communication for business and life-critical applications. To do that 5G must offer higher degree of reliability than the current Internet, where networks are often subjected to Internet attacks, such as denial of service (DoS) and unwanted traffic. Besides improving the mitigation of Internet attacks, we propose that ultra-reliable mobile networks should only carry the expected user traffic to achieve a predictable level of reliability under malicious activity. To accomplish this, we introduce device-oriented *communication security policies*. Mobile networks have classically introduced a policy architecture that includes Policy and Charging Control (PCC) functions in LTE. However, in state of the art, this policy architecture is limited to QoS policies for end devices only. In this paper, we present experimental implementation of a Security Policy Management (SPM) system that accounts communication security interests of end devices. The paper also briefly presents the overall security architecture, where the policies set for devices or services in a network slice providing ultra-reliability, are enforced by a network edge node (via SPM) to only admit the expected traffic, by default treating the rest as *unwanted* traffic.**

Keywords— Policy Management; unwanted traffic, reliability; network edge; communication security policy; 5G;

I. INTRODUCTION

All mobile network generations till 4G have been optimized for consumer markets only. However, the future 5th generation (5G) of mobile networks needs to serve both the consumer and corporate markets, including humans and machine-to-machine communications. Thus, the requirements of future networks are quite diverse compared to their predecessors and cannot be met with one-size-fits-all approach. Instead, networks need to be tailored to major use cases. For this purpose, 5G proposes to carry the traffic for major use cases in different network slices.

In 5G, ultra-reliable communications for life and business-critical applications is a new major use case. Since 5G also falls under the realm of Future Internet, it needs to overcome the drawbacks of the current Internet where service failures routinely take place due to unpredictable malicious activity. Thus, under the current Internet-style communications, it is not possible to achieve ultra-high reliability of services in 5G. We propose a security architecture, whereby the devices connected to a highly-reliable network slice or a highly secure network would only receive the traffic that is expected, and all the unexpected packets would be dropped by the network edge running the user's security agent. This implies that the network should have exact prior knowledge of the traffic that the device or service of a user is expecting. To achieve this, our security architecture relies on: (1) a network edge node that executes Customer Edge Switching (CES) [1] to run the user's security agent; and (2) a Security Policy Management (SPM) that allows the CES to admit or drop flows by provisioning the end-user's communication security policies. We believe that such a policy management is first employed to secure ultra-reliable communications, e.g. possibly to implement highly secure services for the Industrial Internet and firms, and can later be expanded to other 5G use cases.

SPM is in contrast with QoS-centric policy architecture of mobile networks, for example PCC in LTE, where user policies are typically a translation of the user's subscription with the mobile network operator. But, beyond subscription, a user has no control of its policies, and it is served by a mobile-operator firewall that applies the same security policy to all its users. The goal of the common firewall security is to protect the network rather than individual connected devices, and thus we deem it insufficient for ultra-reliability promises of 5G.

This paper presents an experimental implementation of our SPM that leads to a per device communication security policy in policy architecture of mobile networks, and thus contributes to ultra-reliability of services in 5G. Unlike state of the art, SPM allows policies to scale to individual services on a user device, making it possible to run different services on the same device with different policies. Moreover, it allows users to exert a restricted control of their policies. By enriching network with a host's communication security interests, SPM allows a highly-reliable network slice to offer security that is more *host centric* and better-than state-of-the-art. To serve a large volume of traffic and apply user policies, our network edge node relies on the control-plane and data-plane split architecture following the Software Defined Networking (SDN) principles. This allows allocating the required compute capacity from the cloud to process a large volume of traffic and to execute policies.

Our working hypothesis is that the security engine we are creating in CES can also be tailored to other use cases, besides ultra-reliable 5G communication. This is possible by modifying the structure of the policies and via correct policy sourcing. For example, in Industrial Internet, the policies can be created based on policy templates that are part of some contracting service. For smart phones and end-user devices, policies can be sourced via policy templates that are part of a utility program on the user device. For either case, we are targeting a common-core Security Policy Management. The first experimental system of SPM with the above goals in mind will be described in this paper.

978-1-7281-4973-8/20/$31.00

The rest of the paper is structured as follows. Section-II describes the related work. Section-III presents the overall security architecture based on CES and requirements for SPM. Section-IV discusses our implementation of the SPM. Section-V evaluates the SPM with respect to performance, scalability and its integration with CES. Section-VI discusses the use cases and relation to Network Neutrality regulation. Section-VII presents the discussion, and Section-VIII concludes.

II. Related Work

IETF introduced a simple yet extensible architecture, as a reference for systems needing policy-based management [2, 3]. The architecture comprises of four major components: Policy management tools, Policy repository, Policy Decision Point (PDP) and Policy Enforcement Points (PEPs). The policy management tool is generally an interface allowing formulation of policies, which are stored in the Policy repository. Upon receiving a message that requires policy-based processing, PEP sends a request to PDP, where the policy expertise resides. The PDP responds to policy request by contacting Policy repository and other servers. Upon its reception, PEP enforces the policy.

Depending on the desired control and functionality, several components have been added to this reference architecture, to provide policy-driven solutions for Network management [4], Security management [5], and for end-user QoS policies [6]. For example, CISCO security manager in [4] draws upon the reference IETF policy architecture to manage security on individual CISCO devices in a network, to ensure that a uniform security policy is deployed across the network. The solution, like [5], is entirely network oriented, and does not account for the security interests of end devices.

The PCC architecture in LTE is an example of network-managed QoS policies for end users. The architecture implements Policy and Charging Rules Function (PCRF) and Policy Control Enforcement Function (PCEF) to assume the roles of PDP and PEP. PCEF is implemented as a policy agent at the mobile edge node, and it is responsible for enforcing the policy rules and conveying related local events to PCRF. PCRF constructs a QoS rule by leveraging *user* and *session*-specific QoS parameters. The 3GPP specification in [6] shows that 5G too has so far only focused on QoS policies for end users.

The advent of SDN has drawn attention to its usage for policy-driven management in networks, especially for security purposes. The authors in [7] present an SDN-based security framework that allows a network operator to create security policies from a human-readable language. These policies are then enforced via a set of OpenFlow rules in network's data plane. However, the solution is entirely network oriented, and does not address communication security needs of end devices.

The authors in [8] proposed a policy-driven architecture allowing nodes with different authorization levels to engage in communication. The SDN controller sets up the rules so that traffic is authorized only when the originator has a higher authorization level than the receiver. However, the paper only deals with security levels of the hosts and does not use host policies as means of carrying the expected traffic only.

We note that a vast majority of policy-driven solutions for 5G are network oriented, whereby the aim is often to provision more capacity and scale resources to end user needs. The paper in [9], for instance, aims at optimization and orchestration of Virtual Network Functions (VNFs). However, these solutions generally overlook the communication security needs of end devices and services. We argue that in 5G, where new application contexts will continuously emerge, the true value of policy management in networks will come from accounting the interests of hosts/services that require the network to have a desired behavior. Thus, we stress that besides managing the network resources and user-centric QoS policies, as in state of the art, the networks shall also manage communication security interests of devices, at least in ultra-reliable 5G network slices.

For example, Industrial Internet (II) is one such use case of 5G, where end-devices can benefit from a more active security role of the network. The II Consortium presents a generic security framework in [10], stressing that protection mechanisms for end-devices can only be effective, if the risks associated to network connectivity are mitigated. For end-devices, besides other requirements, the framework underlines the need of asserting identities of the communicating parties and that every connection attempt in an II environment should be evaluated against the communication policy of end points, where policy violation shall lead to blocking of future interaction between parties. The framework however only describes security in general terms. In principle, the work presented in this paper realizes the principles laid for II, one of major 5G use cases, since SPM allows the networks to enforce the communication security interests of the end-devices.

When it comes to policy control of Internet services, in Europe and many other countries, the regulation on network neutrality (NN) must be followed [11, 12]. According to the regulation, by default, all traffic must be treated equally. However, there are certain exceptions to it, for example: to sell services of different quality at different prices to consumers; or for security filtering necessary to protect the network and end-user devices. We come back to NN in Section VII.

III. Proposed 5G Security Architecture

Fig. 1 presents the proposed 5G security architecture that relies on a Policy-frontend and SPM to account communication security interests of hosts and network, whereas a CES firewall at the network edge enforces these interests (via policies). We introduce the major components of this security architecture below.

A. Customer Edge Switching

A Customer Edge Switch is an enhanced NAT that resides between the Internet and Packet Data Gateway of a mobile network. A CES node acts as a connection broker for hosts in its network and is responsible for managing the identities and

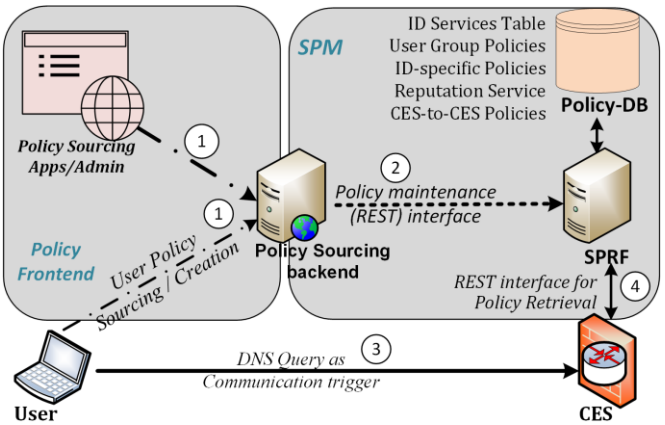

Fig. 1. Security Policy Management (SPM) in a CES network

addresses of all served hosts. It also translates between the two and offers assurances to remote network about the identities.

The CES design follows the control plane (CP) and data plane (DP) split architecture, following SDN principles. In our experimental CES implementation, the DP node comprises of; (a) an OpenvSwitch (OVS) [13] running on Linux; and (b) the *netfilter* functions in the Linux kernel [14]. The role of OVS is to mangle packets from one format to another, and to map addresses and protocol as instructed by the CP. Whereas, the Netfilter executes security rules related to network and hosts, to filter packets and flows in DP.

A CES-CP entity can control several DP nodes, and has one or more globally unique IP addresses to establish signaling channels with other CES nodes, via Customer Edge Traversal Protocol (CETP) [1] running over a socket interface. The scope of the CETP protocol is to negotiate the communication policies over the signaling channel between the CES nodes. The CETP signaling channel runs over IP/TCP/TLS and is structured into three sub-layers. The lowest layer manages a set of transport-layer connections to the remote CES node. The next layer manages the *CES-to-CES* relation and allows exchanging the network-specific policies and events, whereas the top layer is responsible for *host-to-host* or *host-to-service* policy negotiations. Upon success of a host policy negotiation, CES-CP inserts rules in the DP for the actual communication to commence via DP. A detailed first version of Customer Edge Switching (CES), and its policy negotiation is presented in [1].

B. Communication and Security policies in SPM

The CES security framework categorizes policies into *host* and *network*-specific (a) CETP-based communication policies; and (b) Firewall security policies, for a more elaborate control.

A security policy is concerned with filtering of packets in data plane, such that it results in *Accept* or *Drop* of a packet in DP. To offer security policies in a scalable manner, SPM allows network administrators to define policy groups, which among others specify a list of services permitted/denied to a subscribing user. An end point can belong to a policy group based on its subscription (or service contract), and can also specify its own security rules whereby it can further restrict the default group policy for itself, i.e. to disable a service allowed by group policy, or to enforce rate limitations etc. In CES, the format and strength of security policies is compliant to the Linux netfilter module.

A CETP policy consists of a set of policy elements that are exchanged (as *offers* and *requests*) between CES nodes serving the sender and the destination, i.e. to assert identities, assure compliance and hence guarantee that a node interacts with the intended entity only. The CETP policies are negotiated at CES-CP level, and are further classified into host-to-host (H2H) policy and CES-to-CES (C2C) policy. In a real network, a CETP-H2H policy would express communication interests of a host, while it is natural that a network administrator controls the network-level interactions by setting the CETP-C2C policy.

```
"request": [
{"ope":"query", "group": "id", "code": "fqdn", "value":
["hostb1.cesb.", "hostb1.foo."]},
{"ope":"query", "group": "control", "code": "ttl"},
{"ope":"query", "group": "payload", "code": "gre"} ],
"offer": [
{"ope":"info", "group": "payload", "code": "gre"},
{"ope":"info", "group": "id", "code": "fqdn", "value":"a1.cesa."},
{"ope":"info", "group": "control", "code": "ttl", "value":"30"}]
"available": [
{"ope":"info", "group": "control", "code": "ttl", "value":"30"},
{"ope":"info", "group": "payload", "code": "gre"},
{"ope":"info", "group": "id", "code": "fqdn", "value":"a1.cesa."},
{"ope":"info", "group": "id", "code": "hash", "value":"sdjsuefh"}]
```

Fig. 2 Sample CETP-based communication policy of host '*a1.cesa*'

A sample CETP-H2H policy of host *a1.cesa* is presented in Fig. 2. The policy consists of the *request*, *offer* and *available* vectors, each containing a set of policy elements that are either offered to or requested from the remote host's policy. An *offer* vector has policy elements that the CES node of the sender host will offer at start of the policy negotiation to a remote CES. An *offer* is a subset of the *available* policy vector, which has the list of all elements supported by a host's policy and is thus used to respond the policy *requests* from a remote host. A list of all the currently supported CETP policy elements is presented in [1], where the policy elements are further organized using the *group* and *code* fields.

The example policy of host *a1.cesa* in Fig. 2 aims to only setup connections with *hostb1.cesb* and *hostb1.foo* with a TTL= 30sec, such that data transfer in DP is via GRE tunneling. A similar format with network-related policy elements would represent a network's CETP-C2C policy. The complexity of translating the communication security interests to policies can be abstracted via Policy frontend, which presents appropriate policy sourcing tools and maintains backend towards SPM, as shown in Fig. 1.

C. End-user communication via 5G CES node

A CES node relies on a DNS query from its hosts to initiate connection towards a destination. For a communication to occur, both the CETP-C2C and CETP-H2H policy must be negotiated between the CES nodes serving the sender and the destination. The CETP policy negotiation at CES-CP enforces the *host* and *network* communication policies, and a successful policy negotiation at CP sets the rules for underlying exchange of packets in DP between the sender and the destination hosts.

D. Requirements for a Security Policy Management (SPM)

SPM plays a critical role in scaling the CES security to a large number of devices, services, and different use cases, by providing a platform where different stakeholders can manage their communication security interests. It primarily provides policy services to users and network administrators, allowing them to manage their respective *user*- and *network*-oriented communication security policies. Whereas, the CES leverages SPM to enforce these policies in the network edge. Based on our original motivation and related work, we have defined the following requirements and design principles for our SPM:

1) *The SPM architecture should draw upon the reference IETF policy architecture, for ease of its adoption in networks.*

2) *It should support policy tools for both the end users and network administrators, allowing them to manage their respective communication security interests in SPM.*

3) *SPM shall allow a network to offer policy services (to its user) in a scalable manner, by defining policy groups.*

4) *A host policy shall comprise of both immutable fields (set by network) and mutable fields (editable by users).*

5) *Users shall have a restricted control of their policies, such that a user can demote policy of its subscription, or could define its own communication policy for an allowed service.*

6) *It shall be possible to scale the policies to individual services on a user device, allowing multiple services on the same device to have different communication policies.*

7) *SPM shall maintain a policy hierarchy, to settle policy conflicts, for example in a user-subscriber relation, or when an administrator's policy must override a user's policy.*

8) *The CES security architecture (and SPM) shall rely on a stable device identifier, which limits the amount of changes in SPM, due to a change in device's networking/addressing.*

9) *SPM shall leverage the reputation of Internet entities to provision dynamic policies, such that a more strict policy can be executed towards an ill-behaving remote entity.*

The proposed security architecture does not aim to disrupt existing security and policy architectures in networks. Instead, it allows a network to meet communication security needs of hosts/services, by only carrying the traffic permitted by a host's policy. This in turn prevents the wastage of precious radio spectrum to unwanted traffic and contributes to availability of resources for legitimate uses, such as in II, safety, healthcare and other mission-critical use cases.

IV. Implementation of Security Policy Management

This section describes a detailed implementation of SPM, its major components, different phases from policy sourcing and creation to policy management, and policy enforcement via CES in the network edge.

A. Security Policy Rules Function (SPRF)

The SPRF node heavily corresponds in functionality to the services offered by PCRF in LTE, which constructs a rule by fetching the information from various data stores and system components. Fig. 3 presents the overall realization of SPRF, which consists of a REST server, a python-based Policy-API and a Database client. The REST server presents a standard REpresentational State Transfer (REST) [15] interface to serve policy queries towards SPM, where the use of REST over TLS allows authenticating the requesting entity. The Policy-API in our SPRF is a single point of entry to the Policy Database and performs the basic operations of storing, validating, retrieving and editing a policy in the Policy-database. Both the REST server and Policy-API are implemented in Python's *asyncio* framework [17], which allows asynchronous handling of the requests and responses. Thus, while an SPRF module awaits a response, SPRF can start handling another policy request.

The REST server and Policy-API in SPRF provide services to: (*1*) policy tools that source policies directly from a user device or from network administrators; and to (*2*) the *CES node* that enforces the policies in network edge. The SPRF provides policy services in response to the REST policy queries. Upon receiving a REST query, the REST server parses it to extract query parameters. For GET and DELETE policy requests, the query parameters are extracted from the Universal Resource Identifier (URI). For POST and PUT policy queries, typically initiated by a policy creation system, the query parameters are extracted from the HTTP payload. The extracted parameters allow the REST server to route the query to an appropriate Policy-API function. An HTTP 404 or an apt error code is returned in case of an invalid REST query.

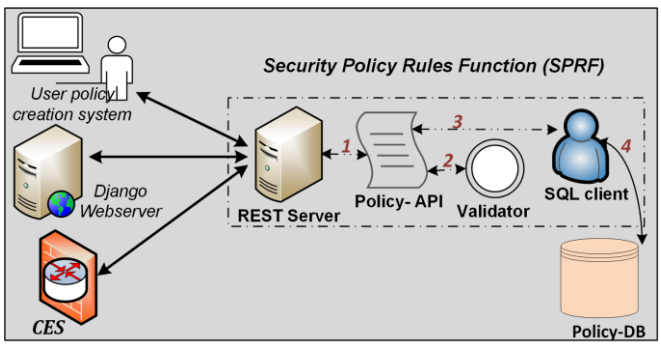

Fig. 3. Detailed implementation of SPRF

SPM validates the inbound query with a *Validator* module, which primarily focuses on POST and PUT requests to assure that only safe policies are inserted in the Policy database. Upon validation, the Policy-API executes an SQL query towards the Policy-DB. This either results in success or an error. In case of success, the response from Policy-DB is provided to Policy-API for further processing, i.e. to respond a GET policy request with a policy format executable at CES node.

B. Policy Management tools

A policy management tool authorizes an entity to manage its policies within SPM. An authorized entity could be a user; a subscriber; or an administrator that has a role in the final user policy. Besides presenting appropriate policy controls, a policy tool assures that only safe and validated policies are inserted in SPM, thus protecting it from malicious policy insertions. This section briefly presents our policy tools for: (a) end users; and (b) network administrators.

1) For end users*,* we have developed a policy creation and bootstrapping system in [16], which relies on the inputs from an Android mobile utility (on a user smartphone) that gathers the list of all the applications on the device, and presents the user an interface to set the communication security policies of the device or its services. The policies are then stored in SPM.

2) For domain administrators*,* we present an HTTP based policy management tool. The tool presents all the device and service policies in a network, and allows an administrator to override a device policy in favor of the network's policy. We choose Django [18] to implement the policy tool, due to its robust Model-View-Controller architecture. Here, the Model handles the database, Views provide frontend, and Controller contains the logic behind actions performed on HTML templates. Model holds the definitions of the Policy-database tables even though the tables do not actually reside in Django. This allows to produce database-centric forms on web pages and validate the policy inputs in run time. The policy tool uses Django services for user/admin authentication and for session handling. The login credentials determine authorization level of a logged-in user and allow it to add/edit the related policies.

The use-case specific policy tools make SPM scalable to various use cases. For example, a machine-dominant industrial network will expectedly only need administrator policy tools to manage the policies of its devices. Whereas, a corporate network serving humans will need policy tools for both the administrators and end users. Additional components for policy sourcing can be use case specific.

C. Policy Database

1) Policy control for Network administrators: The Policy-DB draws upon practices in mobile networks, where a user is

managed as per its subscription. A subscription plan (or a service contract) is typically defined as a *policy group* of QoS policies and permitted services that are assigned to a subscribing user. SPM extends this definition of a subscription plan, such that it also assigns a set of security policies to the subscriber. The security policies defined under a subscription plan allow administrators control over the type of traffic that is carried in the network, to/from a host.

In addition, Policy-DB can store network-oriented CETP-C2C policies, for the purpose of establishing trust and negotiating capabilities with remote networks. The one-time successful negotiation of CETP-C2C policy at the start allows the two CES networks to subsequently exchange the policies of their respective hosts/services, a pre-requisite to host-level connection across networks.

2) End-user policy control: In addition to the policies pre-defined in a subscription group, SPM allows users a restricted control of their policies. The principle of user's policy control is such that: (a) by default a user can only demote its policy, for example, a user security policy can disable a service that is allowed by its subscription. To utilize a service that is disabled by subscription, a user must make a new subscription or agree to operator's conditions. Upon approval, the policy change is noted in Policy-DB and is reflected in network at a scheduled time. In addition, (b) a user can set a communication policy on its device or service, so that it is only reachable to the intended entities. This latter is recorded as CETP-H2H policy in Policy-DB, and is executed via host-policy negotiation between CES nodes serving the sender and destination host/service.

Policy-DB also implements relation handling, for example between end user and administrator policies, to assure that an administrator's policy is always preferred over a user's policy, in case of a policy conflict.

3) Reputation of Internet entities: Our proof-of-concept Policy-DB also maintains a *Reputation Table* of Internet entities. In a real setup, the reputation can be dynamically populated by a trusted third-party providing the reputation services. Based on the provided reputation, it then becomes possible to dynamically enforce a lax or a strict policy with a remote host, service or a network.

4) User identity service: The user-ID record in Policy-DB can be critical to an identity registration service that provisions and records FQDN identities assigned to devices/services in a CES network. An FQDN ID is assigned to a user device on its joining of the network, and a CES network can leverage an identity registration process similar to captive web-portals [20] for this purpose. Upon success of a user's registration, the user-registration service and Policy-API (in SPM) exchange the authentication parameters, and a Fully Qualified Domain name (FQDN) is assigned to the IP of the user device. This FQDN is used to refer the user device in the CES network, and to load the policies from SPM. The authoritative DNS server of the CES network is also updated, to resolve DNS queries towards the newly allocated FQDN. Moreover, we introduced a more granular service FQDN (SFQDN) in [19] to address individual services on a device. An SFQDN is formed by prefixing the service name to the device FQDN.

An FQDN provides a stable host identity, independent of the device's local addressing within the CES network, whereas SFQDN allows policies to scale to individual services on a user device; making it possible to run and control different services on the same device with different communication policies.

Policy database internally maps the device identifiers to a 64-bit unique key, which is used as primary key within Policy-DB. This makes it possible to access the device/service policies in SPM via more than one (user) identity types, for example via FQDN or MSISDN of the user device.

D. Leveraging SPM policy services in CES node

Since SPM provides policy services over a REST interface, a CES node is equipped with a REST policy agent to leverage the policies from SPM. A CES node connects to SPM at its bootstrap and requests the *network* and *host* related security policies. These policies are then configured in a Netfilter-based firewall within DP, in a network and device specific manner. The security rule processing at DP results in: *Accept* or *Drop* of a packet, and a flow must meet the security policies of both the network and the user to be deemed as *accepted*. An accepted packet is carried to its destination, as per the flow states in DP e.g. provisioned by CES-CP or by the security policy itself.

In contrast, a CES node requests the CETP communication policies from SPM at the need of: (a) network-level or (b) host-level interactions. A CES node makes a REST policy query using FQDN of the host initiating the connection, whereas the inbound CES makes policy query using FQDN/SFQDN of the destination in the inbound CETP signaling message, to retrieve the CETP-H2H policy from SPM. To reduce the policy loading delay, the policy agent in CES can cache the frequently host policies. Prior to first host-level interaction across networks, a CES node requests CETP-C2C policy from SPM, in order to establish trust and capabilities with a remote CES network.

V. TESTING AND EVALUATION

We deployed our Security Policy Management (SPM) in a 64-bit Ubuntu-16 virtual machine (VM). The VM was hosted on a machine with Core i5-2400 quad core processors, and had two dedicated processing cores and a 2 GB of available RAM. This section evaluates the performance and scalability of our SPM implementation, and quantifies its security contribution by integrating it with CES function, in the network edge.

A. Performance

Due to the policy retrieval and storage needs, we expect the HTTP GET and POST as the frequent REST queries to SPM. Therefore, we benchmarked our system performance for these operations via WRK [21], an open-source HTTP benchmarking tool, and the test duration is limited to 30 seconds.

Table 1 presents the average delay perceived by CES for a REST policy query initiated towards SPM. The response time is further split into HTTP round-trip time (RTT) and policy-processing delay, where HTTP RTT is the delay to a REST query when no policy processing is involved. Table 2 shows that Policy-API processing results in a rate of nearly 500 REST queries served per second by SPM. From the results, it can also be seen that our Policy-API is designed such that it takes more time handling and validating a POST request and storing the corresponding policy in an efficient manner, so that a GET policy request for retrieving the same policy is responded quickly, to the performance-critical CES (network) nodes.

TABLE 1 - AVERAGE RESPONSE TIME OF SPM TO REST POLICY QUERIES

No. of Connections (Time: 30 sec)	Response Time per query in milliseconds		
	HTTP RTT	*Policy-API GET Processing*	*Policy-API POST processing*
1 Connection	0,41	1,86	19,82
5 Connections	1,86	6,80	28,58
10 Connections	3,6	11,98	37,78
50 Connections	18	59,22	81,15

TABLE 2 – RATE OF REST QUERIES SERVED BY SPM

No. of Connections (Time: 30 sec)	Requests per second to REST Server		
	Maximum rate (No Processing)	*GET Policy request rate*	*POST Policy request rate*
1 Connection	2488	481	166
5 Connections	2686	578	395
10 Connections	2779	632	410
50 Connections	2788	530	383

The asynchronous implementation of REST policy agent in CES allows it to initiate multiple simultaneous policy queries to SPM. However, the single-threaded Policy-API in SPM can only process one REST query at a time, causing a performance bottleneck. Thus, as the number of connections simultaneously initiating REST queries towards SPM increase in Table 1, the number of REST queries awaiting their processing at Policy-API rises, increasing the SPM response time. It is pertinent to mention that above tests have assumed GETting and POSTing of a single host policy per REST query.

B. *Scalability*

Table 3 presents the scalability evaluation of our SPM in supporting a large number of users, services and their policies. The standard REST interface allows SPM to support policy sourcing tools, whereas the use of service-FQDN allow SPM to scale policy services to individual services on the user device. In addition, SPM can account any number of stakeholders in deciding the final executable policy, by assigning a *priority* to policies defined by each stakeholder. In yielding the final policy, SPM overrides a low priority policy with a higher-priority policy. For this paper, we considered (a) users and (b) administrators as two stakeholders in deciding the user policy.

Table 4 shows the result of stressing the memory constraint of SPM, by storing the policies of a large number of hosts and services in Policy-DB. For example, for a network of 1 million devices, where each device has 10 applications, we assumed 1 security policy and 1 CETP-H2H policy per application. Under this assumption, Table-4 shows that the size of Policy-DB would be 3.2 GB. Given the storage capacity of today's computing platforms this is quite trivial. Thus, SPM can easily scale to large networks serving millions of devices.

Our testing does not reveal any major scalability challenges in SPM design. However, when leveraging policy services of SPM, the single-threaded Policy-API defines upper limit on the rate of policy queries served per second by *proof-of-concept* SPM implementation. To support an even larger rate of policy requests, SPM can leverage a multi-threaded Policy-API.

C. *Testing SPM policy services to CES*

In this section, we test the policy services provided by the SPM to CES nodes. For end hosts, a CES node typically requests security policies from SPM: *(a)* at its bootstrap; or *(b)* when a host joins a network, or updates its security policy. We measured that security policy loading delay (in order of few milli-seconds) is not performance critical in the long run, since

TABLE 3 – SCALABILITY EVALUATION OF SPM

Scalability testing	*Scalability Evaluation*	*Improvement scope*
User policies	Scalable	-
Service policies	Scalable	-
Stakeholders in policy definition	Scalable	-
Flood of REST policy queries	Limited scalability	Load-Balancer & Multi-threaded Policy-API

TABLE 4 – SCALABILITY OF CETP-H2H POLICIES

Users	*Basic Firewall policy size [+ 10 rules]*	*CETP-H2H policy size (x 10)*	*Policy-DB size*
1	1524 bytes	1900 Bytes	3.34 KB
100	149 KB	185.5 KB	334.5 KB
1K	1.48 MB	1.81 MB	3.29 MB
25K	37.1 MB	46.1 MB	83.2 MB
1M	1.45 GB	1.78 MB	3.23 GB

the security policies are not requested often. Thus, our testing focuses on the impact of SPM integration on more frequently requested CETP-H2H communication policies for hosts.

Our setup for SPM integration testing comprises of two virtual machines (VMs). The first VM implementing the SPM is already described, whereas the second VM contained two private networks served by their respective CES nodes. The CES nodes leveraged the policy from SPM. The network setup in the second VM is built using LXC containers [22], where each private network has hosts for initiating connection to other hosts/services. Upon an outbound communication initiated by a host or upon the need of accepting an inbound communication towards a host, the CES node serving the host makes a GET query for CETP-H2H policy of the host to SPM.

Fig. 4 reveals the delay incurred by introducing SPM in the CETP-policy negotiation, performed at CES-CP between two CES nodes. The figure reveals that when policies are locally cached at CES, the policy negotiation between two CES nodes completes in slightly less than 2 ms for 1 RTT of policy negotiation, and the corresponding user's DNS request that triggered policy negotiation is responded in 2.6 ms. However, upon loading the policies from SPM, the policy-negotiation between two CES nodes completes in 7 ms and the respective user DNS request is answered in 8 ms. The added delay of nearly 5 ms is due to policy retrieval over REST interface by both CES nodes, serving the sender and the destination. The policy retrieval from SPM contributes to an average delay of 2.5 ms at each CES node separately, in our testing.

Fig. 4. Impact of SPM integration on CETP-Policy negotiation at CES-CP

D. Communications security contribution of SPM

To reveal communication security contribution of CES and SPM, we conducted an experimentation that compares the level of unwanted traffic admitted towards a user device. Our experimental setup consists of three networks served by their respective edge nodes. The setup contains a total of 10 hosts, out of which Network-A and B respectively contain 7 and 2 hosts to initiate traffic towards a service, which runs in Network-C on port 8400 of its only host. We run the CES and NAT functions on the network edges, to compare and reveal the contribution of CES and SPM based security architecture.

For a realistic testing, we assigned a diverse set of profiles to hosts in Network-A and B. The profiles includes *(a) novice-hackers* that scan address (and port) space for possible attack venues; *(b) advanced-hackers* that target a precise IP (or port) binding on an edge node; *(c)* legitimate clients of a service; and *(d)* hosts that abuse Internet to initiate unwanted flows towards a destination. In our test setup, Network-A contains 1 host of each profile and 4 hosts of profile (d), whereas Network-B contains 1 host of each of first 2 profiles. In our testing, the hosts with hacker profiles originated 50 SYNs/sec flood towards Network-C, whereas the other host profiles initiated 2 connections/sec towards the service.

From a receiver perspective, we consider traffic stemming from all the sources, except (c) as *unwanted*, and expect that it should be filtered at the network edge, so that unwanted traffic is not received at a private host and thus it does not become a source of failure, unavailability or low-reliability of service.

When using NAT, a network edge can typically admit the inbound connections to its hosts, (a) either via static IP or port forwarding, or (b) via use of NAT traversal solutions that lead to a *complete* connection state in NAT. In our setup, Network-C employs different variants of these methods to carry traffic towards hosts in its network. Fig. 5 compares these solutions against the level of unwanted traffic they carry towards an end device. The figure shows that more loosely and static a NAT binding is defined, higher number of malicious and unwanted flows reach the user device. But, as the flow state in NAT gets specific, due to inclusion of sender-IP and destination-port, it becomes less likely for *novice-hackers* to abuse a connection state in NAT, and their flows are filtered. As a result, only the flows from advanced hackers or those arriving via NAT traversal methods would reach the service in Network-C.

If instead of static NAT bindings, Network-C employs a NAT traversal solution to admit the inbound connections, this would lead to complete dynamic connection states in NAT. As a result, NAT can filter flows from *hackers* that target a specific IP and port binding on network. However, it is still possible for a host to reach the service in Network-C, by using the service-ID and correct NAT traversal method. The 4th bar in Fig. 5 shows that in our experimentation, private hosts with profile (c) and (d) from Network-A could reach the service in Network-C, via a simulated NAT traversal solution. We label the traffic from hosts with profile (d) as *unwanted* and show it as such in the Fig. 5 (i.e. blue color in 4th bar).

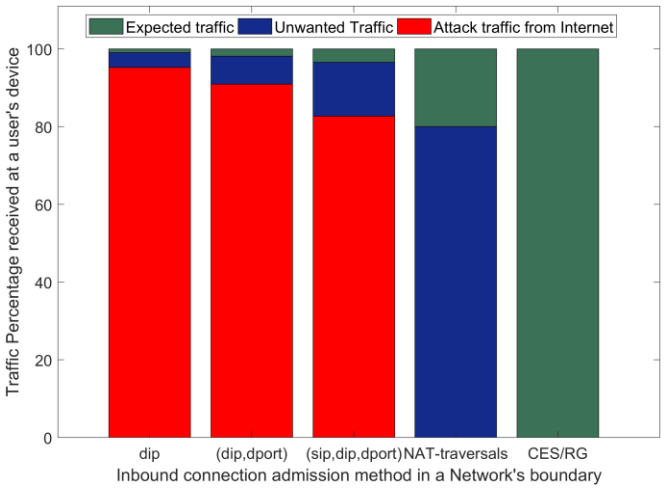

Fig. 5. Impact of CES and SPM deployment on unwanted Internet traffic

However, when network of both the sender and destination employ CES, our enhanced NAT that allows host-policy based communication, the service in Network-C only receives traffic from expected sources. This is because the communication security policies leveraged in CES via SPM leads to network-level filtering of flows from all the unwanted sources. We argue that atleast for ultra-reliable use cases, it is essential to filter unwanted and malicious flows in network, than to rely on host-based security for end devices. As it would otherwise exposes the ills of unwanted traffic not only to the service, but to the device that receives it, and the network that carries it, besides hogging resources of a legitimate user, i.e. DoS.

VI. Use Cases

We evaluate the fine-grained control over flow admission in the network against the network neutrality (NN) regulations in Europe. The European law on NN [11, 12] actively discourages or forbids the ISPs and mobile operators (MNO) to filter or block any traffic in their network, other than a short list of exceptions. One such exception is so called special services. An example in this category would be machine-to-machine communication in a smart grid control network, where devices under policy control would be 5G terminals at the endpoints of a differential line protection set up. Fiber or 5G would be used to carry power measurements to the other endpoint and when a short circuit is detected, the system would automatically trigger circuit breakers and thus cut power to the line with the short circuit and prevent the fault from spreading. In the system, the operator of a control center also needs to be able to read the status of the endpoints. The application is safety critical and does not deal with human communication, so even under NN regulation it makes sense to filter all the unexpected traffic in the network, from unexpected sources. If the network is not a traditional IP network, but SDN controlled, only expected flows will be set up and delivered under CES and SPM control.

A second use case for CES service and SPM could be an organization wary of espionage that uses smart phones as tools to access critical company systems and carry confidential information. Even under the European NN law, most likely, the company could subscribe its employees as mobile service users to an MNO. Due to *special* nature of communication, the smartphones would be connected to a virtual private network and the mobile network would route all traffic to and from the smartphone to a cloud service, where security would be added and access to the company services as well as to the Internet properly controlled. CES and policy control with a list of malicious and whitelist applications and hosts would be present in the cloud, to enforce the communication security policies. The company can choose the services of a security intelligence provider to maintain a portal of acceptable and malicious Apps and other control data in cloud, where different policy variants could apply to unknown Apps.

An intermediate use case would be a hospital with medical equipment, and medical personnel using the equipment connected to a 5G network. For safety and privacy reasons, it would be best if only the traffic from (safe applications of) an expected entity would be admitted to use the hospital network. A hospital network can ideally leverage CES and SPM policy control to secure Internet-connected medical devices/services.

VII. Discussion

We argue that achieving ultra-high reliability in 5G requires 1) overcoming the threats from classical Internet vulnerabilities such as, address spoofing, traffic floods and DDoS etc. that result in frequent service failures in the current Internet; and 2) that networks should offer a more user-centric security to carry only the flows expected by a user device, instead of deploying a network firewall that applies the same security to all its users. It seems that even under European law on NN, this is feasible for special services and for safety/security critical user groups.

In this regard, we have developed a network edge function namely CES, as an enhanced NAT and a cooperative network firewall that overcomes the classical Internet weaknesses such as address spoofing and DDoS by design [1]. In addition, CES allows negotiating policies of communicating hosts at network-level to filter the unwanted traffic, and to assure that only the traffic from policy-compliant sources is carried by the network. The host communication security policies (via SPM) allow a CES network to meet the host's interests. Our proof-of-concept SPM implementation can scale to support an even larger rate of policy queries per second (from CES), by leveraging a multi-threaded Policy-API implementation. Whereas, a stable release of HTTP/2.0 [23] can also contribute, as it supports parallel and asynchronous handling of REST queries by design.

VIII. Conclusion

The paper introduces a security architecture for devices and services in an ultra-reliable network, for example a 5G network slice. The security architecture relies on a policy architecture named SPM, to maintain the communication security policies of devices and services in a network, such that CES firewall at the network edge executes these policies, to only carry the traffic expected by an end host or its service. Our testing shows that this leads to network-level filtering of unwanted and malicious flows, in essence contributing to ultra-reliability and availability goals of 5G.

Unlike the most solutions in state of the art that are solely network oriented, SPM extends a part of the policy control to end hosts. This allows a network to offer more fine-grained and host-oriented security at network level. As a result, it becomes possible to filter the unwanted flows and attacks towards a host at the network, thus assuring availability of the precious radio spectrum and network resources to legitimate hosts and their needs. The CES security architecture relies on globally unique FQDNs as a stable host identity and leverages it to manage the host policies within SPM. A more granular service-FQDN allows CES and SPM to scale the policies to an individual service on the user device.

By presenting SPM, its experimental implementation, and evaluation with CES network function, this paper advocates the idea that all the flows in ultra-reliable networks should be carried only if permitted by both the network and host policy. The paper shows that this can be achieved by extending the policy architecture of the networks to carry the communication security policies of devices (and services), and via enhanced NAT namely CES at the network edge for policy enforcement. Due to SDN-compliant design of CES, the compute capacity to process a large number of flows and to apply policies can be provisioned on demand from a cloud platform, allowing CES and SPM based security architecture to scale to an even large number of hosts, services and edge nodes.

References

[1] Kantola, R., Llorente Santos, J., and Beijar, N. (2016) Policy-based communications for 5G mobile with customer edge switching. Security Comm. Networks, 9: 3070–3082. doi: 10.1002/sec.1253.

[2] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry and S. Waldbusser, "RFC:3198 Terminology for Policy-Based Management," 2001.

[3] G. Waters, J. Wheeler, A. Westerinen, L. Rafalow and R. Moore, "Policy Framework Architecture," Internet-draft, 1999.

[4] "Whitepaper: Cisco Security Manager: Upgrade to a 4.X Version for New Reporting, Monitoring, Analysis, and Other Features," CISCO, 2017.

[5] Choudhary, A. R. (2004), Policy-based network management. Bell Labs Tech. J., 9: 19–29. doi:10.1002/bltj.20002

[6] "5G System; Session Management Policy Control Service; Stage 3 - Release 15 - 29.512," 2018

[7] A. Lara, B. Ramamurthy, "OpenSec: Policy-based security using software-defined networking", IEEE Trans. Netw. Ser. Manage., vol. 13, no. 1, pp. 30-42, Mar. 2016.

[8] X. Liu and H. Xue, "Design of the multi-level security network switch system which restricts covert channel," in 3rd IEEE (ICCSN), 2011.

[9] C. Makaya, "Policy-based NFV Management and Orchestration" in IEEE NFV-SDN, San Francisco, CA, USA:, Nov. 2015.

[10] "Industrial Internet of Things Volume G4: Security Framework," Industrial Internet Consortium, 2016."

[11] "REGULATION (EU) 2015/2120 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL," *Official Journal of the European Union,* p. 18, 2015.

[12] "BEREC Guidelines on the Implementation by National Regulators of European Net Neutrality Rules," 2016.

[13] "Open Flow Switch Specification," Open Networking Foundation, 2012

[14] "Aalto5G/CustomerEdgeSwitching," 31 12 2019. [Online]. Available: https://github.com/Aalto5G/CustomerEdgeSwitching.

[15] R. T. Fielding, PHD dissertation: Architectural Styles and the Design of Network-based Software Architectures, University of California, 2000.

[16] I. F. Kalil, Policy Creation and Bootstrapping System, M.Sc. Thesis, Aalto University, Helsinki, Finland, Feb. 2018.

[17] "asyncio — Asynchronous I/O, event loop, coroutines and tasks," [Online]. Available: https://docs.python.org/3/library/asyncio.html. [Accessed 01 01 2018].

[18] "Django," [Online]. Available: https://www.djangoproject.com/. [Accessed 01 01 2018].

[19] J. Llorente and R. Kantola, "Transition to IPv6 with Realm Gateway 64," IEEE International Conference on Communications (ICC), London, June, 2015.

[20] "Captive Portal (Authenticated DHCP)," Infoblox.

[21] "Github - wrk2," [Online]. Available: https://github.com/giltene/wrk2. [Accessed 01.01.2018].

[22] "LXC," [Online]. Available: https://help.ubuntu.com/lts/serverguide/lxc.html. [Accessed 05 01 2018].

[23] M. Belshe, R. Peon and E. M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, Internet Engineering Task Force (IETF), May 2015, https://www.rfc-editor.org/info/rfc7540.