Interface to Network Security Functions (I2NSF) Problem Statement
draft-dunbar-i2nsf-problem-statement-05.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html
This document describes the motivation and the problem statement for Interface to Network Security Functions (I2NSF).

Table of Contents

1. Introduction...................................................3
2. Requirements Language..........................................4
3. Problem Space..................................................5
   3.1. Challenges Facing Security Service Providers..............5
       3.1.1. Diverse types of Security Functions...............5
       3.1.2. Diverse Interfaces to Control NSFs................6
       3.1.3. Diverse Interface to monitor the behavior of NSFs...7
       3.1.4. More Distributed NSFs and vNSFs....................7
       3.1.5. More Demand to Control NSFs Dynamically............7
       3.1.6. Demand for multi-tenancy to control and monitor NSFs.7
       3.1.7. Lack of Characterization of NSFs and Capability Exchange....................................................7
       3.1.8. Lack of mechanism for NSFs to utilize external profiles ..........................................................8
   3.2. Challenges Facing Customers...............................9
       3.2.1. NSFs from heterogeneous administrative domains......9
       3.2.2. Today’s Control Requests are Vendors Specific......9
1. Introduction

This document describes the motivation and the problem space for the Interface to Network Security Functions (I2NSF) effort.

The growing challenges and complexity in maintaining a secure infrastructure, complying with regulatory requirements, and controlling costs are enticing enterprises into consuming network security functions hosted by service providers. The hosted security service is especially attractive to small and medium size enterprises who suffer from a lack of security experts to continuously monitor, acquire new skills and propose immediate mitigations to ever increasing sets of security attacks.

According to [Gartner-2013], the demand for hosted (or cloud-based) security services is growing. Small and medium-sized businesses (SMBs) are increasingly adopting cloud-based security services to replace on-premises security tools, while larger enterprises are deploying a mix of traditional and cloud-based security services.
To meet the demand, more and more service providers are providing hosted security solutions to deliver cost-effective managed security services to enterprise customers. The hosted security services are primarily targeted at enterprises (especially small/medium ones), but could also be provided to any kind of mass-market customer.

As the result, the Network security functions (NSFs) are provided and consumed in increasingly diverse environments. Users of NSFs could consume network security services hosted by one or more providers, which may be their own enterprise, service providers, or a combination of both.

This document does not elaborate on specific use case. The reader should refer to [I2NSF-ACCESS], [I2NSF-DC] and [I2NSF-Mobile] for a more in-depth discussion on the I2NSF use cases.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

This document makes use of the following terms and acronyms:

DC: Data Center

Network Security Function (NSF): functions to ensure integrity, confidentiality and availability of network communications, to detect unwanted activity, and to block it or at least mitigate its effects on the network.

Hosted security function: Refers to a security function that it is hosted by another network.

Flow-based Network Security Function: A function that inspects network flows according to a policy intended for
enforcing security properties. Flow-based security also means that packets are inspected in the order they are received, and without modification to the packet due to the inspection process (MAC rewrites, TTL decrement action; even NAT would be outside the inspection process).

3. Problem Space

The following sub-sections describe the problems and challenges facing customers and security service providers (called service provider, for short) when security functions are no longer physically hosted by customer’s administrative domain.

The "Customer-Provider" relationship may be between any two parties: different firms or different domains of the same firm. Contractual agreements may be required in such contexts to formally document the customer’s security requirements and the provider’s guarantees to fulfill those requirements. Such agreements may detail protection levels, escalation procedure, alarms reporting, etc. There is currently no standard mechanism to capture those requirements.

Note a service provider may be a customer of another service provider.

3.1. Challenges Facing Security Service Providers

3.1.1. Diverse types of Security Functions

There are many types of NSFs. NSFs by different vendors can have different features and have different interfaces. NSFs can be deployed in multiple locations in a given network, and perhaps have different roles.

Below are a few examples of security functions and locations or contexts in which they are often deployed:

External Intrusion & Attack Protection:
   e.g., Firewall/ACL; Authentication; IPS; IDS; Endpoint Protection; etc;
Security Functions in a DMZ:
e.g., Firewall/ACL; IDS/IPS, authentication and authorization services, NAT, forward proxies, application FWs, AAA; etc.

Internal Security Analysis & report:
e.g., Security Log; Event Correlation; Forensic Analysis; etc;

Internal Data and Content Protection:
e.g., Encryption; Authorization; Public/Private key management for internal database, etc.

Given the diversity of security functions, contexts in which they can be deployed, and constant evolution of these functions, standardizing all aspects of security functions is challenging, most probably not feasible, and not necessary. For example, from an I2NSF perspective, there is no need to standardize on how a firewall filters are created or applied. What is needed is having an interface to control and monitor the behavior of NSFs.

3.1.2. Diverse Interfaces to Control NSFs

To provide effective and competitive solutions and services, Security Service Providers may need to utilize multiple security functions from various vendors to enforce the security policies desired by their customers.

Yet because no widely accepted industry standard security interfaces exist today, management of NSFs (device and policy provisioning, monitoring, etc.) tends to be bespoke, essentially as offered by product vendors. As a result, automation of such services, if it exists at all, is also bespoke. It is worth noting that even with the traditional way of deploying security features, there is still a gap to coordinate among implementations from distinct vendors. This is mainly the reason why mono-vendor security functions are enabled in a given network segment.
3.1.3. Diverse Interface to monitor the behavior of NSFs

Obviously, enabling a security function (e.g., firewall [I-D.ietf-opsawg-firewalls]) does not mean that a network is protected. As such, it is necessary to have a mechanism to monitor the execution status of NSFs.

3.1.4. More Distributed NSFs and vNSFs

The security functions that are invoked to enforce a security policy can be located in different equipment and network locations.

The European Telecommunications Standards Institute (ETSI) Network Function Virtualization (NFV) initiative creates new management challenges for security policies to be enforced by distributed, virtual, network security functions (vNSF).

vNSF has higher risk of failure, migrating, and state changes as their hosting VMs being created, moved, or decommissioned.

3.1.5. More Demand to Control NSFs Dynamically

In the advent of SDN [SDN-Security], more clients, applications or application controllers need to dynamically update their communication policies that are enforced by NSFs. The Security Service Providers have to dynamically update control requests to NSFs upon receiving the requests from their clients.

3.1.6. Demand for multi-tenancy to control and monitor NSFs.

Service providers may require having several operational units to control and monitor the NSFs, especially when NSFs become distributed and virtualized.

3.1.7. Lack of Characterization of NSFs and Capability Exchange

To offer effective security services, service providers need to activate various security functions manufactured by multiple
vendors. Even within one product category (e.g., firewall),
security functions provided by different vendors can have
different features and capabilities: filters that can be designed
and activated by a firewall may or may not support IPv6, depending
on the firewall technology, for example.

Service Provider management system (or controller) needs ways to
retrieve the capabilities of service functions by different
vendors so that it could build an effective security solution.

These capabilities can be documented in a static manner or via an
interface for security functions vendors to register to service
provider security management system. This dynamic capability
registration is useful for automation because security functions
may be subject to software and hardware updates. These updates may
have implications on the policies enforced by the NSFs.

Today, there is no standard method for vendors to describe the
capabilities of their security functions. Without a common
technical framework to describe the capabilities of security
functions, service providers can’t automate the process of
selecting NSFs by different vendors to accommodate customer’s
requirements.

3.1.8. Lack of mechanism for NSFs to utilize external profiles

Many security functions depend on signature files or profiles to
perform, e.g. IPS/IDS Signatures. Different policies might need
different signatures or profiles. Today, most vendors have their
vendor specific signatures or profiles. As the industry moves
towards more open environment, sharing profile or black database
can be win-win strategy for all parties involved. There might be
Open Source provided signature/profiles (e.g. by Snort or others)
in the future.

There is a need to have a standard envelop (i.e. the format) to
allow NSFs to use external profiles.
3.2. Challenges Facing Customers

When customers invoke hosted security services, their security policies may be enforced by a collection of security functions hosted in different domains. Customers may not have security skills. As such, they may not be able to express sufficiently precise requirements or security policies. Usually these customers express expectations (that can be viewed as loose security requirements). Customers may also express guidelines such as which critical communications are to be preserved during critical events, which hosts are to service even during severe security attacks, etc.

3.2.1. NSFs from heterogeneous administrative domains

Many medium and large enterprises have deployed various on-premises security functions which they want to continue to use. They are looking for combining local security functions with remote hosted security functions to achieve more efficient and immediate counter-measures to both Internet-originated attacks and enterprise network-originated attacks.

Some enterprises may only need the hosted security services for their remote branch offices where minimal security infrastructures/capabilities exist. The security solution can consist of NSFs on customer networks and NSFs on service provider networks.

3.2.2. Today’s Control Requests are Vendors Specific

Customers may consume NSFs by multiple service providers. Customers need to express their security requirements, guidelines, and expectations to the service providers, which in turn will be translated into security policies and associated configuration sets to the set of security functions. But no standard technical characterization and/or APIs exist, even for most common security functions.
services. Most security services are accessible only through disparate, proprietary interfaces (e.g., portals, APIs), in whatever format vendors choose to offer.

Without standard interfaces it is complex for customers to update security policies and integrate with services provided by the security service providers. This complexity is induced by the diversity of the configuration models, policy models, supported management interfaces, etc.

The current practices that rely on the use of scripts that generates automatically scripts have to be adjusted each time an implementation from a different vendor is enabled in a provider side.

Customers may also require means to easily update/modify their security requirements with immediate effect in the underlying involved NSFs.

While security agreements are in place, security functions may be solicited without requiring an explicit invocation means. Nevertheless, some explicit invocation means may be required to interact with a service function.

Here is an example of how standard interfaces could help achieve faster implementation time cycles. Let us consider a customer who would like to dynamically allow an encrypted flow with specific port, src/dst addresses or protocol type through the firewall/IPS to enable an encrypted video conferencing call only during the time of the call. With no commonly accepted interface in place, the customer would have to learn about the particular provider’s firewall/IPS interface, and send the request in the provider’s required format. If a firewall/IPS interface standard exists, the customer would be able to send the request, without having to do much preliminary legwork. Such a standard helps providers too since they could now offer the same firewall/IPS interface to represent firewall/IPS services, which may be offered by different vendors’ products. They have now abstracted the firewall/IPS services. Lastly, it helps the firewall/IPS vendors since they could now work on common specifications.
3.2.3. Difficulty to Monitor the Execution of Desired Policies

How a policy is translated into technology-specific actions is hidden from the customers. However, customers still need ways to monitor the delivered security service that is the result of the execution of their desired security requirements, guidelines and expectations.

Today, there is no standard way for customers to get security service assurance (including running "what-if" scenarios to assess the efficiency of the delivered security service) of their specified security policies properly enforced by the security functions in the provider domain.

3.3. Difficulty to Validate Policies across Multiple Domains

One key aspect of a hosted security service with security functions located at different premises is to have a standard interface to express, monitor and verify security policies that combine several distributed security functions. This becomes more crucial when NSFs are instantiated in Virtual Machines because NSFs can be more distributed and sometimes multiple NSFs are combined together to perform one task.

Without standard interfaces and security policy data models, the enforcement of a customer-driven security policy remains challenging because of the inherent complexity brought by the combined invocation of several, yet vendor-specific security functions, but also because of the accompanying complexity of configuration procedures and operational tasks in a multi-vendor, heterogeneous environment.

Ensuring the consistent enforcement of the policies at various domains is challenging. Standard data models are likely to contribute to softening that issue.
3.4. Lack of Standard Interface to Inject Feedback to NSF

Today, many security functions, such as IPS and Antivirus, depend heavily on the associated profiles. They can perform more effective protection if they have the up-to-date profiles. As more sophisticated threats arise, enterprises, vendors, and service providers have to rely on each other to achieve optimal protection. [CA] is one of those initiatives that aim at combining efforts conducted by multiple organizations.

Today there is no standard interface to exchange security profiles between organizations.

3.5. Lack of Standard Interface for Capability Negotiation

There could be situations when the NSFs selected can’t perform the policies from the Security Controller, due to resource constraints. To support the automatic control in the SDN-era, it is necessary to have a set of messages for proper negotiation between the Security Controller and the NSFs.

4. Scope of the proposed work

The primary goal of I2NSF is to define an information model, a set of software interfaces and data models for controlling and monitoring aspects of physical and virtual NSFs. Other aspects of NSFs, such as device or network provisioning and configuration, are out of scope. Controlling and monitoring of NSFs should include the ability to specify, query, monitor, and control the NSFs by one or more management entities. Since different security vendors support different features and functions on their devices, I2NSF will focus on flow-based NSFs that provide treatment to packets/flows, such as IPS/IDS, Web filtering, flow filtering, deep packet inspection, or pattern matching and remediation.

There are two layers of interfaces envisioned in the I2NSF approach:
- The I2NSF Capability Layer specifies how to control and monitor NSFs at a functional implementation level. That is, I2NSF will standardize a set of interfaces by which control and management of NSFs may be invoked, operated, and monitored. (I2NSF will not work on any other aspects of NSFs. Nor will I2NSF at this stage specify how to derive control and monitoring capabilities from higher level security policies for the Capability Layer.)

- The I2NSF Service Layer defines how clients’ security policies may be expressed and monitored. The Service Layer is out of scope for this phase of I2NSF’s work. However, I2NSF will provide a forum for Informational drafts on data models, APIs, etc. that demonstrate how service layer policies may be translated to Capability Layer functions.

The concrete work at the I2NSF Capability Layer includes development of

- An information model that defines concepts required for standardizing the control and monitoring of NSFs.
- A set of YANG data models, derived from the above information model.
- The capability registry (IANA) that enables the characteristics and behavior of NSFs to be specified using a vendor-neutral vocabulary without requiring the NSFs themselves to be standardized. The registry enables various mechanisms, including policy rules, to be used to match monitor and control functions to the needs of an application and/or environment.
- The proper secure communication channels to carry the controlling and monitoring information between the NSFs and their management entity (or entities).

Standard interfaces for monitoring and controlling the behavior of NSFs are essential building blocks for Security Service Providers to automate the use of different NSFs from multiple vendors by their Security management entities. This work will leverage the existing protocols and data models defined by I2RS, Netconf, and NETMOD.
I2NSF may be invoked by any (authorized) client—e.g., upstream applications (controllers), orchestration systems, security portals, etc.

5. Other Potential Uses of I2NSF

The I2NSF framework allows the clients to view, request, and/or verify the security functions/policies offered by providers at different premises. This framework can make it possible for a cluster of devices requiring the similar security policies to have consistent policies across multiple sites.

Network service providers can provide "Hosted Security Functions" services. Network providers can also act as security function brokers to facilitate if not optimize the enforcement of customer-driven security policies. They can expose a service catalog and standard mechanisms by which enterprises (or applications) can query, request, or/and verify the needed security functions or policies.

With the standard interfaces for clients to request the required security functions and policies, network operators can leverage their current service to enterprises (e.g. VPN, private IP services) and access to a vast population of end users to offer a set of consolidated Security solutions and policies. Network operators can be instrumental in defining a common interface and framework as part of an IETF-conducted specification effort.

6. Related Industry Initiatives

6.1. Related IETF WGs

IETF NETCONF: I2NSF should consider using the NETCONF protocol exchange security policy provisioning information between participating devices/security functions and the computation logic (a.k.a., a security Policy Decision Point (PDP)) that resides in the control plane and which makes the decisions to dynamically allocate resources and enforce customer-driven security policies.
NETMOD ACL Model: [I-D.ietf-netmod-acl-model] describes the very basic attributes for access control. I2NSF will extend the ACL data model to be more comprehensive, for example, extend to multiple actions and policies, and describes various services associated with the security functions under consideration.

In addition, I2NSF has to specify ways to monitor/report of Packet Based Security Functions.

I2RS: the WG currently discusses the specification of an interface between the forwarding and the control planes, to facilitate the dynamic enforcement of traffic forwarding policies based upon IGP/BGP route computation results. I2NSF is looking specifically into expressing security policies in two layers. I2NSF should leverage the protocols and data models developed by I2RS.

I2NSF aims to develop the additional information models and data models for distributed security functions, like the firewall and IPS/IDS. The policy structure specified by [I-D.hares-i2rs-bnp-info-model] can be used by I2NSF to be extended to include recursive actions to other security functions.

The IETF SFC WG specifies service function chaining techniques while treating service functions as a black box; VNFpool is about the reliability and availability of the virtualized network functions. But neither addresses how service functions are invoked, or configured.

Both SFC and VNFpool do not cover in-depth specification (e.g. rules for the requested FW) to invoke security functions. In SFC and VNFpool, a firewall function is a black box that is treated in the same way as a video optimization function. SFC and VNFpool do not cover the negotiation part, e.g. Client needs Rules x/y/z for FW, but the Provider can only offer x/z.

The IETF SACM (Security Assessment and Continuous Monitoring) WG specifies mechanisms to assess endpoint security. The endpoints can be routers, switches, clustered DB, or an installed piece of software. SACM is about "How to encode that policy in a manner where assessment can be automated". For example:
- a Solaris 10 SPARC or Windows 7 system used in an environment that requires adherence to a policy of Mission Critical Classified,
- rules like "The maximum password age must be 30 days" and "The minimum password age must be 1 day"

[I2NSF-GAP] has a more extensive study comparing I2NSF with various existing efforts in similar/adjacent areas.

6.2. Relationship with ETSI NFV ISG

ETSI’s NFV ISG defines the architecture to pool together many virtual network functions to be managed and consumed collectively.

I2NSF is one of the enabling tools for NFV, specifically the VNF as a Service (VNFaaS) specified by ETSI NFV Group Specification Use Cases [gs_NFV].

ETSI’s NFV ISG effort is actively contributed by service providers. It defines a detailed service model for VNFaaS as well as requirements that should be taken into account by the I2NSF initiative.

6.3. OpenStack Firewall/Security as a Service

Open source projects like OpenStack and CloudStack have begun to tackle the issues of interfaces to security functions but much work remains.

OpenStack completed the Firewall as a Service project and specified the set of APIs for Firewall services [API]

OpenStack has defined the APIs for managing Security Groups [SG]

The attributes defined by OpenStack Firewall/Security as a Service are at this point are basic. However, they can serve as the basis of the information model that the I2NSF IETF initiative aims to specify.
6.4. Security as a Service by Cloud Security Alliance

https://cloudsecurityalliance.org/research/secaas/#_get-involved

SaaS by CSA is at the initial stage of defining the scope of work.

7. Manageability Considerations

Management of NSFs usually include configuration of devices, signaling and policy provisioning. I2NSF will only focus on the policy provisioning part.

8. Security Considerations

Having a secure access to control and monitor NSFs is crucial for hosted security service. Therefore, proper secure communication channels have to be carefully specified for carrying the controlling and monitoring information between the NSFs and their management entity (or entities).

9. IANA Considerations

This document requires no IANA actions. RFC Editor: Please remove this section before publication.

10. References

10.1. Normative References


10.2. Informative References

[SG] http://docs.openstack.org/admin-guide-cloud/content/securitygroup_api_abstractions.html

[API] http://docs.openstack.org/admin-guide-cloud/content/fwaas_api_abstractions.html


11. Acknowledgments

Acknowledgments to Diego Lopez, Ed Lopez, Andy Malis, John Strassner, and many others for review and contribution to the content.

This document was prepared using 2-Word-v2.0.template.dot.
11.1. Appendix: Relationship with Open Source Communities

One of the goals of the I2NSF initiative is to form a collaborative loop from IETF to Industry Open Source Communities.
Open-source initiatives are not to be considered as an alternative to formal standardization processes. On the contrary, they are complementary, with the former acting as an enabler and accelerator of the latter. Open-source provides an ideal mechanism to quick prototyping and validating contending proposals, and demonstrating the feasibility of disruptive ideas that could otherwise not be considered. In this respect, open-source facilitates the engagement in the standardization process of small (and typically more dynamic) players such as start-ups and research groups, which would see better opportunities of being heard and a clearer rewards to their efforts. An open-source approach is extremely useful as well for the production of open reference implementations of the standards at the same (or even faster) pace they are defined. The availability of such reference implementations translate into much simpler interoperability and conformance assessments for both providers and users, and can become the basis for incremental differentiation of a common solution, thus allowing a cooperative competition ("cooperation") model.
Inter-Cloud DDoS Mitigation API
draft-fang-i2nsf-inter-cloud-ddos-mitigation-api-00

Abstract

This document defines an Inter-Cloud DDoS Mitigation Abstract Layer and corresponding standardized APIs to enable the exchange of real-time automated information to enable DDoS mitigation across Cloud Service Providers and Network Service Providers.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must...
include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ................................. 3
   1.1. Terminology ................................ 3
2. Problem Statement .......................... 4
3. Inter-Cloud DDoS Mitigation Layer ........ 5
4. Inter-Cloud DDoS Mitigation API .......... 6
5. Security Considerations .................. 7
6. IANA Considerations ....................... 7
7. References .................................. 7
   7.1 Normative References .................... 7
   7.2 Informative References .................. 7
Authors’ Addresses ............................. 7
1. Introduction

The recent growth in volume and scale of Distributed Denial of Service (DDoS) attacks, particularly its impact on the large pipes of Inter-Cloud, Inter-Provider connections, calls for mechanisms to enable DDoS mitigation across Cloud Service Providers (CSPs) and Network Service Providers (NSPs). These mechanisms require to define an Inter-Cloud DDoS Mitigation Abstract Layer with corresponding standardized APIs to allow real time, automated information exchange among CSPs and NSPs, and achieve rapid protective response and effective Inter Cloud/Inter Provider DDoS attack mitigation. The need for such standard Inter-Cloud DDoS Mitigation APIs is strong and urgent.

This document defines the Inter-Cloud DDoS Mitigation Abstract Layer and APIs.

This document focuses on Inter-Cloud, Inter-Provider automated exchange of DDoS Mitigation information, although similar APIs could be used within each cloud for handling malicious traffic.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This document uses the terminology defined in [I-D.draft-hares-i2nsf-use-case-gap-analysis].

In addition, this document uses the following terms.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP</td>
<td>Border Gateway Protocol</td>
</tr>
<tr>
<td>CSP</td>
<td>Cloud Service Provider</td>
</tr>
<tr>
<td>DC</td>
<td>Data Center</td>
</tr>
<tr>
<td>DCI</td>
<td>Data Center Interconnect</td>
</tr>
<tr>
<td>DDoS</td>
<td>Distributed Denial of Service</td>
</tr>
<tr>
<td>DLC</td>
<td>Disruption Life Cycle</td>
</tr>
<tr>
<td>Inter-Cloud</td>
<td>The interconnection between the cloud of different providers</td>
</tr>
<tr>
<td>NSP</td>
<td>Network Service Provider</td>
</tr>
<tr>
<td>SDN</td>
<td>Software Defined Network</td>
</tr>
<tr>
<td>SVR</td>
<td>Server</td>
</tr>
</tbody>
</table>

Fang et al. Expires <April 21, 2016> [Page 3]
2. Problem Statement

Along with the rapid growth of cloud services, the large pipes of Inter-Cloud, Inter-Provider connections are increasingly the subject of DDoS attacks. Since these connections are between clouds of different providers, implementing mechanism to achieve rapid protective response in case of attack is challenging. While within its own cloud each provider may be able to protect effectively its network using various DDoS protection techniques, for the Inter-Cloud/Inter-Provider links, each provider does not have full visibility of the attack, and therefore response times may be longer, counter-measures may be less effective, and therefore the severity and impact of the attacks may be very significant.

Large DDoS attacks targeting the Inter-Cloud, Inter-Provider links may consume the available bandwidth or the router/switch/server resources within tens of seconds. While the attack is on, legitimate traffic is prevented from being forwarded over the saturated links. With saturated Inter-Cloud, Inter-Provider links, even if within each cloud the DDoS mitigation may be working effectively, it can quickly be rendered irrelevant.

Today, exchange of DDoS attack information and mitigation strategy among providers is largely manual and typically relies on customized operation processes established ad hoc between each provider. Because of largely manual escalation procedures, providers’ reaction times to DDoS attacks to Inter-Cloud, Inter-Provider links tends to be slow (it can easily take tens of minutes if not hours to put effective mitigation measures in place) compared to Intra-Cloud DDoS mitigation, and thus the damage caused by such attacks can be substantial. The reaction time may exceed the Disruption Life Cycle (DLC) of the attack.

Sophisticated and determined malicious attackers are able to quickly learn the intended Inter-Cloud Inter-Provider link capabilities and limitations through probing. This includes bandwidth capacity, saturation resistance, and DDoS absorption resilience of the link. The attacker is also able to learn the DDoS countermeasures and their response times, from which the attacker can infer the DLC that can be exacted toward the intended target. The DLC is measured by the assailant from the time the attack is initiated to the time the mitigation response becomes evident. An attacker can then use this information to design the attacks in such a way that the current and subsequent attacks inflict the most harm.

In order to achieve rapid protective response, the exchange of DDoS mitigation information between providers must be enabled in real time and in an automated, standardized fashion.
3. Inter-Cloud DDoS Mitigation Layer

The Inter-Cloud DDoS Mitigation Layer and its corresponding standardized, secure Inter-Cloud DDoS Mitigation APIs is illustrated in Figure 1.

Today there is no accepted industry common DDoS Mitigation Layer that can be used to reduce the reaction time and increase the effectiveness of mitigation in case of attack.

The Inter-Cloud DDoS Mitigation Abstract Layer provides standardized secure APIs that can be used by each provider to programmatically initiate real time information exchanges to other providers to provide visibility of the attack and coordinate DDoS mitigation.
mechanisms, Exchanged information may include signatures and forensic of the attack, timestamps, and black-holing countermeasures.

The Inter-Cloud DDoS Mitigation Abstract Layer provides corresponding API calls to exchange mitigation information on the following areas.

DDoS Protection Types:

- TCP flood rate limiting
- UDP flood rate limiting
- TCP SYN.ACK/RST flood protection and authentication
- Maximum concurrent connections per interval rate limiting
- Maximum number of new connections allowed per interval rate limiting
- Maximum fragment packets allowed per interval rate limiting
- Maximum number of packets allowed per interval rate limiting
- Black-holing

BGP Signaling and Mitigation

- BGP /24 route advertisement with community string option
- Mitigation support for /32 with type and rate limit thresholds
- /32 removal from mitigation
- BGP support for /24 removal

Attack Lifecycle Monitoring and Reporting

- Volume and scale of the attack, signatures, forensic
- Timestamps

4. Inter-Cloud DDoS Mitigation API

TBD.
5. Security Considerations

TBD.

6. IANA Considerations

TBD.

7. References

7.1 Normative References


7.2 Informative References


Authors’ Addresses

Luyuan Fang
Microsoft
15590 NE 31st St
Redmond, WA 98052
Email: lufang@microsoft.com

Deepak Bansal
Microsoft
15590 NE 31st St
Redmond, WA 98052
Email: dbansal@microsoft.com
Analysis of Use Cases and Gaps in Technology for I2NSF
draft-hares-i2nsf-use-case-gap-analysis-00.txt

Abstract

This document provides a summary of the I2NSF use cases plus a summary of the state of the art in industries and IETF work which is relevant to the Interface to Network Security Function (I2NSF). The I2NSF focus is to define data models and interfaces in order to control and monitor the physical and virtual aspects of network security functions. The use cases are organized in two basic scenarios. In the access network scenario, mobile and residential users access NSF capabilities using their network service provider infrastructure. In the data center scenario customers manage NSFs hosted in the data center infrastructure.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 20, 2016.
Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
Provisions Relating to IETF Documents
(http://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1.  Introduction ................................................. 3
  1.1.  What is I2NSF ........................................... 4
  1.2.  I2NSF Standardization ................................. 4
  1.3.  Structure of the document ............................ 5
  2.  Requirements Language ................................... 6
  3.  Terminology .............................................. 6
  4.  Use Cases ................................................. 7
    4.1.  General Use Cases .................................... 7
    4.1.1.  Instantiation and Configuration of NSFs .......... 8
    4.1.2.  Updating of NSFs .................................. 8
    4.1.3.  Collecting the Status of NSFs .................... 9
    4.1.4.  Validation of NSFs ................................ 9
    4.2.  Access Networks ...................................... 9
    4.2.1.  vNSF Deployment .................................. 10
    4.2.2.  vNSF Customer Provisioning ....................... 10
    4.3.  Cloud Datacenter Scenario ......................... 10
    4.3.1.  On-Demand Virtual Firewall Deployment ........... 11
    4.3.2.  Firewall Policy Deployment Automation .......... 11
    4.4.  Considerations on Policy and Configuration ...... 12
    4.4.1.  Translating Policies into NSF Capabilities ...... 13
  5.  Gap Analysis ............................................. 14
    5.1.  Structure of the gap analysis ...................... 14
    5.2.  IETF Gap analysis .................................... 15
    5.2.1.  Traffic Filters ................................... 15
    5.3.  ETSI NFV ............................................ 22
    5.3.1.  ETSI Overview ................................... 22
    5.3.2.  I2NSF Gap Analysis ............................... 23
    5.4.  OPNFV ................................................ 24
    5.4.1.  OPNFV Moon Project ............................... 24
    5.4.2.  Gap Analysis for OPNFV Moon Project ........... 26
    5.5.  OpenStack Security Firewall ....................... 26

1. Introduction

Enterprise, residential, and mobile customers are becoming more and more aware of the need for network security, just to find that security services are hard to operate and become expensive in the case of reasonably sophisticated ones. This general trend has caused numerous operators and security vendors to start to leverage on cloud-based models to deliver security solutions. In particular, the methods around Network Function Virtualization (NFV) are meant to facilitate the elastic deployment of software images providing the network services, and require the management of various resources by customers, who may not own or physically host those network functions.

There are numerous benefits by defining such interfaces. Operators could provide more flexible and customized security services for specific users and this would provide more efficient and secure protection to each user.

This document provides an analysis of the use cases, gaps analysis of existing technology, recommendations for requirements for I2NSF, and security considerations.
1.1. What is I2NSF

A Network Security Function (NSF) is a function used to ensure integrity, confidentiality, or availability of network communications, to detect unwanted network activity, or to block or at least mitigate the effects of unwanted activity. NSFs are provided and consumed in increasingly diverse environments. Users could consume network security services enforced by NSFs hosted by one or more providers - which may be their own enterprise, service providers, or a combination of both. Similarly, service providers may offer their customers network security services that are enforced by multiple security products, functions from different vendors, or open source technologies. NSFs may be provided by physical and/or virtualized infrastructure. Without standard interfaces to control and monitor the behavior of NSFs, it has become virtually impossible for providers of security services to automate service offerings that utilize different security functions from multiple vendors.

1.2. I2NSF Standardization

The Interface to NSF devices (I2NSF) work proposes to standardize a set of software interfaces and data modules to control and monitor the physical and virtual NSFs. Since different security vendors support different features and functions, the I2NSF will focus on the flow-based NSFs that provide treatment to packets or flows such found
in IPS/IDS devices, web filtering devices, flow filtering devices, deep packet inspection devices, pattern matching inspection devices, and re-mediation devices.

There are two layers of interfaces envisioned in the I2NSF approach:

- The I2NSF Capability Layer specifies how to control and monitor NSFs at a functional implementation level. This is the focus for this phase of the I2NSF Work.
- The I2NSF Service Layer defines how the security policies of clients may be expressed and monitored.

For the I2NSF capability layer, the I2NSF work proposes an interoperable protocol that passes NSF provisioning rules and orchestration information between I2NSF client on a network manager and I2NSF agent on an NSF device. It is envisioned that clients of the I2NSF interfaces include management applications, service orchestration systems, network controllers, or user applications that may solicit network security resources.

The I2NSF work to define this protocol includes the following work:

- defining an informational model that defines the concepts for standardizing the control and monitoring of NSFs,
- defining a set of Yang data models from the information model that identifies the data that must be passed,
- creating a capability registry (an IANA registry) that identifies the characteristics and behaviours of NSFs in vendor-neutral vocabulary without requiring the NSFs to be standardized,
- examining existing secure communication mechanisms to identify the appropriate ones for carrying the data that provisions and monitors information between the NSFs and their management entity (or entities).

1.3. Structure of the document

This document reviews the terminology (section 3), analyzes the use cases (section 4) and gaps in current technology (section 5), recommends certain requirements for I2NSF protocol (section 6), and discusses security consideration (section 8).
2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. Terminology

- Network Security Function (NSF): A functional block within a network infrastructure to ensure integrity, confidentiality and availability of network communications, to detect unwanted activity, and to deter and block this unwanted activity or at least mitigate its effects on the network.

- vNSF: Virtual Network Security Function: A network security function that runs as a software image on a virtualized infrastructure, and can be requested by one domain but may be owned or managed by another domain.

- type of NSFs: NSFs considered in this draft include virtualized and non-virtualized NSFs.

- Cloud DC: A data center that is not on premises of enterprises, but has compute/storage resources that can be requested or purchased by the enterprises. The enterprise is actually getting a virtual data center. The Cloud Security Alliance (CSA) (http://cloudsecurityalliance.org) focus on adding security to this environment. A specific research topic is security as a service within the cloud data center.

- Cloud-based security functions: Network Security Function (NSF) hosted and managed by service providers or different administrative entity.

- DC: Data Center

- Domain: The term Domain in this draft has the following different connotations in different scenarios:
  * Client--Provider relationship, i.e. client requesting some network security functions from its provider;
* Domain A - Domain B relationship, i.e. one operator domain requesting some network security functions from another operator domain; or

* Applications -- Network relationship, i.e. an application (e.g. cluster of servers) requesting some functions from network, etc.

The domain context is important because it indicates the interactions the security is focused on.

- I2NSF agent - a piece of software in a device that implements a network security function which receives provisioning information and requests for operational data (monitoring data) across the I2NSF protocol from an I2NSF client.

- I2NSF client - A security client software that utilizes the I2NSF protocol to read, write or change the provisioning network security device via software interface using the I2NSF protocol (denoted as I2RS Agent)

- I2NSF Management System - I2NSF client operates within an network management system which serves as a collections and distribution point for security provisioning and filter data. This management system is denoted as I2NS management system in this document.

- Virtual Security Function: a security function that can be requested by one domain but may be owned or managed by another domain.

4. Use Cases

This section discusses general use cases, access use cases, and cloud use cases.

4.1. General Use Cases

User request security services through specific clients (a customer app, the NSP BSS/OSS or management platform...) and the appropriate NSF network entity will invoke the (v)NSF's according to the user service request. We will call this network entity the security controller. The interaction between the entities discussed above (client, security controller, NSF) is shown in the following diagram:
Interface 1 is used for receiving security requirements from client and translating them into commands that NSFs can understand and execute. Moreover, it is also responsible for giving feedback of the NSF security statistics to client. Interface 2 is used for interacting with NSFs according to commands, and collect status information about NSFs.

4.1.1. Instantiation and Configuration of NSFs

Client sends collected security requirements through Interface 1 to the security controller in the NSP network, which then translates them into a set of security functions. Then the corresponding NSFs are instantiated and configured through Interface 2.

As an example, consider an enterprise user A who wants to prevent a certain kind of traffic from flowing to their network. Such a requirement is sent from client to security controller through Interface 1. The security controller translates the requirement into a firewall function plus a rules for filtering out TCP and/or UDP data packets. Then it instantiates a firewall NSF through Interface 2. The corresponding filter rules are also configured onto this firewall NSF through Interface 2.

4.1.2. Updating of NSFs

A user can direct the client to require the update of security service functions, including adding/deleting a security service function and updating configurations of former security service function.

As an example, consider a user who has instantiated a security service before and decides to enable an additional IDS service. This requirement will be sent to the security controller through Interface 1 and be translated, so the security controller instantiates and configures an IDS NSF through Interface 2.
4.1.3. Collecting the Status of NSFs

When users want to get the executing status of security service, they can request the status statistics information of NSFs from the client. The security controller will collect NSF status statistics information through Interface 2, consolidate them, and give feedback to client through Interface 1. This interface can be used to collect not only individual service information, but also aggregated data suitable for tasks like infrastructure security assessment.

4.1.4. Validation of NSFs

Customers may require to validate NSF availability, provenance, and its correct execution. This validation process, especially relevant for vNSFs, includes at least

Integrity of the NSF. Ensure that the NSF is not manipulated.

Isolation. The execution of the NSF is self-contained for privacy requirements in multi-tenancy scenarios.

In order to achieve this the security controller has to collect security measurements and share them with an independent and trusted third party, allowing the user to attest the NSF by using Interface 1 and the information of the trusted third party.

4.2. Access Networks

This scenario describes use cases for users (enterprise user, network administrator, residential user...) that request and manage security services hosted in the network service provider (NSP) infrastructure. Given that NSP customers are essentially users of their access networks, the scenario is essentially associated with their characteristics, as well as with the use of vNSFs.

The Virtual CPE described in [NFVUC] use cases #5 and #7 cover the model of virtualization for mobile and residential access, where the operator may offload security services from the customer local environment (or even the terminal) to the operator infrastructure supporting the access network.

These use cases defines the operator interaction with vNSFs through automated interfaces, typically by B2B communications performed by the operator management systems (OSS/BSS).
4.2.1. vNSF Deployment

The deployment process consists of instantiating a NSF on a Virtualization Infrastructure (NFVI), within the NSP administrative domain(s) or with other external domain(s). This is a required step before a customer can subscribe to a security service supported in the vNSF.

4.2.2. vNSF Customer Provisioning

Once a vNSF is deployed, any customer can subscribe to it. The provisioning lifecycle includes:

- Customer enrollment and cancellation of the subscription to a vNSF.
- Configuration of the vNSF, based on specific configurations, or derived from common security policies defined by the NSP.
- Retrieve and list of the vNSF functionalities, extracted from a manifest or a descriptor. The NSP management systems can demand this information to offer detailed information through the commercial channels to the customer.

4.3. Cloud Datacenter Scenario

In a datacenter, network security mechanisms such as firewalls may need to be added or removed dynamically for a number of reasons. It may be explicitly requested by the user, or triggered by a pre-agreed-upon service level agreement (SLA) between the user and the provider of the service. For example, the service provider may be required to add more firewall capacity within a set timeframe whenever the bandwidth utilization hits a certain threshold for a specified period. This capacity expansion could result in adding new instances of firewalls. Likewise, a service provider may need to provision a new firewall instance in a completely new environment due to a new requirement.

The on-demand, dynamic nature of deployment essentially requires that the network security "devices" be in software or virtual form factors, rather than in a physical appliance form. (This is a provider-side concern. Users of the firewall service are agnostic, as they should, as to whether or not the firewall service is run on a VM or any other form factor. Indeed, they may not even be aware that their traffic traverses firewalls.)

Furthermore, new firewall instances need to be placed in the "right zone" (domain). The issue applies not only to multi-tenant
environments where getting the tenant right is of paramount importance but also to environments owned and operated by a single organization with its own service segregation policies. For example, an enterprise may mandate that firewalls serving Internet traffic and business-to-business (B2B) traffic be separate; or that IPS/IDS services for investment banking and non-banking traffic be separate for regulatory reasons.

4.3.1. On-Demand Virtual Firewall Deployment

A service provider operated cloud data center could serve tens of thousands of clients. Clients’ compute servers are typically hosted on virtual machines (VMs), which could be deployed across different server racks located in different parts of the data center. It is often not technically and/or financially feasible to deploy dedicated physical firewalls to suit each client’s myriad security policy requirements. What is needed is the ability to dynamically deploy virtual firewalls for each client’s set of servers based on established security policies and underlying network topologies.

---+-----------------------------+-----
|                             |     |
|                             |     |
+---+                         +-+-+
|vFW|                         |vFW|
+---+                         +-+-+
|    Client #1                |  Client #2
---+-------+---               ---+-------+---
+-+-+   +-+-+                 +-+-+   +-+-+
|vM |   |vM |                 |vM |   |vM |
+---+   +---+                 +---+   +---+

Figure 3: NSF in DataCenter

4.3.2. Firewall Policy Deployment Automation

Firewall configuration today is a highly complex process that involves consulting established security policies, translating those policies into firewall rules, further translating those rules into vendor-specific configuration sets, identifying all the firewalls, and pushing configurations to those firewalls.

This is often a time consuming, complex and error-prone process even within a single organization/enterprise framework. It becomes far more complex in provider-owned cloud networks that serve myriad customers.
Automation can help address many of these issues. Automation works best when it can leverage a common set of standards that will work across multiple entities.

4.3.2.1. Client-Specific Security Policy in Cloud VPNs

Clients of service provider operated cloud data centers need not only secure virtual private networks (VPNs) but also virtual security functions that enforce the clients’ security policies. The security policies may govern communications within the clients’ own virtual networks and those with external networks. For example, VPN service providers may need to provide firewall and other security services to their VPN clients. Today, it is generally not possible for clients to dynamically view, much less change, what, where and how security policies are implemented on their provider-operated clouds. Indeed, no standards-based framework that allows clients to retrieve/manage security policies in a consistent manner across different providers exists.

4.4. Considerations on Policy and Configuration

NSF configurations can vary from simple rules (i.e. block a DDoS attack) to very complex configuration (i.e. define a user firewall rules per application, protocol, source and destination port and address). The possibility of using configuration templates per control and management type is a common option as well.

A NSP can push security policies using complex configurations in their managed vNSF through its management system. The open Control and management interface has to accommodate this application-driven behavior.

Computer-savvy customers may pursue a similar application-driven configuration through the open Control and management interface, but standard residential and mobile customers may prefer to use the definition of security policies in the form of close-to-natural-language sentences with high-level directives or a guide configuration process. The representation for these policies will be of the form:

```
+-------+   +------+   +------+   +------------------+
|Subject| + |Action| + |Object| + |Field_type = Value|
+-------+   +------+   +------+   +------------------+
```

Figure 4: High-Level Security Policy Format

Subject indicates the customer or device in the access.
Action can include a variety of intent-based actions: check, redirect, allow, block, record, inspect.

Object can be optional and specifies the nature of the action. The default is all the customer traffic, but others possible values are connections and connections attempts.

Field_type allows to create fine-grained policies, including destinations list (i.e. IPs, domains), content types (i.e. files, emails), windows of time (i.e. weekend), protocol or network service (i.e. HTTP).

An example of a customer policy is:

"My son is allowed to access Facebook from 18:30 to 20:00"

4.4.1. Translating Policies into NSF Capabilities

Policies expressed in the above model are suitable for what we depicted as Interface 1 in Figure 2. In order to allow the security controller to deal with the different NSFs an intermediate representation used for expressing specific configurations in a device-independent format is required. For this purpose, the definition of a set of security capabilities provides a means for categorizing the actions performed by network security functions. An initial, high-level set of such capabilities consists of:

- Identity Management: Includes all services related with identity, authentication and key management. Some examples are:
  * AAA (Authentication, Authorization, Accounting) services
  * Remote identity management
  * Remote identity management

- Traffic Inspection: A common use case for customers accessing the Internet or additional services through it is security supervision. Control and Management interfaces will allow the configuration of the vNSF inspection features: signatures updates, behavioral parameters or type of traffic to supervise. Some examples are:
  * IDS/IPS (Intrusion Detection System/Intrusion Prevention System,
  * Deep packet inspection,
* Data leakage protection,

  o Traffic Manipulation: A more intrusive use case of NSF includes the capacity of manipulate the client traffic. Control and Management interfaces will allow the configuration of the NSF manipulation features, such as redirect and block rules. Some examples are:
    * Redirect traffic, as in the case of captive portals,
    * Block traffic: Firewalls, intrusion prevention system, DDOS/ Anti-DOS (Distributed Denial-of-Service/Anti-Denial-of- Service),
    * Encrypt traffic: VPN services that encapsulate and encrypt the user traffic. A SSL VPN is a representative example.

  o Impersonation: Some NSFs can impersonate a customer service or Internet service to provide security functions. Control and Management interfaces will allow the configuration of the service to impersonate and his behavioral. Some examples are:
    * Honeypots, impersonating customer services, such as HTTP, NetBios or SSH,
    * Anonymization services, hiding the source identity, as in the case of TOR.

Service Chain will allow for more than one of the aforementioned functions to engage in a specific order to a particular flow

5. Gap Analysis

5.1. Structure of the gap analysis

This document provides a analysis of the gaps in the state of art in the following industry forums:

IETF working groups (section 5.2)

ETSI Network Functions Virtualization Industry Specification Group (ETSI NFV ISG), (section 5.3)

OPNFV Open Source Group (section 5.4)

Open Stack - Firewall as a service (OpenStack Firewall FaaS) (section 5.5) (http://docs.openstack.org/admin-guide-cloud/content/install_neutron-fwaas-agent.html)
5.2. IETF Gap analysis

The IETF gap analysis first examines the IETF mechanisms which have been developed to secure the IP traffic flows through a network. Traffic filters have been defined by IETF specifications at the access points, the middle-boxes, or the routing systems. Protocols have been defined to carry provisioning and filtering traffic between a management system and an IP system (router or host system). Current security work (SACM working group (WG), MILE WG, and DOTS WG) is providing correlation of events monitored with the policy set by filters. This section provides a review the filter work, protocols, and security correlation for monitors.

5.2.1. Traffic Filters

5.2.1.1. Overview

The earliest filters defined by IETF were access filters which controlled the acceptance of IP packet data flows. Additional policy filters were created as part of the following protocols:

- COPS protocol [RFC2748] for controlling access to networks,
- Next steps in Signalling (NSIS) work (architecture: [RFC4080] protocol: [RFC5973]), and
- the Port Control Protocol (PCP) to enables IPv4 to IPv6 flexible address and port mapping for NATs and Firewalls,

Today NETMOD and I2RS Working groups are specifying additional filters in Yang modules to be used as part of the NETCONF or I2RS enhancement of NETCONF/RESTCONF.

The routing filtering is outside the scope of the flow filtering, but flow filtering may be impacted by route filtering. An initial model for the routing policy is in [I-D.shaikh-rtgwg-policy-model]

This section provides an overview of the flow filtering as an introduction to the I2NSF GAP analysis. Additional detail on NETCONF, NETMOD, I2RS, PCP, and NSIS is available in the Detailed I2NSF analysis.
5.2.1.1.1. Data Flow Filters in NETMOD and I2RS

The current work on expanding these filters is focused on combining a configuration and monitoring protocol with Yang data models. [I-D.ietf-netmod-acl-model] provides a set of access lists filters which can permit or deny traffic flow based on headers at the MAC, IP layer, and Transport layer. The configuration and monitoring protocols which can pass the filters are: NETCONF protocol [RFC6241], RESTCONF [I-D.ietf-netconf-restconf], and the I2RS protocol. The NETCONF and RESTCONF protocols install these filters into forwarding tables. The I2RS protocol uses the ACLs as part of the filters installed in an ephemeral protocol-independent filter-based RIB [I-D.kini-i2rs-fb-rib-info-model] which controls the flow of traffic on interfaces specifically controlled by the I2RS filter-based FIB.

netconf
+---------------+    /  \\     +---------------+
| Device: ACLs  |-- / \---|Device: ACLS   |
| I2RS FB RIB  |          | I2RS FB RIB   |
| routing policy|          | routing policy|
|               | =             |               |
+---------------+  data flow +---------------+

Figure 5 - I2RS Filter-Based RIB

The I2RS protocol is a programmatic interface to the routing system. At this time, the I2RS is targeted to be extensions to the NETCONF/RESTCONF protocols to allow the NETCONF/RESTCONF protocol to support a highly programmatic interface with high bandwidth of data, highly reliable notifications, and ephemeral state (see [I-D.ietf-i2rs-architecture]). Please see the background section on I2RS for additional details on the requirements for this extension to the NETCONF/RESTCONF protocol suite.

The vocabulary set in [I-D.ietf-netmod-acl-model] is limited, so additional protocol independent filters were written for the I2RS Filter-Based RIBs in [I-D.hares-i2rs-bnp-eca-data-model], and protocol specific filters for SFC [I-D.dunbar-i2rs-discover-traffic-rules].

One thing important to note is that NETCONF and RESTCONF manage device layer yang models. However, as figure 6 shows, there are multiple device level, network-wide level, and application level yang modules. The access lists defined by the device level forwarding table may be impacted by the routing protocols, the I2RS ephemeral protocol independent Filter-Based FIB, or some network-wide security issue (IPS/IDS).
5.2.1.1.2. I2NSF Gap analysis

The gap is that none of the current work on these filters considers all the variations of data necessary to do IPS/IDS, web-filters, stateful flow-based filtering, security-based deep packet inspection, or pattern matching with re-mediation. The I2RS Filter-Based RIB work is the closest associated work, but the focus has not been on IDS/IPS, web-filters, security-based deep packet inspection, or pattern matching with re-mediation.

The I2RS Working group (I2RS WG) is focused on the routing system so security expertise for these IDP/IPS, Web-filter, security-based deep-packet inspection has not been targeted for this WG.

Another gap is there is no capability registry (an IANA registry) that identifies the characteristics and behaviours of NSFs in vendor-neutral vocabulary without requiring the NSFs to be standardized.

What I2NSF can use from NETCONF/RESTCONF and I2RS

I2NSF should consider using NETCONF/RESTCONF protocol and the I2RS proposed enhancement to the NETCONF/RESTCONF protocol.

5.2.1.2. Middle-box Filters

5.2.1.2.1. Midcom

Midcom Summary: MIDCOM developed the protocols for applications to communicate with middle boxes. However, MIDCOM have not used by the industry for a long time. This is because there was a lot of IPR
encumbered technology and IPR was likely a bigger problem for IETF than it is today. MIDCOM is not specific to SIP. It was very much oriented to NAT/FW devices. SIP was just one application that needed the functionality. MIDCOM is reservation-oriented and there was an expectation that the primary deployment environment would be VoIP and real-time conferencing, including SIP, H.323, and other reservation-oriented protocols. There was an assumption that there would be some authoritative service that would have a view into endpoint sessions and be able to authorize (or not) resource allocation requests. In other word, there’s a trust model there that may not be applicable to endpoint-driven requests without some sort of trusted authorization mechanisms/tools. Therefore, there is a specific information model applied to security devices, and security device requests, that was developed in the context of an SNMP MIB. There is also a two-stage reservation model, which was specified in order to allow better resource management.

Why I2NSF is different than Midcom

MIDCOM is different than I2NSF because its SNMP scheme doesn’t work with the virtual network security functions (vNSF) management.

MidCom RFCs:

[RFC3303] - Midcom architecture
[RFC5189] - Midcom Protocol Semantics
[RFC3304] - Midcom protocol requirements

5.2.1.3. Security Work

5.2.1.3.1. Overview

Today’s NSFs in security devices can handle flow-based security by providing treatment to packets/flows, such as IPS/IDS, Web filtering, flow filtering, deep packet inspection, or pattern matching and remediation. These flow-based security devices are managed and provisioned by network management systems.

No standardized set of interoperable interfaces control and manage the NSFs so that a central management system can be used across security devices from multiple Vendors. I2NSF work plan is to standardize a set of interfaces by which control and management of NSFs may be invoked, operated, and monitored by:

creating an information model that defines concepts required for standardizing the control and monitoring of NSFs, and from the
information model create data models. (The information model will be used to get early agreement on key technical points.)

creating a capability registry (at IANA) that enables the characteristics and behavior of NSFs to be specified using a vendor-neutral vocabulary without requiring the NSFs themselves to be standardized.

define the requirements for an I2NSF protocol to pass this traffic. (Hopefully re-using existing protocols.)

The flow-filtering configuration and management must fit into the existing security area’s work plan. This section considers how the I2NSF fits into the security area work under way in the SACM (security automation and control), DOTS (DDoS Open Threat Signalling), and MILE (Management Incident Lightweight Exchange).

5.2.1.3.2. Security Work and Filters

In the proposed I2NSF work plan, the I2NSF security network management system controls many NSF nodes via the I2NSF Agent. This control of data flows is similar to the COPS example in section 5.7.4.

```
+------------+
| I2NSF      |
| Client     |
|  security  |
| NMS system |
+------------+

+-----+      /    \      +-----+
|I2NSF|--/       \--|I2NSF|
|Agent|         |Agent|
| NSF ---|          | NSF ---|
|-----+        data flow +-----|

Figure 7
```

The other security protocols work to interact within the network to provide additional information in the following way:

- SACM [I-D.ietf-sacm-architecture] describes an architecture which tries to determine if the end-point security policies and the reality (denoted as security posture) align.
[I-D.ietf-sacm-terminology] defines posture as the configuration and/or status of hardware or software on an endpoint as it pertains to an organization’s security policy. Filters can be considered on the configuration or status pieces that needs to be monitored.

- DOTS (DDoS Open Threat Signalling) - is working on coordinating the mitigation of DDoS attacks. A part of DDoS attach mitigation is to provide lists of addresses to be filtered via IP header filters.

- MILE (Managed Incident LIghtweight Exchange) - is working on creating a standardized format for incident and indicator reports, and creating a protocol to transport this information. The incident information MILE collects may cause changes in data-flow filters on one or more NSFs.

5.2.1.3.3. I2NSF interaction

The network management system that the I2NSF client resides on may interact with other clients or agents developed for the work ongoing in the SACM, DOTS, and MILES working groups. This section describes how the addition of I2NSF’s ability to control and monitor NSF devices is compatible and synergistic with these existing efforts.
Figure 8 provides a diagram of a system the I2NSF, SACM, DOTS and MILES client-agent or consumer-broker-provider are deployed together. The following are possible positive interactions these scenario might have:

- An security network management system (NMS) can contain a SACM repository and be connected to SACM information provider and a SACM consumer. The I2NSF may provide one of the ways to change the forwarding filters.

- The security NMS may also be connected to DOTS DDoS clients managing the information and configuring the rules. The I2NSF may provide one of the ways to change forwarding filters.

- The MILES client on a security network management system talking to the MILES agent on the node may react to the incidents by using I2NSF to set filters. DOTS creates black-lists, but does not have a complete set of filters.
5.2.1.3.4. Benefits from the Interaction

I2NSF’s ability to provide a common interoperable and vendor neutral interface may allow the security NMS to use a single change to change filters. SACM provides an information model to describe end-points, but does not link this directly to filters.

DOTS creates black-lists based on source and destination IP address, transport port number, protocol ID, and traffic rate. Like NETMOD’s, ACLS are not sufficient for all filters or control desired by the NSF boxes.

The incident data captured by MILES will not have enough filter information to provide NSF devices with general services. The I2NSF will be able to handle the MILE incident data and create alerts or reports for other security systems.

5.3. ETSI NFV

5.3.1. ETSI Overview

Network Function Virtualization (NFV) provides the service providers with flexibility, cost effective and agility to offer their services to customers. One such service is the network security function which guards the exterior of a service provider or its customers.

The flexibility and agility of NFV encourages service providers to provide different products to address business trends in their market to provide better service offerings to their end user. A traditional product such as the network security function (NSF) may be broken into multiple virtual devices each hosted from another vendor. In the past, network security devices may have been single sourced from a small set of vendors - but in the NFV version of NSF devices, this reduced set of sources will not provide a competitive edge. Due to this market shift, the network security device vendors are realizing that the proprietary provisioning protocols and formats of data may be a liability. Out of the NFV work has arisen a desire for a single interoperable network security device provisioning and control protocol.

The I2NSF will be deployed along networks using other security and NFV technology. As section 3 described, the NFV NSF security is deployed alongside other security functions (AAA, SACM, DOTS, and MILE devices) or deep-packet-inspection. The ETSI Network Functions Virtualization: NFV security: Security and Trust guidance document (ETSI NFV SEC 003 1.1.1 (2014-12)) indicates that multiple administrative domains will deployed in carrier networks. One example of these multiple domains is hosting of multiple tenant
domains (telecom service providers) on a single infrastructure domain (infrastructure service) as figure 9 shows. The ETSI Inter inter-VNFM document (aka Ve-Vnfn) between the element management system and the Virtual network function is the equivalent of the interface between the I2NSF client on a management system and the I2NSF agent on the network security feature VNF.

Figure 9

The ETSI proof of concept work has worked on the following security proof of concepts:

- #16 - NFVaaS with Secure, SDN controlled WAN Gateway,

5.3.2. I2NSF Gap Analysis

The I2NSF will be deployed on top of virtual computing linked together by virtual routers configured by NETCONF/RESTCONF or I2RS which provision and monitoring the L1, L2, L3 and service pathways through the network.
In the NFV-related productions, the current architecture does not have a protocol to maintain an interoperability provisioning from I2NSF client to I2NSF agent. The result is that service providers have to manage the interoperability using private protocols. In response to this problem, the device manufacturers and the service providers have begun to discuss an I2NSF protocol for interoperable passing of provisioning and filter in formation.

Open source work (such as OPNFV) provides a common code base for providers to start their NFV work from. However, this code base faces the same problem. There is no defacto standard protocol.

5.4. OPNFV

The OPNFV (www.opnfv.org) is a carrier-grade integrated, open source platform focused on accelerating the introduction of new Network Function Virtualization (NFV) products and service. The OPNFV Moon project is focused on adding the security interface for a network management system within the Tenant NFVs and the infrastructure NFVs (as shown in figure 4). This section provides an overview of the OPNFV Moon project and a gap analysis between I2NSF and the OPNFV Moon Project.

5.4.1. OPNFV Moon Project

The OPNFV moon project (https://wiki.opnfv.org) is a security management system. NFV uses cloud computing technologies to virtualize the resources and automate the control. The Moon project is working on a security manager for the Cloud computing infrastructure (https://wiki.opnfv.org/moon). The Moon project proposes to provision a set of different cloud resources/services for VNFs (Virtualized Network Functions) while managing the isolation of VNS, protection of VNFs, and monitoring of VNS. Moon is creating a security management system for OPNFV with security managers to protect different layers of the NFV infrastructure. The Moon project is choosing various security project mechanisms "a la cart" to enforcement related security managers. A security management system integrates mechanisms of different security aspects. This project will first propose a security manager that specifies users’ security requirements. It will also enforce the security managers through various mechanisms like authorization for access control, firewall for networking, isolation for storage, logging for tractability, etc.

The Moon security manager operates a VNF security manager at the ETSI VeVnfm level where the I2NSF protocol is targeted as figure 10 shows. Figure 10 also shows how the OPNFV VNF Security project mixes the I2NSF level with the device level.
The Moon project lists the following gaps in OpenStack:

- No centralized control for compute, storage, and networking. OpenStack uses Nova for computing and Swift for software. Each system has a configuration file and its own security policy. This lacks the synchronization mechanism to build a complete secure configuration for OPNF.
- No dynamic control so that if a user obtains the token, there is no way to obtain control over the user.
- No customization or flexibility to allow integration into different vendors.
- No fine grain authorization at user level. Authorization is only at the API.

Moon addresses these issues adding authorization, logging, IDS, enforcement of network policy, and storage protection. Moon is based on OpenStack Keystone.

Deliverable time frame: 2S 2015
5.4.2. Gap Analysis for OPNFV Moon Project

OpenStack congress does not provide vendor independent systems.

5.5. OpenStack Security Firewall

OpenStack has advanced features of: a) API for managing security groups (http://docs.openstack.org/admin-guide-cloud/content/section_securitygroups.html) and b) firewalls as a service (http://docs.openstack.org/admin-guide-cloud/content/fwaas_api_abstractions.html).

This section provides an overview of this open stack work, and a gap analysis of how I2NSF provides additional functions.
5.5.1. Overview of API for Security Group

The security group with the security group rules provides ingress and egress traffic filters based on port. The default group drops all ingress traffic and allows all egress traffic. The groups with additional filters are added to change this behaviour. To utilize the security groups, the networking plug-in for Open Stack must implement the security group API. The following plug-ins in OpenStack currently implement this security: ML2, Open vSwitch, Linux Bridge, NEC, and VMware NSX. In addition, the correct firewall driver must be added to make this functional.

5.5.2. Overview of Firewalls as a Service

Firewall as a service is an early release of an API that allows early adopters to test network implementations. It contains APIs with parameters for firewall rules, firewall policies, and firewall identifiers. The firewall rules include the following information:

- identification of rule (id, name, description)
- identification tenant rule associated with,
- links to installed firewall policy,
- IP protocol (tcp, udp, icmp, none)
- source and destination IP address
- source and destination port
- action: allow or deny traffic
- status: position and enable/disabled

The firewall policies include the following information:

- identification of the policy (id, name, description),
- identification of tenant associated with,
- ordered list of firewall rules,
- indication if policy can be seen by tenants other than owner, and
- indication if firewall rules have been audited.

The firewall table provides the following information:
o identification of firewall (id, name, description),

o tenant associated with this firewall,

o administrative state (up/down),

o status (active, down, pending create, pending delete, pending update, pending error)

o firewall policy ID this firewall is associated with

5.5.3. I2NSF Gap analysis

The OpenStack work is preliminary (security groups and firewall as a service). This work does not allow any of the existing network security vendors provide a management interface. Security devices take time to be tested for functionality and their detection of security issues. The OpenStack work provides an interesting simple set of filters, and may in the future provide some virtual filter service. However, at this time this open source work does not address the single management interfaces for a variety of security devices.

I2NSF is proposing rules that will include Event-Condition-matches (ECA) with the following matches

- packet based matches on L2, L3, and L4 headers and/or specific addresses within these headers,
- context based matches on schedule state and schedule, [Editor: Need more details here.]

The I2NSF is proposing action for these ECA policies of:

- basic actions of deny, permit, and mirror,
- advanced actions of: IPS signature filtering and URL filtering.

5.6. CSA Secure Cloud

5.6.1. CSA Overview

The Cloud Security Alliance (CSA) (www.cloudsecurityalliance.org) defined security as a service (SaaS) in their Security as a Service working group (SaaS WG) during 2010-2012. The CSA SaaS group defined ten categories of network security (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_V1_0.pdf) and provides implementation guidance for each of
these ten categories. This section provides an overview of the CSA SaaS working groups documentation and a Gap analysis for I2NSF.

5.6.1.1. CSA Security as a Service (SaaS)

The CSA SaaS working group defined the following ten categories, and provided implementation guidance on these categories:

1. Identity Access Management (IAM) (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_1_IAM_Implementation_Guidance.pdf)


4. Email Security (email) (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_4_Email_Security_Implementation_Guidance.pdf),


9. Business Continuity and Disaster Recovery (BCDR) (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_9_BCDR_Implementation_Guidance.pdf), and

The sections below give an overview these implementation guidances

5.6.1.2. Identity Access Management (IAM)

document: (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_1_IAM_Implementation_Guidance.pdf)

The identity management systems include the following services:

- Centralized Directory Services,
- Access Management Services,
- Identity Management Services,
- Identity Federation Services,
- Role-Based Access Control Services,
- User Access Certification Services,
- Privileged User and Access Management,
- Separation of Duties Services, and
- Identity and Access Reporting Services.

The IAM device communications with the security management system that controls the filtering of data. The CSA SaaS IAM specification states that interoperability between IAM devices and secure access network management systems is a problem. This 2012 implementation report confirms there is a gap with I2NSF.

Figure 11

5.6.1.3. Data Loss Prevention (DLP)


The data loss prevention (DLP) services must address:

- origination verification,
- integrity of data,
- confidentiality and access control,
- accountability,
- avoiding false positives on detection, and
- privacy concerns.

The CSA SaaS DLP device communications require that it have the enforcement capabilities to do the following:

- alert and log data loss,
- delete data on system or passing through,
- filter out (block/quarantine) data,
- reroute data,
- encrypt data

```
+-------------------+  |        +        |
| DLP device        |  |--- SLA -----| secure |
|                   |  | Alert and log| access |
|                   |  | delete data  | NMS    |
|                   |  | filter/reroute| encrypt data |
|                   |  |              |        |
|                   |  |----------|       |        |
|                   |  | Filter enforcement |        |
|                   |  |----------|        |
```

Figure 12
5.6.1.4. Web security(Web))

Document:
https://downloads.cloudsecurityalliance.org/initiatives/secaas/
SecaaS_Cat_3_Web_Security_Implementation_Guidance.pdf

The web security services must address:

- Web 2.0/Social Media controls,
- Malware and Anti-Virus controls,
- Data Loss Prevention controls (over Web-based services like Gmail or Box.net),
- XSS, JavaScript and other web specific attack controls
- Web URL Filtering,
- Policy control and administrative management,
- Bandwidth management and quality of service (QoS) capability, and
- Monitoring of SSL enabled traffic.

The CSA SaaS Web services device communications require that it have the enforcement capabilities to do the following:

- alert and log malware or anti-virus data patterns,
- delete data (malware and virus) passing through systems,
- filter out (block/quarantine) data,
- filter Web URLs,
- interact with policy and network management systems,
- control bandwidth and QoS of traffic, and
- monitor encrypted (SSL enabled) traffic,

All of these features either require the I2NSF standardized I2NSF client to I2NSF agent to provide multi-vendor interoperability.
5.6.1.5. Email Security (email))


The CSA Document recommends that email security services must address:

- Common electronic mail components,
- Electronic mail architecture protection,
- Common electronic mail threats,
- Peer authentication,
- Electronic mail message standards,
- Electronic mail encryption and digital signature,
- Electronic mail content inspection and filtering,
- Securing mail clients, and
- Electronic mail data protection and availability assurance techniques

The CSA SaaS Email security services requires that it have the enforcement capabilities to do the following:

- provide the malware and spam detection and removal,
alert and provide rapid response to email threats,
identify email users and secure remote access to email,
do on-demand provisioning of email services,
filter out (block/quarantine) email data,
know where the email traffic or data is residing (to regulatory issues), and
be able to monitor encrypted email,
be able to encrypt email,
be able to retain email records (while abiding with privacy concerns), and
interact with policy and network management systems.

All of these features require the I2NSF standardized I2NSF client to I2NSF agent to provide multi-vendor interoperability.

Figure 14

5.6.1.6. Security Assessment

Document:
https://downloads.cloudsecurityalliance.org/initiatives/secaas/
SecaaS_Cat_5_Security_Assessments_Implementation_Guidance.pdf

The CSA SaaS Security assessment indicates that assessments need to be done on the following devices:

- hypervisor infrastructure,
o network security compliance systems,
o Servers and workstations,
o applications,
o network vulnerabilities systems,
o internal auditor and intrusion detection/prevention systems (IDS/IPS), and
o web application systems.

All of these features require the I2NSF working group standardize the way to pass these assessments to and from the I2NSF client on the I2NSF management system and the I2NSF Agent.

5.6.1.7. Intrusion Detection

Document:

The CSA SaaS Intrusion detection management includes intrusion detection through: devices:

o Network traffic inspection, behavioural analysis, and flow analysis,
o Operating System, Virtualization Layer, and Host Process Events monitoring,
o monitoring of Application Layer Events, and
o Correlation Techniques, and other Distributed and Cloud-Based Capabilities

Intrusion response includes both:

o Automatic, Manual, or Hybrid Mechanisms,

The CSA SaaS recommends the intrusion security management systems include provisioning and monitoring of all of these types of intrusion detection (IDS) or intrusion protection devices. The management of these systems requires also requires:
Central reporting of events and alerts,
administrator notification of intrusions,
Mapping of alerts to Cloud-Layer Tenancy,
Cloud sourcing information to prevent false positives in
detection, and
allowing for redirection of traffic to allow remote storage or
transmission to prevent local evasion.

All of these features require the I2NSF standardized I2NSF client to
I2NSF agent to provide multi-vendor interoperability.

<table>
<thead>
<tr>
<th>IDS/IPS</th>
<th>Info</th>
<th>secure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>alert/log intrusion</td>
<td>notify administrator</td>
</tr>
<tr>
<td></td>
<td>Map alerts to Tenant</td>
<td>filter/reroute traffic</td>
</tr>
<tr>
<td></td>
<td>remote data storage</td>
<td>---------------</td>
</tr>
<tr>
<td>+------</td>
<td>---------------------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>Filter enforcement</td>
<td>-----</td>
</tr>
</tbody>
</table>

Figure 15

5.6.1.8. Security Information and Event Management (SEIM)

Document: https://downloads.cloudsecurityalliance.org/initiatives/secaas/
SecaaS_Cat_7_SIEM_Implementation_Guidance.pdf)

The Security Information and Event Management (SEIM) receives data
from a wide range of security systems such as Identity management
systems (IAM), data loss prevention (DLP), web security (Web), email
security (email), intrusion detection/prevention (IDS/IPS)),
enetration, disaster recovery, and network security. The SEIM
combines this data into a single streams. All the requirements for
data to/from these systems are replicated in these systems needs to
give a report to the SIEM system.

A SIEM system would be prime candidate to have a I2NSF client that
gathers data from an I2NSF Agent associated with these various types
of security systems. The CSA SaaS SIEM functionality document
suggests that one concern is to have standards that allow timely recording and sharing of data. I2NSF can provide this.

5.6.1.9. Encryption

Document:
https://downloads.cloudsecurityalliance.org/initiatives/secaas/
SecaaS_Cat_8_Encryption_Implementation_Guidance.pdf

The CSA SaaS Encryption implementation guidance document considers how one implements and manages the following security systems:

- key management systems (KMS), control of keys, and key life cycle;
- Shared Secret encryption (Symmetric ciphers),
- No-Secret or Public Key Encryption (asymmetric ciphers),
- hashing algorithms,
- Digital Signature Algorithms,
- Key Establishment Schemes,
- Protection of Cryptographic Key Material (FIPS 140-2; 140-3),
- Interoperability of Encryption Systems, Key Conferencing, Key Escrow Systems, and others

application of Encryption for Data at rest, data in transit, and data in use;

PKI (including certificate revocation "CRL");

Future application of such technologies as Homomorphic encryption, Quantum Cryptography, Identitybased Encryption, and others;

Crypto-system Integrity (How bad implementations can under mind a crypto-system), and

Cryptographic Security Standards and Guidelines

The wide variety of encryption services require the security management systems be able to provision, monitor, and control the systems that are being used to encrypt data. This document indicates in the implementation sections that the standardization of interfaces to/from management systems are key to good key management systems, encryption systems, and crypto-systems.
5.6.1.10. Business Continuity and Disaster Recovery (BC/DR)

Document:
https://downloads.cloudsecurityalliance.org/initiatives/secaas/
SecaaS_Cat_9_BCDR_Implementation_Guidance.pdf

The CSA SaaS Business Continuity and Disaster Recovery (BC/DR) implementation guidance document considers the systems that implement the contingency plans and measures designed and implemented to ensure operational resiliency in the event of any service interruptions. BC/DR systems includes:

- Business Continuity and Disaster Recovery BC/DR as a service, including categories such as complete Disaster Recovery as a Service (DRaaS), and subsets such as file recovery, backup and archive,
- Storage as a Service including object, volume, or block storage;
- old Site, Warm Site, Hot Site backup plans;
- IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service);
- Insurance (and insurance reporting programs)
- Business Partner Agents (business associate agreements);
- System Replication (for high availability);
- Fail-back to Live Systems mechanisms and management;
- Recovery Time Objective (RTO) and Recovery Point Objective (RPO);
- Encryption (data at rest [DAR], data in motion [DIM], field level);
- Realm-based Access Control;
- Service-level Agreements (SLA); and

These BC/DR systems must handle data backup and recovery, server backup/recovery, and data center (virtual/physical) backup and recovery. Recovery as a service (RaaS) means that the BC/DR services are being handled by management systems outside the enterprise.
The wide variety of BC/DR requires the security management systems to be able to communicate provisioning, monitor, and control those systems that are being used to back-up and restore data. An interoperable protocol that allows provision and control of data center's data, servers, and data center management devices is extremely important to this application. Recovery as a Service (SaaS) indicates that these services need to be able to be remotely managed.

The CSA SaaS BC/BR documents indicate how important a standardized I2NSF protocol is.

5.6.1.11. Network Security Devices

Document:
https://downloads.cloudsecurityalliance.org/initiatives/secaas/
SecaaS_Cat_10_Network_Security_Implementation_Guidance.pdf

The CSA SaaS Network Security implementation recommendation includes advice on:

- How to segment networks,
- Network security controls,
- Controlling ingress and egress controls such as Firewalls (Stateful), Content Inspection and Control (Network-based), Intrusion Detection System/Intrusion Prevention Systems (IDS/IPS), and Web Application Firewalls,
- Secure routing and time,
- Denial of Service (DoS) and Distributed Denial of Service (DDoS) Protection/Mitigation,
- Virtual Private Network (VPN) with Multiprotocol Label Switching (MPLS) Connectivity (over SSL), Internet Protocol Security (IPsec) VPNs, Virtual Private LAN Service (VPLS), and Ethernet Virtual Private Line (EVPL),
- Threat Management,
- Forensic Support, and
- Privileged User/Use Monitoring.

These network security systems require provisioning, monitoring, and the ability for the security management system to subscribe to
receive logs, snapshots of capture data, and time synchronization. This document states the following:

"It is critical to understand what monitoring APIs are available from the CSP, and if they match risk and compliance requirements",

"Network security auditors are challenged by the need to track a server and its identity from creation to deletion. Audit tracking is challenging in even the most mature cloud environments, but the challenges are greatly complicated by cloud server sprawl, the situation where the number of cloud servers being created is growing more quickly than a cloud environments ability to manage them."

A valid threat vector for cloud is the API access. Since a majority of CSPs today support public API interfaces available within their networks and likely over the Internet.

The CSA SaaS network security indicates that the I2NSF must be secure so that the I2NSF Client-Agent protocol does not become a valid threat vector. In additions, the need for the management protocol like I2NSF is critical in the sprawl of Cloud environment.

5.6.2. I2NSF Gap Analysis

The CSA Security as a Service (SaaS) document show clearly that there is a gap between the ability of the CSA SaaS devices to have a vendor neutral, inoperable protocol that allow the multiple of network security devices to communicate passing provisioning and informational data. Each of the 10 implementation agreements points to this as a shortage. The I2NSF yang models and protocol is needed according to the CSA SaaS documents.

5.7. In-depth Review of IETF protocols

5.7.1. NETCONF and RESTCONF

The IETF NETCONF working group has developed the basics of the NETCONF protocol focusing on secure configuration and querying operational state. The NETCONF protocol [RFC6241] may be run over TLS [RFC6639] or SSH ([RFC6242]. NETCONF can be expanded to defaults [RFC6243], handling events ([RFC5277] and basic notification [RFC6470], and filtering writes/reads based on network access control models (NACM, [RFC6536]). The NETCONF configuration must be committed to a configuration data store (denoted as config=TRUE). Yang models identify nodes within a configuration data store or an operational data store using a XPath expression (document root ---to --- target source). NETCONF uses an RPC model and provides protocol
for handling configs (get-config, edit-config, copy-config, delete-config, lock, unlock, get) and sessions (close-session, kill-session). The NETCONF Working Group has developed RESTCONF, which is an HTTP-based protocol that provides a programmatic interface for accessing data defined in YANG, using the datastores defined in NETCONF.

RESTCONF supports "two edit condition detections" - time stamp and entity tag. RESTCONF uses a URI encoded path expressions. RESTCONF provides operations to get remote servers options (OPTIONS), retrieve data headers (HEAD), get data (GET), create resource/invoke operation (POST), patch data (PATCH), delete resource (DELETE), or query.

RFCs for NETCONF
o NETCONF [RFC6242]
o NETCONF monitoring [RFC6022]
o NETCONF over SSH [RFC6242]
o NETCONF over TLS [RFC5539]
o NETCONF system notification [RFC6470]
o NETCONF access-control (NACM) [RFC6536]
o RESTCONF [I-D.ietf-netconf-restconf]
o NETCONF-RESTCONF call home [I-D.ietf-netconf-call-home]
o RESTCONF collection protocol [I-D.ietf-netconf-restconf-collection]
o NETCONF Zero Touch Provisioning [I-D.ietf-netconf-zerotouch]

5.7.2. I2RS Protocol

Based on input from the NETCONF working group, the I2RS working group decided to re-use the NETCONF or RESTCONF protocols and specify additions to these protocols rather than create yet another protocol (YAP).

The required extensions for the I2RS protocol are in the following drafts:
o Ephemeral state [I-D.ietf-i2rs-ephemeral-state],
o Publication-Subscription notifications
[I-D.ietf-i2rs-pub-sub-requirements],
o Traceability [I-D.ietf-i2rs-traceability],
o Security requirements [I-D.hares-i2rs-auth-trans]

At this time, NETCONF and RESTCONF cannot handle the ephemeral data store proposed by I2RS, the publication and subscription requirements, the traceability, or the security requirements for the transport protocol and message integrity.

5.7.3. NETMOD Yang modules

NETMOD developed initial Yang models for interfaces ([RFC7223]), IP address ([RFC7277]), IPv6 Router advertisement ([RFC7277]), IP Systems ([RFC7317]) with system ID, system time management, DNS resolver, Radius client, SSH, syslog ([I-D.ietf-netmod-syslog-model]), ACLS ([I-D.ietf-netmod-acl-model]), and core routing blocks ([I-D.ietf-netmod-routing-cfg] The routing working group (rtgwg) has begun to examine policy for routing and tunnels.

Protocol specific Working groups have developed yang models for ISIS ([I-D.ietf-isis-yang-isis-cfg]), OSPF ([I-D.ietf-ospf-yang]), and BGP (merge of [I-D.shaikh-idr-bgp-model] and [I-D.zhdankin-idr-bgp-cfg] with the bgp policy proposed multiple Working groups (idr and rtgwg)). BGP Services yang models have been proposed for PPB EVPN ([I-D.tsingh-bess-pbb-evpn-yang-cfg]), EVPN ([I-D.zhuang-bess-evpn-yang]), L3VPN ([I-D.zhuang-bess-l3vpn-yang]), and multicast MPLS/BGP IP VPNs ([I-D.liu-bess-mvpn-yang]).

5.7.4. COPS

One early focus on flow filtering based on policy enforcement of traffic entering a network is the 1990s COPS ([RFC2748] design (PEP and PDP) as shown in figure 16. The Policy decision point kept network-wide policy (E.g. ACLs) and sent it to Policy enforcements which then would control what data flows between the two. These decision points controlled data flow from PEP to PEP. [RFC3084] describes COPS use for policy provisioning.
COPS had a design of Policy Enforcement Points (PEP), and policy Decision Points (PDP) as shown in figure 16. These decision points controlled flow from PEP to PEP.

Why COPS is no longer used

Security in the network in 2015 uses specific devices (IDS/IPS, NAT firewall, etc) with specific policies and profiles for each types of device. No common protocol or policy format exists between the policy manager (PDP) and security enforcement points.

COPS RFCs: [RFC4261], [RFC2940], [RFC3084], [RFC3483]

Why I2NSF is different COPS

COPS was a protocol for policy related to Quality of Service (QoS) and signalling protocols (e.g. RSVP) (security, flow, and others). I2NSF creates a common protocol between security policy decision points (SPDP) and security enforcement points (SEP). Today’s security devices currently only use proprietary protocols. Manufacturers would like a security specific policy enforcement protocol rather than a generic policy protocol.

5.7.5. PCP

As indicated by the name, the Port Control Protocol (PCP) enables an IPv4 or IPv6 host to flexibly manage the IP address and port mapping information on Network Address Translators (NATs) or firewalls, to facilitate communication with remote hosts.

PCP RFCs:

[RFC6887]

[RFC7225]

[I-D.ietf-pcp-authentication]

Why is I2NSF different from PCP:

Here are some aspects that I2NSF is different from PCP:

- PCP only supports the management of port and address information rather than any other security functions
- Cover the proxy, firewall and NAT box proposals in I2NSF

5.7.6. NSIS - Next steps in Signalling

NSIS is for standardizing an IP signalling protocol (RSVP) along data path for end points to request its unique QoS characteristics, unique FW policies or NAT needs (RFC5973) that are different from the FW/NAT original setting. The requests are communicated directly to the FW/NAT devices. NSIS is like east-west protocols that require all involved devices to fully comply to make it work.

NSIS is path-coupled, it is possible to message every participating device along a path without having to know its location, or its location relative to other devices (this is particularly a pressing issue when you’ve got one or more NATs present in the network, or when trying to locate appropriate tunnel endpoints).

A diagram should be added here showing I2NSF and NSIS

Why I2NSF is different than NSIS:

- The I2NSF requests from clients do not go directly to network security devices, but instead to controller or orchestrator that can translate the application/user oriented policies to the involved devices in the interface that they support.
- The I2NSF request does not require all network functions in a path to comply, but it is a protocol between the I2NSF client and the I2NSF Agent in the controller and orchestrator
- I2NSF defines client (applications) oriented descriptors (profiles, or attributes) to request/negotiate/validate the network security functions that are not on the local premises.

Why we believe I2NSF has a higher chance to be deployed than NSIS:
Open Stack already has a proof-of-concept/preliminary implementation, but the specification is not complete. IETF can play an active role to make the specification for I2NSF complete. IETF can complete and extend the OpenStack implementation to provide an interoperable specification that can meet the needs and requirements of operators and is workable for suppliers of the technology. The combination of a carefully designed interoperable IETF specification with an open-source code development Open Stack will leverage the strengths of the two communities, and expand the informal ties between the two groups. A software development cycle has the following components: architecture, design specification, coding, and interoperability testing. The IETF can take ownership of the first two steps, and provide expertise and a good working atmosphere (in hack-a-thons) in the last two steps for OpenSTack or other open-source coders.

IETF has the expertise in security architecture and design for interoperable protocols that span controllers/routers, middle-boxes, and security end-systems.

IETF has a history of working on interoperable protocols or virtualized network functions (L2VPN, L3VPN) that are deployed by operators in large scale devices. IETF has a strong momentum to create virtualized network functions (see SFC WG in routing) to be deployed in network boxes. [Note: We need to add SACM and others here].

6. Summarized Requirements

The I2NSF framework should provide a set of standard interfaces that facilitate:

- Dynamic creation, enablement, disablement, and removal of network security functions;
- Policy-driven placement of new function instances in the right administrative domain;
- Attachment of appropriate security and traffic policies to the function instances;
- Management of deployed instances in terms of fault monitoring, utilization monitoring, event logging, inventory, etc.

Moreover, an I2NSF must support different deployment scenarios:

- Single and multi-tenant environments: The term multi-tenant does not mean just different companies subscribing to a provider’s
offering. It can for instance cover administrative domains/ departments within a single firm that require different security and traffic policies.

- Premise-agnostic: Said network security functions may be deployed on premises or off premises of an organization.

The I2NSF framework should provide a standard set of interfaces that enable:

- Translation of security policies into functional tasks. Security policies may be carried out by one or more security functions. For example, a security policy may be translated into an IDS/IPS policy and a firewall policy for a given application type.

- Translation of functional tasks into vendor-specific configuration sets. For example, a firewall policy needs to be converted to vendor-specific configurations.

- Retrieval of information such as configuration, utilization, status, etc. Such information may be used for monitoring, auditing, troubleshooting purposes. The above functionality should be available in single- or multi-tenant environments as well as on-premise or off-premise clouds.

7. IANA Considerations

No IANA considerations exist for this document.

8. Security Considerations

The relationship between different actors define the security level for the different use cases and must be associated with administrative domains:

- Closed environments where there is only one administrative network domain. More permissive access controls and lighter validation shall be allowed inside the domain because of the protected environment. Integration with existing identity management systems is also possible.

- Open environments where some NSFs can be hosted in different administrative domains, and more restrictive security controls are required. The interfaces to the NSFs must use trusted channels. Identity frameworks and federations are common models for authentication and Authorization. Security controllers will be in charge of this functionalities.
Virtualization applied to NSF environment (vNSF) generate several concerns in security, being one of the most relevant the attestation of the vNSF by the clients. A holistic analysis has been done in [NFVSEC].

9. Contributors

I2NSF is a group effort. The following people contributed actively to the initial use case text: Diego R. Lopez (Telefonica I+D), Xiaojun Zhuang (China Mobile), Minpeng Qi (China Mobile), Sumandra Majee (F5), Nic Leymann (Deutsche Telekom), Linda Dunbar (Huawei).

10. References

10.1. Normative References


10.2. Informative References


[I-D.ietf-i2rs-architecture]

[I-D.ietf-i2rs-ephemeral-state]

[I-D.ietf-i2rs-problem-statement]

[I-D.ietf-i2rs-pub-sub-requirements]
Voit, E., Clemm, A., and A. Prieto, "Requirements for Subscription to YANG Datastores", draft-ietf-i2rs-pub-sub-requirements-03 (work in progress), October 2015.

[I-D.ietf-i2rs-rib-data-model]

[I-D.ietf-i2rs-rib-info-model]

[I-D.ietf-i2rs-traceability]

[I-D.ietf-i2rs-usecase-reqs-summary]

[I-D.ietf-i2rs-yang-l2-network-topology]
[I-D.ietf-i2rs-yang-network-topo]

[I-D.ietf-isis-yang-isis-cfg]

[I-D.ietf-netconf-call-home]
Watsen, K., "NETCONF Call Home and RESTCONF Call Home", draft-ietf-netconf-call-home-06 (work in progress), May 2015.

[I-D.ietf-netconf-restconf]

[I-D.ietf-netconf-restconf-collection]

[I-D.ietf-netconf-zerotouch]
Watsen, K., Clarke, J., and M. Abrahamsson, "Zero Touch Provisioning for NETCONF Call Home (ZeroTouch)", draft-ietf-netconf-zerotouch-02 (work in progress), March 2015.

[I-D.ietf-netmod-acl-model]

[I-D.ietf-netmod-routing-cfg]

[I-D.ietf-netmod-syslog-model]
Wildes, C. and K. Sreenivasa, "SYSLOG YANG model", draft-ietf-netmod-syslog-model-03 (work in progress), March 2015.
[I-D.ietf-ospf-yang]
Yeung, D., Qu, Y., Zhang, J., Bogdanovic, D., and K.
Sreenivasa, "Yang Data Model for OSPF Protocol", draft-
ietf-ospf-yang-00 (work in progress), March 2015.

[I-D.ietf-pcp-authentication]
Control Protocol (PCP) Authentication Mechanism", draft-
ietf-pcp-authentication-09 (work in progress), May 2015.

[I-D.ietf-pcp-optimize-keepalives]
Reddy, T., Patil, P., Isomaki, M., and D. Wing,
"Optimizing NAT and Firewall Keepalives Using Port Control
Protocol (PCP)", draft-ietf-pcp-optimize-keepalives-06
(work in progress), May 2015.

[I-D.ietf-pcp-proxy]
Perreault, S., Boucadair, M., Penno, R., Wing, D., and S.
Cheshire, "Port Control Protocol (PCP) Proxy Function",
draft-ietf-pcp-proxy-08 (work in progress), May 2015.

[I-D.ietf-sacm-architecture]
Cam-Winget, N., Lorenzin, L., McDonald, I., and l.
loxx@cisco.com, "Secure Automation and Continuous
Monitoring (SACM) Architecture", draft-ietf-sacm-
architecture-03 (work in progress), March 2015.

[I-D.ietf-sacm-terminology]
Waltermire, D., Montville, A., Harrington, D., Cam-Winget,
N., Lu, J., Ford, B., and M. Kaeo, "Terminology for
Security Assessment", draft-ietf-sacm-terminology-06 (work
in progress), February 2015.

[I-D.kini-i2rs-fb-rib-info-model]
Kini, S., Hares, S., Dunbar, L., Ghanwani, A., Krishnan,
R., Bogdanovic, D., Tantsura, J., and R. White, "Filter-
Based RIB Information Model", draft-kini-i2rs-fb-rib-info-
model-01 (work in progress), July 2015.

[I-D.l3vpn-service-yang]
Litkowski, S., Shakir, R., Tomotaki, L., and K. D’Souza,
"YANG Data Model for L3VPN service delivery", draft-l3vpn-
service-yang-00 (work in progress), February 2015.

[I-D.liu-bess-mvpn-yang]
Liu, Y., and F. Guo, "Yang Data Model for Multicast in
MPLS/BGP IP VPNs", draft-liu-bess-mvpn-yang-00 (work in
progress), April 2015.

[I-D.shaikh-idr-bgp-model]  

[I-D.shaikh-rtgwg-policy-model]  

[I-D.tsingh-bess-pbb-evpn-yang-cfg]  

[I-D.zhang-i2rs-l1-topo-yang-model]  

[I-D.zhdankin-idr-bgp-cfg]  

[I-D.zhuang-bess-evpn-yang]  

[I-D.zhuang-bess-l3vpn-yang]  

[RFC2748]  

[RFC2940]  


Internet-Draft        I2NSF Existing Work Analysis          October 2015


Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI  48176
USA

Email: shares@ndzh.com

Antonio Pastor
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid  28006
Spain

Email: antonio.pastorperales@telefonica.com
Ke Wang
China Mobile
32 Xuanwumenxi Ave, Xicheng District
Beijing 100053
China
Email: wangkeyj@chinamobile.com

Dacheng Zhang
Beijing
China
Email: dacheng.zdc@aliabab-inc.com

Myo Zarny
Goldman Sachs
30 Hudson Street
Jersey City, NJ 07302
USA
Email: myo.zarny@gs.com
Software-Defined Networking Based Security Services using Interface to Network Security Functions
draft-jeong-i2nsf-sdn-security-services-03

Abstract

This document describes a framework, objectives, requirements, and use cases for security services based on Software-Defined Networking (SDN) using a common Interface to Network Security Functions (I2NSF). It first proposes the framework of SDN-based security services in the I2NSF framework. It then explains two use cases, such as centralized firewall system and centralized DDoS-attack mitigation system.

Status of This Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on April 11, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.
This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ................................................. 3
2. Requirements Language ...................................... 4
3. Terminology .................................................. 4
4. Overview .................................................... 5
5. Objectives ................................................... 7
6. Requirements ................................................ 7
7. Use Cases .................................................... 8
   7.1. Centralized Firewall System ............................ 8
   7.2. Centralized DDoS-attack Mitigation System ............ 9
8. Security Considerations .................................... 11
9. Acknowledgements ........................................... 11
10. References ............................................... 11
   10.1. Normative References ................................. 11
   10.2. Informative References .............................. 11
1. Introduction

Software-Defined Networking (SDN) is a set of techniques that enables users to directly program, orchestrate, control and manage network resources through software (e.g., SDN applications). It relocates the control of network resources to a dedicated network element, namely SDN controller. The SDN controller uses interfaces to arbitrate the control of network resources in a logically centralized manner. It also manages and configures the distributed network resources, and provides the abstracted view of the network resources to the SDN applications. The SDN applications can customize and automate the operations (including management) of the abstracted network resources in a programmable manner via the interfaces [RFC7149] [ITU-T.Y.3300] [ONF-SDN-Architecture] [ONF-OpenFlow].

Due to the increase of sophisticated network attacks, the legacy security services become difficult to cope with such network attacks in an autonomous manner. SDN has been introduced to make networks more controllable and manageable, and this SDN technology will be promising to autonomously deal with such network attacks in a prompt manner.

This document describes a framework, objectives and requirements to support the protection of network resources through SDN-based security services using a common interface to Network Security Functions (NSF) [i2nsf-framework]. It uses an interface to NSF (I2NSF) for such SDN-based security services that are performed in virtual machines through network functions virtualization [ETSI-NFV].

This document addresses the challenges of the existing systems for security services. As feasible solutions to handle these challenges, this document proposes two use cases of the security services, such as centralized firewall system and centralized DDoS-attack mitigation system.

For the centralized firewall system, this document raises limitations in legacy firewalls in terms of flexibility and administration costs. Since in many cases, access control management for firewall is manually performed, it is difficult to add the access control policy rules corresponding to new network attacks in a prompt and autonomous manner. Thus, this situation requires expensive administration costs. This document introduces a use case of SDN-based firewall system to overcome these limitations.

For the centralized DDoS-attack mitigation system, this document raises limitations in legacy DDoS-attack mitigation techniques in terms of flexibility and administration costs. Since in many cases, network configuration for the mitigation is manually performed, it is
difficult to dynamically configure network devices to limit and control suspicious network traffic for DDoS attacks. This document introduces a use case of SDN-based DDoS-attack mitigation system to provide an autonomous and prompt configuration for suspicious network traffic.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Terminology

This document uses the terminology described in [RFC7149], [ITU-T.Y.3300], [ONF-SDN-Architecture], [ONF-OpenFlow], [ITU-T.X.1252], and [ITU-T.X.800]. In addition, the following terms are defined below:

- Software-Defined Networking: A set of techniques that enables to directly program, orchestrate, control, and manage network resources, which facilitates the design, delivery and operation of network services in a dynamic and scalable manner [ITU-T.Y.3300].

- Access Control: A procedure used to determine if an entity should be granted access to resources, facilities, services, or information based on pre-established rules and specific rights or authority associated with the requesting party [ITU-T.X.1252].

- Access Control Policy: The set of rules that define the conditions under which access may take place [ITU-T.X.800].


- Network Resources: Network devices that can perform packet forwarding in a network system. The network resources include network switch, router, gateway, WiFi access points, and similar devices.

- Firewall: A firewall that is a device or service at the junction of two network segments that inspects every packet that attempts to cross the boundary. It also rejects any packet that does not satisfy certain criteria for disallowed port numbers or IP addresses.

- Centralized Firewall System: A centralized firewall that can establish and distribute access control policy rules into network
resources for efficient firewall management. These rules can be managed dynamically by a centralized server for firewall. SDN can work as a network-based firewall system through a standard interface between firewall applications and network resources.

- Centralized DDoS-attack Mitigation System: A centralized mitigator that can establish and distribute access control policy rules into network resources for efficient DDoS-attack mitigation. These rules can be managed dynamically by a centralized server for DDoS-attack mitigation. SDN can work as a network-based mitigation system through a standard interface between DDoS-attack mitigation applications and network resources.

4. Overview

This section describes the referenced architecture to support SDN-based security services, such as centralized firewall system and centralized DDoS-attack mitigation system. Also, it describes a framework for SDN-based security services using I2NSF.

![High-level Architecture for SDN-based Security Services](image)

As shown in Figure 1, security functions as security services (e.g., firewall, DDoS-attack mitigation, web filter, deep packet inspection) run on the top of SDN controller [ITU-T.Y.3300] [ONF-SDN-Architecture]. When an administrator enforces security...
policies for such security services through an application interface, SDN controller generates the corresponding access control policy rules to meet such security policies in an autonomous and prompt manner. According to the generated access control policy rules, the network resources such as switches take an action to mitigate network attacks, for example, dropping packets with suspicious patterns.

Figure 2: A Framework for SDN-based Security Services using I2NSF

Figure 2 shows a framework to support SDN-based security services using I2NSF [i2nsf-framework]. As shown in Figure 2, client and application gateway (AppGW) can use security services by delivering their high-level security policies to security controller via service layer interface. Security controller asks security function(s) function-level security services via capability layer interface. The security functions run on top of virtual machines through NFV [ETSI-NFV]. Security functions asks switch controller to perform their required security services on switches under the supervision of switch controller.
Capability layer interface between security controller and security functions can be implemented by Network Configuration Protocol (NETCONF) [RFC6241] with a data modeling language called YANG [RFC6020] that describes function-level security services.

5. Objectives

- Prompt reaction to new network attacks: SDN-based security services allow private networks to defend themselves against new sophisticated network attacks.
- Automatic defense from network attacks: SDN-based security services identify the category of network attack (e.g., malware and DDoS attacks) and take counteraction for the defense without the intervention of network administrators.
- Network-load-aware resource allocation: SDN-based security services measure the overhead of resources for security services and dynamically select resources considering load balance for the maximum network performance.

6. Requirements

SDN-based security services provide dynamic and flexible network resource management to mitigate network attacks, such as malware and DDoS attacks. In order to support this capability, the requirements for SDN-based security services are described as follows:

- SDN-based security services are required to support the programmability of network resources to mitigate network attacks.
- SDN-based security services are required to support the orchestration of network resources and SDN applications to mitigate network attacks.
- SDN-based security services are required to provide an application interface allowing the management of access control policies in an autonomous and prompt manner.
- SDN-based security services are required to provide a resource-control interface for the control of network resources to mitigate network attacks.
- SDN-based security services are required to provide the logically centralized control of network resources to mitigate network attacks.
7. Use Cases

This section introduces two use cases for security services based on SDN: (i) centralized firewall system and (ii) centralized DDoS-attack mitigation system.

7.1. Centralized Firewall System

For the centralized firewall system, a centralized network firewall can manage each network resource and firewall rules can be managed flexibly by a centralized server for firewall (called Firewall). The centralized network firewall controls each switch for the network resource management and the firewall rules can be added or deleted dynamically.

The procedure of firewall operations in the centralized firewall system is as follows:

1. Switch forwards an unknown flow’s packet to SDN Controller.
2. SDN Controller forwards the unknown flow’s packet to an appropriate security service application, such as Firewall.
3. Firewall analyzes the headers and contents of the packet.
4. If Firewall regards the packet as a malware’s packet with a suspicious pattern, it reports the malware’s packet to SDN Controller.
5. SDN Controller installs new rules (e.g., drop packets with the suspicious pattern) into switches.
6. The malware’s packets are dropped by switches.

For the above centralized firewall system, the existing SDN protocols can be used through standard interfaces between the firewall application and switches [RFC7149][ITU-T.Y.3300] [ONF-SDN-Architecture][ONF-OpenFlow].

Legacy firewalls have some challenges such as the expensive cost, performance, management of access control, establishment of policy, and packet-based access mechanism. The proposed framework can resolve these challenges through the above centralized firewall system based on SDN as follows.

- Cost: The cost of adding firewalls to network resources such as routers, gateways, and switches is substantial due to the reason that we need to add firewall on each network resource. To solve
this, each network resource can be managed centrally such that a single firewall is manipulated by a centralized server.

- **Performance:** The performance of firewalls is often slower than the link speed of network interfaces. Every network resource for firewall needs to check firewall rules according to network conditions. Firewalls can be adaptively deployed among network switches, depending on network conditions in the framework.

- **The management of access control:** Since there may be hundreds of network resources in an administered network, the dynamic management of access control for security services like firewall is a challenge. In the framework, firewall rules can be dynamically added for new malware.

- **The establishment of policy:** Policy should be established for each network resource. However, it is difficult to describe what flows are permitted or denied for firewall within a specific organization network under management. Thus, a centralized view is helpful to determine security policies for such a network.

- **Packet-based access mechanism:** Packet-based access mechanism is not enough for firewall in practice since the basic unit of access control is usually users or applications. Therefore, application level rules can be defined and added to the firewall system through the centralized server.

### 7.2. Centralized DDoS-attack Mitigation System

For the centralized DDoS-attack mitigation system, a centralized DDoS-attack mitigation can manage each network resource and manipulate rules to each switch through a centralized server for DDoS-attack mitigation (called DDoS-attack Mitigator). The centralized DDoS-attack mitigation system defends servers against DDoS attacks outside private network, that is, from public network.

Servers are categorized into stateless servers (e.g., DNS servers) and stateful servers (e.g., web servers). For DDoS-attack mitigation, traffic flows in switches are dynamically configured by traffic flow forwarding path management according to the category of servers [AVANT-GUARD]. Such a management should consider the load balance among the switches for the defense against DDoS attacks.

The procedure of DDoS-attack mitigation operations in the centralized DDoS-attack mitigation system is as follows:

1. **Switch** periodically reports an inter-arrival pattern of a flow’s packets to SDN Controller.
2. SDN Controller forwards the flow’s inter-arrival pattern to an appropriate security service application, such as DDoS-attack Mitigator.

3. DDoS-attack Mitigator analyzes the reported pattern for the flow.

4. If DDoS-attack Mitigator regards the pattern as a DDoS attack, it computes a packet dropping probability corresponding to suspiciousness level and reports this DDoS-attack flow to SDN Controller.

5. SDN Controller installs new rules into switches (e.g., forward packets with the suspicious inter-arrival pattern with a dropping probability).

6. The suspicious flow’s packets are randomly dropped by switches with the dropping probability.

For the above centralized DDoS-attack mitigation system, the existing SDN protocols can be used through standard interfaces between the DDoS-attack mitigator application and switches [RFC7149] [ITU-T.Y.3300] [ONF-SDN-Architecture] [ONF-OpenFlow].

The centralized DDoS-attack mitigation system has challenges similar to the centralized firewall system. The proposed framework can resolve these challenges through the above centralized DDoS-attack mitigation system based on SDN as follows.

- **Cost**: The cost of adding DDoS-attack mitigators to network resources such as routers, gateways, and switches is substantial due to the reason that we need to add DDoS-attack mitigator on each network resource. To solve this, each network resource can be managed centrally such that a single DDoS-attack mitigator is manipulated by a centralized server.

- **Performance**: The performance of DDoS-attack mitigators is often slower than the link speed of network interfaces. The checking of DDoS attacks may reduce the performance of the network interfaces. DDoS-attack mitigators can be adaptively deployed among network switches, depending on network conditions in the framework.

- **The management of network resources**: Since there may be hundreds of network resources in an administered network, the dynamic management of network resources for performance (e.g., load balancing) is a challenge for DDoS-attack mitigation. In the framework, as dynamic network resource management, traffic flow forwarding path management can handle the load balancing of network switches [AVANT-GUARD]. With this management, the current
and near-future workload can be spread among the network switches for DDoS-attack mitigation. In addition, DDoS-attack mitigation rules can be dynamically added for new DDoS attacks.

- The establishment of policy: Policy should be established for each network resource. However, it is difficult to describe what flows are permitted or denied for new DDoS-attacks (e.g., DNS reflection attack) within a specific organization network under management. Thus, a centralized view is helpful to determine security policies for such a network.

8. Security Considerations

This document shares all the security issues of SDN that are specified in the "Security Considerations" section of [ITU-T.Y.3300].

9. Acknowledgements

This work was partly supported by the ICT R&D program of MSIP/IITP [10041244, SmartTV 2.0 Software Platform] and ETRI.

This document has greatly benefited from inputs by Jinyong Kim, Mahdi Daghmehchir-Firoozjaei, and Geumhwan Cho.

10. References

10.1. Normative References


10.2. Informative References


Authors' Addresses

Jaehoon Paul Jeong
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 440-746
Republic of Korea

Phone: +82 31 299 4957
Fax: +82 31 290 7996
EMail: pauljeong@skku.edu
URI: http://cpslab.skku.edu/people-jaehoon-jeong.php
Hyoungshick Kim  
Department of Software  
Sungkyunkwan University  
2066 Seobu-Ro, Jangan-Gu  
Suwon, Gyeonggi-Do 440-746  
Republic of Korea  
Phone: +82 31 299 4324  
Fax: +82 31 290 7996  
EMail: hyoung@skku.edu  
URI: http://seclab.skku.edu/people/hyoungshick-kim/  

Jung-Soo Park  
Electronics and Telecommunications Research Institute  
218 Gajeong-Ro, Yuseong-Gu  
Daejeon, 305-700  
Republic of Korea  
Phone: +82 42 860 6514  
EMail: pjs@etri.re.kr
Network Working Group
Internet Draft
Intended status: Informational
Expires: April 2016

E. Lopez
Fortinet
D. Lopez
Telefonica
L. Dunbar
J. Strassner
Huawei
X. Zhuang
China Mobile
J. Parrott
BT
R Krishnan
Dell
S. Durbha
CableLabs

October 19, 2015

Framework for Interface to Network Security Functions
draft-merged-i2nsf-framework-04.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

Abstract

This document defines a set of abstractions for guiding the functionality provided by I2NSF. In the design of interfaces to allow for the provisioning of network security functions (NSFs), a critical consideration is to prevent the creation of implied constraints on NSF capability and functionality.

This document makes the recommendation that such interfaces be designed from the paradigm of processing packets and flows on the network. NSFs ultimately are packet-processing engines that inspect packets traversing networks, either directly or in the context of sessions in which the packet is associated.
1. Introduction

This document describes the framework for the Interface to Network Security Functions (I2NSF), and defines a reference model along with functional components for I2NSF. It also describes how I2NSF facilitates Software-defined network (SDN) and Network Function Virtualization (NVF) control, while avoiding potential constraints that could limit NSFs internal functionality and capability.

The I2NSF use cases ([I2NSF-ACCESS], [I2NSF-DC] and [I2NSF-Mobile]) call for standard interfaces for clients (e.g., applications, application controllers, or users), to inform the network what they are willing to receive, in other words, the security rules for their specific traffic. It also provides a standard interface for them to monitor the security functions hosted and managed by service providers.

[I2NSF-Problem] describes the motivation and the problem space for Interface to Network Security Functions.
2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

BSS: Business Support System

Controller: used interchangeably with Service Provider Security Controller or management system throughout this document.

FW: Firewall

IDS: Intrusion Detection System

IPS: Intrusion Protection System

NSF: Network Security Functions, defined by [I2NSF-Problem]

OSS: Operation Support System

vNSF: refers to NSF being instantiated on Virtual Machines.

3. Interfaces to Flow-based NSFs

The emergence of SDN and NFV has resulted in the need to create application programming interfaces (APIs) in support of dynamic requests from various applications or application controllers. Flow-based NSFs [I2NSF-Problem] inspects packets in the order that they are received.

The Interface to Flow-based NSFs can be generally grouped into three types:
1) Configuration - deals with the management and configuration of the NSF device itself, such as port address configurations. Configuration deals with attributes that are relatively static.

2) Signaling - which represents logging and query functions between the NSF and external systems. Signaling API functions may also be well defined by other protocols such as SYSLOG, DOTS, etc.

3) Rules Provisioning - used to control the rules that govern how packets are treated by the NSFs. Due to the need of applications/controllers to dynamically control what traffic they need to receive, much of the I2NSF efforts towards interface development will be in this area.

This draft proposes that a rule provisioning interface to NSFs can be developed on a packet-based paradigm. While there are many classifications of existing and emerging NSFs, a common trait shared by them is in the processing of packets based on the content (header/payload) and context (session state, authentication state, etc) of received packets.

An important concept is the fact that attackers do not have standards as to how to attack networks, so it is equally important not to constrain NSF developers to offering a limited set of security functions. In other words, the introduction of I2NSF standards should not make it easier for attackers to compromise the network. Therefore, in constructing standards for rules provisioning interfaces to NSFs, it is equally important to allow support for vendor-specific functions, to allow the introduction of NSFs that evolve to meet new threats. Proposed standards for rules provisioning interfaces to NSFs SHOULD NOT:

- Narrowly define NSF categories, or their roles when implemented within a network
- Attempt to impose functional requirements or constraints, either directly or indirectly, upon NSF developers
- Be a limited lowest-common denominator approach, where interfaces can only support a limited set of standardized functions, without allowing for vendor-specific functions
- Be seen as endorsing a best-common-practice for the implementation of NSFs

By using a packet-based approach to the design of such provisioning interfaces, the goal is to create a workable interface to NSFs that aids in their integration within legacy, SDN, and/or NFV environments, while avoiding potential constraints which could limit their functional capabilities.

Even though security functions come in a variety of form factors and have different features, provisioning to Flow-based NSFs can be categorized by

- Subject-Match values, based on packet data, packet header, or packet payload, which can be one or more header fields or bits in the packets, or the various combination of them;
- Object-Match values, based on context (e.g., state, direction of the traffic, time, geo-location, etc.);
- Action-Egress processing, such as invoke signaling, packet forwarding and/or transformation, SDN/NFV integration;
- Functional Profile - a functional profile is a specific organization of characteristics and/or behavior of that define the functionality offered by an entity (e.g., IPS:<Profile>, signature file, Anti-virus file, URL filtering file, etc.). Integrated and one-pass checks on the content of packets are examples of a functional profile.

The functional profile or signature file is one of the key properties that determine the effectiveness of the NSF, and is mostly vendor-specific today.

4. Reference Models in Managing NSFs

This document only focuses on the framework of rules provisioning and monitoring of flow-based NSFs.

The following figure shows various interfaces for managing the provisioning & monitoring aspects of flow-based NSFs.
4.1. NSF Facing (Capability Layer) Interface

This is the interface between the Service Provider’s management system (or Security Controller) and the NSFs that are selected to enforce the desired network security. This interface is called the Capability Interface in the I2NSF context.

4.2. Client Facing (Service Layer) Interface

This interface is for clients or Application Controller to express and monitor security policies for their specific flows. The Client Facing interface is called the Server Layer Interface in the I2NSF context.
context. The I2NSF Service Layer allows the client to define and monitor the client specific policies and their execution status.

A single client layer policy may need multiple NSFs or NSF instantiations that are used collectively to achieve the desired enforcement.

4.3. Vendor Facing Interface

When service providers have multiple types of security functions provided by different vendors, it is necessary to have an interface for vendors to register their NSFs indicating the capabilities of their NSFs.

The Registration Interface can be defined statically or instantiated dynamically at runtime. If new functionality that is exposed to the user is added to an NSF, then the vendor MUST notify the Service Provider management system of its updated interface.

4.4. The network connecting the Security Controller and NSFs

Most likely, the NSFs are not directly attached to the Security Controller; for example, NSFs can be distributed across the network. The network that connects the Security Controller with the NSFs can be the same network that carries the data traffic, or can be a dedicated network for management purposes only. In either case, packet loss could happen due to failure, congestion, or other reasons.

Therefore, the transport mechanism used to carry the control messages and monitoring information should provide reliable message delivery. Transport redundancy mechanisms such as Multipath TCP (MPTCP) [MPTCP] and the Stream Control Transmission Protocol (SCTP) [RFC3286] will need to be evaluated for applicability. Latency requirements for control message delivery must also be evaluated.

The connection between Security Controller and NSFs could be:
- Closed environments, where there is only one administrative domain. Less restrictive access control and simpler validation can be used inside the domain because of the protected environment.

- Open environments, where some NSFs (virtual or physical) can be hosted in external administrative domains or reached via external network domains. This requires more restrictive security controls to be placed over the I2NSF interface. The information over the I2NSF interfaces must use trusted channels, such as TLS, SASL (RFC4422), or the combination of the two.

Over the Open Environment, I2NSF needs to provide identity information, along with additional data that Authentication, Authorization, and Accounting (AAA) frameworks can use. This enables those frameworks to perform AAA functions on the I2NSF traffic.

4.5. Interface to vNSFs

Even though there is no difference between virtual network security functions (vNSF) and physical NSFs from the policy provisioning perspective, there are some unique characteristics in interfacing to the vNSFs:

- There could be multiple instantiations of one single NSF being distributed across a network. When different instantiations are visible to the Security Controller, different policies may be applied to different instantiations of one single NSF (e.g., to reflect the different roles that each vNSF is designated for).
- When multiple instantiations of one single NSF appear as one single entity to the Security Controller, the policy provisioning has to be sent to the NSF’s sub-controller, which in turn disseminates the polices to the corresponding instantiations of the NSF, as shown in the Figure 2 below.
- Policies to one vNSF may need to be retrieved and moved to another vNSF of the same type when client flows are moved from one vNSF to another.
- Multiple vNSFs may share the same physical platform
- There may be scenarios where multiple vNSFs collectively perform the security policies needed.

```
+------------------------+
| Security Controller    |
+------------------------+

| NSF-A +--------------+ | NSF-B +--------------+
|       |Sub Controller|       |       |sub Controller|
|       +--------------+       |       +--------------+
|       | NSF-A#1 | ... | NSF-A#n|       | NSF-B#1 | ... | NSF-B#m|
|       |       NSF-A cluster |       |       NSF-B cluster |
```

Figure 2: Cluster of NSF Instantiations Management

5. Flow-based NSF Capability Characterization

There are many types of flow-based NSFs. Firewall, IPS, and IDS are the commonly deployed flow-based NSFs. However, the differences among them are definitely blurring, due to technological capacity increases, integration of platforms, and new threats. At their core:

- Firewall - A device or a function that analyzes packet headers and enforces policy based on protocol type, source address, destination address, source port, destination port, and/or other attributes of the packet header). Packets that do not match policy are rejected. Note that additional functions, such as logging and notification of a system administrator, could optionally be enforced as well.

- IDS (Intrusion Detection System) - A device or function that analyzes whole packets, both header and payload, looking for known events. When a known event is detected, a log message is generated detailing the event. Note that additional functions, such as
notification of a system administrator, could optionally be enforced as well.

IPPS (Intrusion Prevention System) - A device or function that analyzes whole packets, both header and payload, looking for known events. When a known event is detected the packet is rejected. Note that additional functions, such as logging and notification of a system administrator, could optionally be enforced as well.

To prevent constraints on NSF vendors’ creativity and innovation, this document recommends the Flow-based NSF interfaces to be designed from the paradigm of processing packets on the network. Flow-based NSFs ultimately are packet-processing engines that inspect packets traversing networks, either directly or in the context of sessions in which the packet is associated.

Flow-based NSFs differ in the depth of packet header or payload they can inspect, the various session/context states they can maintain, and the specific profiles and the actions they can apply. An example of a session is "allowing outbound connection requests and only allowing return traffic from the external network".

Accordingly, the NSF capabilities are characterized by the level of packet processing and context that a NSF supports, the profiles and the actions that the NSF can apply. The term "context" includes anything that can influence the action(s) taken by the NSF, such as time of day, location, session state, and events.

Vendors can register their NSFs using the Subject-Object-Action-Function categories described in Section 2, with detailed specification of each category as shown in the table below:

<table>
<thead>
<tr>
<th>Layer 2 Header</th>
<th>Layer 2 header fields: Source/Destination/s-VID/c-VID/EtherType/.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 3</td>
<td>Layer header fields:</td>
</tr>
<tr>
<td>IPv4 Header</td>
<td>protocol</td>
</tr>
<tr>
<td></td>
<td>dest port</td>
</tr>
<tr>
<td></td>
<td>src port</td>
</tr>
<tr>
<td></td>
<td>src address</td>
</tr>
<tr>
<td></td>
<td>dest address</td>
</tr>
<tr>
<td></td>
<td>dscp</td>
</tr>
<tr>
<td></td>
<td>length</td>
</tr>
<tr>
<td></td>
<td>flags</td>
</tr>
<tr>
<td></td>
<td>ttl</td>
</tr>
<tr>
<td>IPv6 Header</td>
<td>addr</td>
</tr>
<tr>
<td></td>
<td>protocol/nh</td>
</tr>
<tr>
<td></td>
<td>src port</td>
</tr>
<tr>
<td></td>
<td>dest port</td>
</tr>
<tr>
<td></td>
<td>src address</td>
</tr>
<tr>
<td></td>
<td>dest address</td>
</tr>
<tr>
<td></td>
<td>length</td>
</tr>
<tr>
<td></td>
<td>traffic class</td>
</tr>
<tr>
<td></td>
<td>hop limit</td>
</tr>
<tr>
<td></td>
<td>flow label</td>
</tr>
<tr>
<td></td>
<td>dscp</td>
</tr>
<tr>
<td>TCP</td>
<td>Port</td>
</tr>
<tr>
<td>SCTP</td>
<td>syn</td>
</tr>
<tr>
<td>DCCP</td>
<td>ack</td>
</tr>
<tr>
<td></td>
<td>fin</td>
</tr>
<tr>
<td></td>
<td>rst</td>
</tr>
<tr>
<td></td>
<td>? psh</td>
</tr>
<tr>
<td></td>
<td>? urg</td>
</tr>
<tr>
<td></td>
<td>? window</td>
</tr>
<tr>
<td></td>
<td>sockstress</td>
</tr>
<tr>
<td>Note: bitmap could be used to represent all the fields</td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td>flood abuse</td>
</tr>
<tr>
<td></td>
<td>fragment abuse</td>
</tr>
<tr>
<td></td>
<td>Port</td>
</tr>
<tr>
<td>HTTP layer</td>
<td>hash collision</td>
</tr>
<tr>
<td></td>
<td>http - get flood</td>
</tr>
<tr>
<td></td>
<td>http - post flood</td>
</tr>
<tr>
<td></td>
<td>http - random/invalid url</td>
</tr>
<tr>
<td></td>
<td>http - slowloris</td>
</tr>
<tr>
<td></td>
<td>http - slow read</td>
</tr>
<tr>
<td></td>
<td>http - r-u-dead-yet (rudy)</td>
</tr>
<tr>
<td></td>
<td>http - malformed request</td>
</tr>
<tr>
<td></td>
<td>http - xss</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>https - ssl session exhaustion</th>
</tr>
</thead>
<tbody>
<tr>
<td>IETF PCP</td>
<td>Configurable Ports</td>
</tr>
<tr>
<td>IETF TRAM</td>
<td>profile</td>
</tr>
</tbody>
</table>

Table 1: Subject Capability Index

<table>
<thead>
<tr>
<th>Object (context) matching Capability Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session</td>
</tr>
<tr>
<td>Session state,</td>
</tr>
<tr>
<td>bidirectional state</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>time span</td>
</tr>
<tr>
<td>time occurrence</td>
</tr>
<tr>
<td>Events</td>
</tr>
<tr>
<td>Event URL, variables</td>
</tr>
<tr>
<td>Location</td>
</tr>
<tr>
<td>Text string, GPS coords, URL</td>
</tr>
<tr>
<td>Connection Type</td>
</tr>
<tr>
<td>Internet (unsecured), Internet (secured by VPN, etc.), Intranet, ...</td>
</tr>
<tr>
<td>Direction</td>
</tr>
<tr>
<td>Inbound, Outbound</td>
</tr>
<tr>
<td>State</td>
</tr>
<tr>
<td>Authentication State</td>
</tr>
<tr>
<td>Authorization State</td>
</tr>
<tr>
<td>Accounting State</td>
</tr>
<tr>
<td>Session State</td>
</tr>
</tbody>
</table>

Table 2: Object Capability Index

<table>
<thead>
<tr>
<th>Action Capability Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress port</td>
</tr>
<tr>
<td>SFC header termination,</td>
</tr>
<tr>
<td>VxLAN header termination</td>
</tr>
<tr>
<td>Actions</td>
</tr>
<tr>
<td>Pass</td>
</tr>
<tr>
<td>Deny</td>
</tr>
<tr>
<td>Mirror</td>
</tr>
</tbody>
</table>

Table 3: Action Capability Index

<table>
<thead>
<tr>
<th>Egress</th>
<th>Encap SFC, VxLAN, or other header</th>
</tr>
</thead>
</table>

Table 4: Function Capability Index

<table>
<thead>
<tr>
<th>Functional profile Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile types</td>
</tr>
<tr>
<td>Signature</td>
</tr>
</tbody>
</table>

6. Structure of Rules for governing NSFs

6.1. Capability Layer Rules and Monitoring

The purpose of the Capability Layer is to define explicit rules for individual NSFs to treat packets, as well as methods to monitor the execution status of those functions.

[ACL-MODEL] has defined rules for the Access Control List supported by most routers/switches that forward packets based on packets’ L2, L3, or sometimes L4 headers. The actions for Access Control Lists include Pass, Drop, or Redirect.

The functional profiles (or signatures) for NSFs are not present in [ACL-MODEL] because the functional profiles are unique to specific NSFs. For example, most vendors’ IPS/IDS have their proprietary functions/profiles. One of the goals of I2NSF is to define a common envelop format for exchanging or sharing profiles among different organizations to achieve more effective protection against threats.

The "subject" of the I2NSF policies should not only include the matching criteria specified by [ACL-MODEL] but also the L4-L7 fields depending on the NSFs selected.

The I2NSF Capability Layer has to specify the "Object" (i.e. the context surrounding the packets).
The I2NSF "actions" should extend the actions specified by [ACL-MODEL] to include applying statistics functions that clients provide.

The rules for Flow-Based NSF can be extended from the Policy Core Information Model [RFC3060] and Policy Core Information Model Extension [RFC3460] which is the base for ITU-T X.1036 [ITU-T-X1036]:

```
+-------------------------+
| Capability-Layer-Rules  |
+-------------------------+
          |                  |
+--------+                  |
| Compound Condition |      |
| action |        |       |
+--------+                  |
          |                  |
+--------+                  |
| simple conditions |      |
+--------+                  |
          |                  |
+--------+                  |
| Logical OR: || |
+--------+                  |
          |                  |
+--------+                  |
| Logical NOT: ! |
+--------+                  |
          |                  |
+--------+                  |
| Logical AND: && |
+--------+                  |
          |                  |
+--------+                  |
| Profile |
+--------+                  |
          |                  |
+--------+                  |
| Time |
+--------+                  |
          |                  |
+--------+                  |
| States |
+--------+                  |
          |                  |
+--------+                  |
| Port |
+--------+                  |
          |                  |
+--------+                  |
| IPv4 Header | IPv6 Header | MAC Header | * |
+--------+                  |
          |                  |
+--------+                  |
| * |
+--------+                  |
          |                  |
+--------+                  |
```

Figure 3: Structure of Capability Layer Rules

Policy consistency among multiple security function instances is very critical because security policies are no longer maintained by one central security devices, but instead are enforced by multiple security functions instantiated at various locations.
6.2. Service Layer Policy

This layer is for clients or Application Controller to express & monitor the needed security policies for their specific flows.

Some Customers may not have security skills. As such, they are not able to express requirements or security policies that are precise enough. These customers may express expectations or intent. Customers may also express guidelines such as which certain types of destinations are not allowed for certain groups. As the result, there could be many depths or layers of Service Layer policies. Here are some examples of more abstract service layer security Policies:

- Pass for Subscriber "xxx"
- enable basic parental control
- enable "school protection control"
- allow Internet traffic from 8:30 to 20:00
- scan email for malware detection protect traffic to corporate network with integrity and confidentiality
- remove tracking data from Facebook [website = *.facebook.com]
- my son is allowed to access facebook from 18:30 to 20:00

One Service Layer Security Policy may need multiple security functions at various locations to achieve the enforcement. Service layer Security Policy may need to be updated by users or Application controller when user’s service requirements have been changed. Some service layer policies may not be granted because the carrier or Enterprises imposes additional constraints on what the user can have. [I2NSF-Demo] describes an implementation of translating a set of service layer policies to the Capability Layer instructions to NSFs.

I2NSF will first focus on simple service layer policies that are modeled as closely as possible on the Capability Layer. The I2NSF simple service layer should have similar structure as I2NSF capability layer, however with more client oriented expression for the subject, object, action, and function.

There have been several industry initiatives to address network policies, such as OpenStack’s Group-based Policy (GBP), IETF Policy...
Core Information Model-PCIM [RFC3060, RFC3460], and others. I2NSF will not work on general network service policies, but instead will define a standard interface for clients/applications to inform the Flow-based NSFs on the rules for treating traffic.

However, the notion of Groups (or roles), Target, Context (or conditions), and Action do cover what is needed for clients/applications to express the rules on how their flows can be treated by the Flow-Based NSFs in networks. The goal is to have a policy structure that can be mapped to the Capability layer’s Subject-Object-Action-Function paradigm.

Using PCIM (RFC3060, which ITU-T X.1036 was based on) as a basis is possible. However, RFC3060 was created for general network policies. This means that in some areas, it provides more than what I2NSF needs, and in other areas, it needs extension. This is especially pronounced regarding Policy Context and Policy Conditions (e.g., the direction, time, and other contextual events that govern the policies to NSFs).

The I2NSF simple service layer can have the following entities:

- Composite Groups or Roles (I2NSF-Role): This is a group of users, applications, virtual networks, or traffic patterns to which a service layer policy can be applied. An I2NSF-Role may be mapped to a client virtual Subnet (i.e. with private address prefix), a subnet with public address families, specific applications, destinations, or any combination of them with logical operators (Logical AND, OR, or NOT). An I2NSF-Role can have one or more Policy Rule Sets.
- Target. This is used by the application client to establish communications over the network. A Target can be mapped to a physical/logical ingress port, a set of destinations, or a physical/logical egress port.
- Policy Rule Set. A Policy Rule Set is used to determine how the traffic between a pair of I2NSF-Role and Target is to be treated. A Policy Rule Set consists of one or more Policy Rules.
- Policy Rule. A Policy Rule consists of a Policy Conditions and a set of Actions to be applied to the traffic.
- Policy Condition. Describes when a Policy Rule set is to be applied. It can be expressed as a direction, a list of L4 ports, time range, or a protocol, etc.
Policy Action: This is the action applied to the traffic that matches the Conditions. An action may be a simple ACL action (i.e. allow, deny, mirroring), applying a well known statistics functions (e.g. X minutes count, Y hours court), applying client specified functions (with URL provided), or may refer to an ordered sequence of functions.

Figure 4: Rule Structure for Simple Service Layer
7. Capability Negotiation

When an NSF can't perform the desired provisioning (e.g., due to resource constraints), it MUST inform the controller.

The protocol needed for this security function/capability negotiation may be somewhat correlated to the dynamic service parameter negotiation procedure [RFC7297]. The Connectivity Provisioning Profile (CPP) template documented in RFC7297, even though currently covering only Connectivity (but includes security clauses such as isolation requirements, non-via nodes, etc.), could be extended as a basis for the negotiation procedure. Likewise, the companion Connectivity Provisioning Negotiation Protocol (CPNP) could be a candidate to proceed with the negotiation procedure.

The "security as a service" would be a typical example of the kind of (CPP-based) negotiation procedures that could take place between a corporate customer and a service provider. However, more security specific parameters have to be considered.

8. Types of I2NSF clients

It is envisioned that I2NSF clients include:

- Application Controller:

  For example, Video Conference Mgr/Controller needs to dynamically inform network to allow or deny flows (some of which are encrypted) based on specific fields in the packets for a certain time span. Otherwise, some flows can't go through the NSFs (e.g. FW/IPS/IDS) in the network because the payload is encrypted or packets' protocol codes are not recognized by those NSFs.

- Security Administrators

- Enterprise
- Operator Management System dynamically updates, monitors and verifies the security policies to NSFs (by different vendors) in a network.
- Third party system

- Security functions send requests for more sophisticated functions upon detecting something suspicious, usually via a security controller.

9. Manageability Considerations

Management of NSFs usually includes
- life cycle management and resource management of vNSFs
- configuration of devices, such as address configuration, device internal attributes configuration, etc,
- signaling, and
- policy rules provisioning.

I2NSF will only focus on the policy rule provisioning part, i.e., the last bullet listed above.

10. Security Considerations

Having a secure access to control and monitor NSFs is crucial for hosted security service. Therefore, proper secure communication channels have to be carefully specified for carrying the controlling and monitoring information between the NSFs and their management entity (or entities).

11. IANA Considerations

This document requires no IANA actions. RFC Editor: Please remove this section before publication.
12. References

12.1. Normative References


12.2. Informative References


13. Acknowledgments

Acknowledgements to xxx for his review and contributions.

This document was prepared using 2-Word-v2.0.template.dot.
Authors’ Addresses

Edward Lopez
Fortinet
899 Kifer Road
Sunnyvale, CA 94086
Phone: +1 703 220 0988
Email: elopez@fortinet.com

Diego Lopez
Telefonica
Email: diego.r.lopez@telefonica.com

XiaoJun Zhuang
China Mobile
Email: zhuangxiaojun@chinamobile.com

Linda Dunbar
Huawei
Email: Linda.Dunbar@huawei.com

John Strassner
Huawei
Email: John.sc.Strassner@huawei.com

Joe Parrott
BT
Email: joe.parrott@bt.com

Ramki Krishnan
Dell
Email: ramki_krishnan@dell.com

Seetharama Rao Durbha
CableLabs
Email: S.Durbha@cablelabs.com
Remote Attestation Procedures for virtualized NSFs (vNSFs) through the I2NSF Security Controller

draft-pastor-i2nsf-vnsf-attestation-00

Abstract

This document describes the procedures a user can follow to assess the trust on a virtualized NSF and its user-defined configuration through the I2NSF Security Controller. The procedure to assess trustworthiness is based on a remote attestation between the user and the vNSF.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 20, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
2. User Expectations about Trust ................................. 4
   2.1. First Step: User-Agnostic Attestation .................. 4
   2.2. Second Step: User-Specific Attestation ................ 4
3. Application of Trusted Computing .............................. 5
   3.1. Applying Trusted Computing .............................. 8
4. Trusted vNSF Platforms .......................................... 9
   4.1. Requirements for a Trusted vNSF Platform ............... 9
      4.1.1. Trusted Boot ........................................ 9
      4.1.2. Remote Attestation Service ......................... 10
      4.1.3. Secure Boot ......................................... 10
4. Remote Attestation Procedures ................................ 10
5. Trusted Channel with the Security Controller ................ 11
6. Security Considerations ....................................... 13
7. IANA Considerations ........................................... 14
8. References .................................................... 14
   8.1. Normative References .................................... 14
   8.2. Informative References ................................. 14
8. Authors’ Addresses ........................................... 14
1. Introduction

As described in [I-D.pastor-i2nsf-merged-use-cases], when virtualization is applied to the NSF environment (vNSF) it implies several additional concerns in security. The most relevant threats associated with a security virtual platform are:

- An unknown/unauthorized user can try to impersonate another user that can legitimately access virtualized NSF services. This attack may lead to accessing the policies and applications of the attacked user or to generate network traffic outside a the security functions with a falsified identity.

- An authorized user may misuse assigned privileges to alter the network traffic processing of other users in the virtualization platform. This can become especially serious when such a user has administration privileges granted by the virtualization provider, the ISP or the local network operator.

- A user may try to install malformed elements (policy or application), trying to directly take the control of a NSF or virtualization platform (for example by exploiting a vulnerability on one of the functions or may try to intercept or modify the traffic of other users in the same platform.

- A malicious virtualization provider can modify the software running on it (the operating system or a concrete vNSF) to alter the behaviour of the latter. This event has a high impact on all users accessing vNSFs as the virtualization provider has the highest level of privilege on the software in execution.

- A user with physical access to the virtualization platform can modify the behavior of hardware components, or the components themselves. Furthermore, it can access a serial console (most devices offer this interface for maintenance reasons) to access the NSF software with the same level of privilege of the virtualization provider.

Mutual authentication and, what is more important, the attestation of the virtualization platform and the vNSFs by users could address these threats to an acceptable level of risk.

The user can have a proof that their vNSFs and policies are correctly (from the user point of view) enforced by the Security Controller. Taking into account the threat identified above, this document first identifies the user expectations regarding remote trust establishment, briefly analyzes Trusted Computing techniques, and finally describes the proposed mechanisms for remote establishment of
trust through the Security Controller.

2. User Expectations about Trust

From a high-level standpoint, in a virtualized I2NSF platform, the user connects and authenticates to the Security Controller, which then initialises the user’s vNSFs and policies. Afterwards, the user traffic reaches the Internet via the virtualized platform which hosts the user’s vNSFs. The user’s expectations of the platform behavior are thus twofold:

- The user traffic will be treated according to the user-specified vNSFs and policies, and no other processing will be performed by the Security Controller or the platform itself (e.g. traffic eavesdropping).
- Each vNSF (and its corresponding policies) behaves as configured by the user.

We will refer to the attestation of these two expectations as the "user-agnostic attestation" and the "user-specific attestation".

2.1. First Step: User-Agnostic Attestation

This is the first interaction between a user and a Security Controller: the user wants to attest that he is connected to a genuine Security Controller before he continues with the authentication. In this context, two properties characterise the genuineness of the Security Controller:

1. That the identity of the Security Controller is correct
2. That it will process the user’s credentials and set up the user vNSFs and policies properly.

Once these two properties are proven to the user, the user knows that their credentials will only be used by the Security Controller to set up the execution platform for their vNSFs.

2.2. Second Step: User-Specific Attestation

From the security enforcement point of view, the user agnostic attestation focuses on the initialization of the execution platform for the vNSFs. This second step aims to prove to the user that their security is enforced accordingly with their choices (i.e. vNSFs and policies). The attestation can be performed at the initialization of the vNSFs, before the user traffic is processed by the vNSFs, or
during the execution of the vNSFs.

Support of static vNSF attestation is REQUIRED for a Security Controller managing vNSFs, and MUST be performed before the user traffic is redirected through any set of vNSFs. The Security Controller MUST provide a proof to the user that the instantiated vNSFs and policies are the ones chosen.

Additionally to the vNSFs instantiation attestation, a continuous attestation of the vNSFs execution MAY be required by a user to ensure their security.

3. Application of Trusted Computing

In a nutshell, Trusted Computing (TC) aims at answering the following question: "As a user or administrator, how can I have some assurance that a computing system is behaving as it should?". The major enterprise level TC initiative is the Trusted Computing Group [TCG], which has been established for more than a decade, that primarily focuses on developing TC for commodity computers (servers, desktops, laptops, etc.).

The overall scheme proposed by TCG for using Trusted Computing is based on a step-by-step extension of trust, called a Chain of Trust. It uses a transititive mechanism: if a user can trust the first execution step and each step correctly attests the next executable software for trustworthiness, then a user can trust the system.
Figure 1: Applying Trusted Computing

Effectively, during the loading of each piece of software, the integrity of each piece of software is measured and stored inside a log that reflects the different boot stages, as illustrated in the figure above. Later, at the request of a user, the platform can present this log (signed with the unique identity of the platform) to the user, which can be checked by the user to prove the platform identity and attest the state of the system. The base case for the extension of the Chain of Trust is called the Core Root of Trust. This Core Root of Trust (CRTM) encompasses the minimal combination of hardware and software elements that a remote verifier needs to trust in order to validate the entire Chain of Trust.
From the Chain of Trust extension point of view, any component that needs to be attested is considered an adversary and must not be able to modify its behaviour measurement. It is then necessary the use of hardware mechanisms in combination with software components to create a strong unforgeable identity and provide safer storage of secrets. Therefore the main components of the Core Root of Trust are:

1. A specialized hardware component to store the log and measurements away from the software access

2. An initial isolated component that is able to measure the first non-trusted software, which will be then trusted to measure the next stage software.

The TCG has created a standard for the design and usage of a secure cryptoprocessor to address the storage of keys, general secrets, identities and platform integrity measurements: the Trusted Platform Module (TPM). Since the TPM is located outside of the execution path it cannot measure it, thus the need of a second trusted component to do the measurements and send them to the TPM. The initial straightforward solution was to trust the first to-be-executed block of software, using extensive auditing or formal verification for example; but this is still an implicit trust since nothing guarantees the user that it is the actual executed code. The CRTM can be strengthened by either an out-of-band discrete component or through in-band enforcement.

When using a TPM as a root of trust, measurements of the software stack are stored in special on-board Platform Configuration Registers (PCRs) on a discrete TPM. There are normally a small number of PCRs that can be used for storing measurements, however it is not possible to directly write to a PCR; instead measurements must be stored using a process called Extending PCRs.

The extend operation can update a PCR by producing a global hash of the concatenated values of the previous PCR value with the new measurement value, for example: $\text{PCR} = \text{SHA-1} (\text{PCR} \mid \text{measurement})$

The Extend operation allows for an unlimited number of measurements to be captured in a single PCR, since the size of the value is always the same and it retains a verifiable ordered chain of all the previous measurements. Furthermore, it is computationally infeasible for an attacker to calculate two different hashes that will match the same resulting value of an PCR extend operation.
3.1. Applying Trusted Computing

Attestation of the virtualization platform will thus rely on a process of measuring the booted software and storing a chained log of measurements, typically referred to as Trusted Boot. The user must be able to send a nonced query of the state of a platform to the Root of Trust for a signed set of platform measurements. The user will either validate the signed set of measurements with a trusted third party verifier who will assess whether the software configuration is trusted, or the user can check for themselves against their own set of reference digest values (measurements) that they have obtained a priori, and having already known the public endorsement key of the remote Root of Trust. Optionally, to preserve the privacy of a platform identity, an Attestation Identity Key (AIK) can be generated and used by the Root of Trust for the purpose of attestation instead of the Public Endorsement Key. This must be done in conjunction with a third party Privacy Certificate Authority (CA) who is trusted to not reveal the real identities, but to act as an intermediary.

As well as for providing a signed audit log of boot measurements, the PCR values can also be used as an identity for dynamically decrypting encrypted blobs on the platform (such as encryption keys or configurations that belong to operating system components). Software can choose to submit pieces of data to be encrypted by the Root of Trust (which has its own private asymmetric key and PCR registers) and only have it decrypted based on a criteria. This criteria can be that the platform booted into a particular state (e.g. a set of PCR values). Once the desired criteria is described and the sensitive data is encrypted by the root of trust, the data has been sealed to that platform state. The sealed data will only be decrypted when the platform measurements held in the the root of trust match the particular state.

As described above, Trusted Boot requires the use of a root of trust for safely storing measurements and secrets. Since the Root of Trust is self-contained and isolated from all the software that is measured, it is able to produce a signed set of platform measurements to a local or remote user. Trusted Boot however does not provide enforcement of a configuration, since the root of trust is a passive component not in the execution path, and is solely used for safe independent storage of secrets and platform measurements. It will respond to attestation requests with the exact measurements that were made during the software boot process. Sealing and unsealing of sensitive data is also a strong advantage of Trusted Boot, since it prevents leakage of secrets in the event of an untrusted software configuration.

Trusted Boot should not be confused with a different mechanism known
as "Secure Boot", as they both are designed to solve different problems. Secure Boot is a mechanism for a platform owner to lock a platform to only execute particular software. Software components that do not match the configuration digests will not be loaded or executed. This mechanism is particularly useful in preventing bootkits from successfully infecting a platform on reboot. A common standard for implementing Secure Boot is described in [UEFI]. Secure Boot only enforces a particular configuration of software, it does not allow a user to attest or quote for a series of measurements.

4. Trusted vNSF Platforms

In the I2NSF environment users directly interact with the Security Controller, which will become the essential element to implement the measurements described in the procedures described above, relaying on a TPM for the Root of Trust.

4.1. Requirements for a Trusted vNSF Platform

Although a discrete hardware TPM is RECOMMENDED, relaxed alternatives (such as embedded CPU TPMs, or memory and execution isolation mechanisms) MAY also be applied when the required level of assurance is lower. This reduced level of assurance MUST be communicated to the user by the Security Controller during the initial mutual authentication phase.

4.1.1. Trusted Boot

All users who interact with a Security Controller MUST be able to:

a. Identify the Security Controller based on the public key of a Root of Trust.

b. Retrieve a set of measurements of all the base software the Security Controller has booted (i.e. the vNSF platform).

This requires that firmware and software MUST be measured before loading, with the resulting value being used to extend the appropriate PCR register. The following list describes which PCR registers SHOULD be used during a Trusted Boot process:

- PCRs 00-03: for use by the CRTM (Initial EEPROM or PC BIOS)
- PCRs 04-07: for use by the bootloader stages
- PCRs 08-15: for use by the booted base system
4.1.2. Remote Attestation Service

A service MUST be present for providing signed measurements from the RoT to the end user.

NOTE: The details for the remote attestation protocol have to be defined.

4.1.3. Secure Boot

Using a mechanism such as Secure Boot helps provide strong prevention of software attacks. Furthermore, in combination with a hardware-based TPM, Secure Boot can provide some resilience to physical attacks (e.g. preventing a class of offline attacks and unauthorised system replacement). For vNSF platform providers, it is RECOMMENDED that Secure Boot is employed wherever possible with an appropriate firmware update mechanism, due to the possible threat of software/firmware modifications in either public places or privately with inside attackers.

5. Remote Attestation Procedures

The establishment of trust with the Security Controller and the vNSF platform consists of three main phases, which need to be coordinated by the user through a trusted application executed by a trusted network-attached device:

1. Trusted channel with the Security Controller. During this phase, the user securely connects to the Security Controller to avoid that user data can be tampered with or modified by an attacker if the network cannot be considered trusted. The establishment of the trusted channel is completed after the next step.

2. Security Controller attestation. During this phase, the user verifies that the Security Controller components responsible for handling the user’s credentials and for the isolation with respect to other potential users are behaving correctly. Furthermore, it is verified that the identity of the platform attested is the same of the one presented by the Security Controller during the establishment of the secure connection.

3. Platform attestation. During this step, that can be repeated periodically until the user connection is terminated, the Security Controller verifies the integrity of the elements composing the user vNSF platform. The components responsible for this task have been already attested during the previous phase.
5.1. Trusted Channel with the Security Controller

A trusted channel is an enhanced version of the secured channel that, differently from the latter, requires the integrity verification of the contacted endpoint by the other peer during the initial handshake. However, simply transmitting the integrity measurements over the channel does not guarantee that the platform verified is the channel endpoint. The public key or the certificate for the secure communication MUST be included as part of the measurements presented by the contacted endpoint during the remote attestation. This way, a malicious platform cannot relay the attestation to another platform as its certificate will not be present in the measurements list of the genuine platform.

In addition, the problem of a potential loss of control of the private key must be addressed (a malicious endpoint could prove the identity of the genuine endpoint). This is done by defining a long-lived Platform Property Certificate. Since this certificate connects the platform identity to the AIK public key, an attacker cannot use a stolen private key without revealing his identity, as it may use the certificate of the genuine endpoint but cannot create a quote with the AIK of the other platform.

Finally, since the platform identity can be verified from the
Platform Property Certificate, the information in the certificate to be presented during the establishment of a secure communication are redundant. This allows for the use of self-signed certificates, what would simplify operational procedures in virtualized environments, especially when they are multi-tenant. Thus, in place of certificates signed by trusted CAs, the use of self-signed certificates (which still need to be included in the measurements list) is RECOMMENDED.

The steps required for the establishment of a trusted channel with the Security Controller are as follows:

1. The user application begins the trusted channel handshake with the selected Security Controller.

2. The certificate of the Security Controller is collected and used for verifying the binding of the attestation result to the contacted endpoint.

3. The user application performs Security Controller attestation either locally or with the help of a Trusted Third Party.

4. If the result of the attestation is positive, the application continues the handshake and establishes the trusted channel. Otherwise, it closes the connection.

5.2. Security Controller Attestation

During the establishment of the trusted channel, the user attests the Security Controller by verifying the identity of the contacted endpoint and its integrity. Initially the Security Controller measures all the hardware and software components involved in the boot process, in order to build the chain of trust.

Since a user terminal may not have enough capabilities to perform the integrity verification of a Security Controller the user application MAY request the status of a Security Controller to a Trusted Third Party (TTP), which is in charge of communicating with it. This choice has the additional advantage of preventing an attacker from easily determining the software running at the Security Controller.

If the user application directly performs the remote attestation it performs the following steps:

1. Ask the Security Controller to generate an integrity report with the format defined in [TSCGIRSS].
2. The Security Controller retrieves the measurements and asks the TPM to sign the PCRs with an Attestation Identity Key (AIK). This signature provides the user with the evidence that the measurements received belong to the Security Controller being attested.

3. Once the integrity report has been generated it is sent back to the user application.

4. The user application first checks if the integrity report is valid by verifying the quote and the certificate associated to the AIK, and then determines if the Security Controller is behaving as expected, i.e. its software has not been compromised and isolation among the users connected to it is enforced. As part of the verification, the application also checks that the digest of the certificate, received during the trusted channel handshake, is present among measurements.

If the user application is running on a terminal with low computation resources, it may contact a TTP which, in turn, attests the Security Controller and returns the result of the integrity evaluation to the user, following the same steps depicted above.

5.3. Platform Attestation

The main outcome of the Security Controller attestation is to detect whether or not it is correctly configuring the virtualization container for the vNSFs belonging to the connecting user (the virtualization platform, or vNSF platform) in a way that the user’s traffic is processed only by the NSFs within the container. The platform attestation, instead, evaluates the integrity of the NSFs running within the platform. The steps are essentially similar to the ones described in the previous section.

Attesting vNSFs typically running as virtual machines can become a rather costly operation, especially if periodic monitoring is required by the requested level of assurance, and there are several proposals to make them feasible, from the proposal of virtual TPMs in [VTPM] to the application of Virtual Machine Introspection through an integrity monitor described by [VMIA].

6. Security Considerations

This document is specifically oriented to security and it is considered along the whole text.
7. IANA Considerations

This document requires no IANA actions.

8. References

8.1. Normative References

[I-D.pastor-i2nsf-merged-use-cases]
Pastor, A., Lopez, D., Wang, K., Zhuang, X., Qi, M.,
Zarny, M., Majee, S., Leymann, N., Dunbar, L., and M.
Georgiades, "Use Cases and Requirements for an Interface
to Network Security Functions",
draft-pastor-i2nsf-merged-use-cases-00 (work in progress),
June 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/
RFC2119, March 1997,

[TCG] "Trusted Computing Group (TCG)",
<https://www.trustedcomputinggroup.org/>.

[TSCGIRSS] "Infrastructure Work Group Integrity Report Schema
Specification, Version 1.0",
<https://www.trustedcomputinggroup.org/>.

8.2. Informative References

[UEFI] "UEFI Specification Version 2.2 (Errata D), Tech. Rep.".

[VMIA] "Verifying system integrity by proxy".

[VTPM] "vTPM:Virtualizing the Trusted Platform Module",
<https://www.usenix.org/legacy/events/sec06/tech/berger.html>.
Authors’ Addresses

Antonio Pastor  
Telefonica I+D  
Zurbaran, 12  
Madrid, 28010  
Spain  
Phone: +34 913 128 778  
Email: antonio.pastorperales@telefonica.com

Diego R. Lopez  
Telefonica I+D  
Zurbaran, 12  
Madrid, 28010  
Spain  
Phone: +34 913 129 041  
Email: diego.r.lopez@telefonica.com
Information Model of Interface to Network Security Functions
Capability Interface
draft-xia-i2nsf-capability-interface-im-04.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on April 19, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of

Internet-Draft      I2NSF Capability Interface IM         October 2015

Publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

This draft is focused on the capability interface of NSFs (Network Security Functions) and proposes its information model for controlling the various network security functions.

Table of Contents

1. Introduction ................................................ 2
2. Conventions used in this document ........................... 3
   2.1. Terminology ............................................ 3
3. Overall Analysis of Security Capability ..................... 4
   3.1. Network Security Control ............................... 5
   3.2. Content Security Control ............................... 6
   3.3. Attack Mitigation Control .............................. 8
4. Information Model Design .................................... 8
   4.1. Overall Structure ...................................... 8
   4.2. Information Model for Network Security Control ......... 9
   4.3. Information Model for Content Security Control ........ 16
   4.4. Information Model for Attack Mitigation Control ....... 17
5. IM Grammar of Security Policy .............................. 18
6. Security Considerations .................................... 21
7. IANA Considerations ........................................ 21
8. References ................................................. 21
   8.1. Normative References .................................. 21
   8.2. Informative References ................................ 22
9. Acknowledgments ............................................ 22

1. Introduction

Due to the rapid development and deployment of cloud computing services, the demand of cloud-based security services is also rapidly growing. The customers of them can be enterprises, User Equipment (UE) of mobile network, Internet of Things (IoT), and residential access users [I-D.draft-pastor-i2nsf-merged-use-cases].

From [I-D.draft-merged-i2nsf-framework], there could be two types of I2NSF interfaces:
Interface between I2NSF clients with security controller: [I-D.xia-i2nsf-service-interface-DM] describes the information model used by this type of interface. This is a service-oriented interface, the main objective is to unify the communication channel and the security service request information model between various high-level application (e.g., openstack, or various BSS/OSS) with various security controllers. The design goal of the service interface is to decouple the security service in the application layer from various kinds of security devices and their device-level security functions. A feasible model may be the intent-based information model approach derived from RBAC.

North-bound interface provided by NSFs (e.g., FW, AAA, IPS, Anti-DDOS, or Anti-Virus), no matter whether the NSFs are run in Virtual Machines (VMs) or physical appliances. In this document, this type of interface is also referred to as "capability interface". Any network entities (e.g., I2NSF client, or network/security controller) control and monitor the NSFs via this interface. Unfortunately, today’s situation is different NSF vendors have different interfaces and information models for the security functions management.

In principle, the capability interface is used by the NSFs for decoupling from the up-layer security service requests and highlighting the security capabilities they can provide. Specially, this draft is focused on the information model of controlling part of the capability interface and discusses its contents and structure. The monitoring part is not discussed in this draft.

This document is structured as following: Section 3 presents an overall analysis of security capability for I2NSF interface. Section 4 discusses the detailed structure and content of the information model. Section 4 specifies the information model of security policy in Routing Backus-Naur Form [RFC5511].

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

2.1. Terminology

AAA - Access control, Authorization, Authentication

ACL - Access Control List
3. Overall Analysis of Security Capability

At present, a variety of NSFs presented by multiple security vendors provide various security capabilities to customers. Logically, whether these security capabilities are run in VMs or physical appliance, multiple NSFs can be combined together as a whole to provide security services over the given network traffic.

Most of today’s security capabilities can fall into several categories according to their basic functionalities: network security control, content security control, attack mitigation...
control. Each category further covers more specific security capabilities, which are described below.

3.1. Network Security Control

Network security control is a category with only one security capability which has the similar goal as network firewall to some level. Its main function is inspecting and processing the network traffic traversing network based on predefined security policies.

With respect to the inspecting part, this security capability is abstractly a packet-processing engine that inspects packets traversing networks, either directly or in context to flows to which the packet is associated. Considering from the perspective of packet-processing, the implementations of it differ in the depths of packet headers and/or payloads they can inspect, the various flow and context states they can maintain, and the actions they can apply. Therefore, it can be characterized by the level of packet-processing and context that it can support, and the actions that it can apply, so does its policy design paradigm.

Here, a "Subject-Object-Action-Function" paradigm is proposed as the basis for the security policy design:

- **Subject:** Refer to the kind of information or attributes acquired directly from the packet headers or payloads that can be used in the security policy as the match conditions. It can be any fields or attributes in the packet L2/L3/L4 header, or special segment of bytes in the packet payload;

- **Object:** Refer to the context information for the packet or flow. It can be:
  - **User:** The user (or user group) information to which network flow is associated: User has many attributes such as name, id, password, type, authentication mode and so on. Name/id is often used in the security policy to identify the user. Besides, NSF is aware of the IP address of the user provided by unified user management system via network. Based on name-address association, NSF is capable of enforcing the security functions over the given user (or user group);
  - **Schedule:** Time or time range when network flow happens;
- Region: The location where network traffic is associated with. The region can be the geographic location such as country, province, and city as well. It can also be the logical network location such as IP address, network section and network domain as well;

- Target: Under the circumstances of network, it mainly refers to the service, application and device. A service is an application identified by a protocol type and port number, such as TCP, UDP, ICMP and IP. An application is a computer program for a specific task or purpose. It provides a finer granularity than service in matching traffic. The attributes that can identify a device include type (i.e., router, switch, pc, ios, or android) and the device’s owner as well;

- State: It refers to various states to which the network flow is associated. It can be either the TCP session state (i.e., new, established, related, invalid, or untracked), or the access mode of the devices (i.e., wire, wireless, or vpn).

  o Action: A variety of actions for traffic filtering or suppression can be performed, which include deny, permit, mirror, diversion, rate limiting, black/white list, QoS actions, route black hole and so on;

  o Function: Refer to the other security capabilities that can be called for more advanced treatment over the network traffic.

The above design paradigm is very general and easily extensible, and so can avoid any potential constraints which could limit the implementation of the network security control capability.

3.2. Content Security Control

Content security control is another category of security capabilities applied to application layer. Through detecting the contents carried over the traffic in application layer, these capabilities can realize various security purposes, such as defending against intrusion, inspecting virus, filtering malicious URL or junk email, blocking illegal web access or data retrieve.

Generally, each type of threat in application layer has its unique characteristic and requires to be handled with the unique method accordingly, which can be a logically independent security capability. Since there are a large number of types of threats in the application layer, as well as the new type of threat occurs quickly, there will also be a large number of security capabilities.
Therefore, some basic principles for security capabilities’ management and utilization need to be considered:

- Flexibility: each security capability should be an independent function, with minimum overlap or dependency to other capabilities. By this design, the security capabilities can be utilized and assembled together freely, and changes of one capability will not influence others;

- Generality: through a level of abstraction and consolidation, each capability should have an unified interface to make it programmable, or report its process result and some statistics information. Furthermore, it facilitates the intercommunication between multi-vendors;

- Scalability: The system must have the capability of scale up/down or scale in/out. Thus it can meet various performance requirements derived from the mutable network traffic or service request. Additionally, the security capability must support reporting its statistics information to the security controller to assist its decision on whether the capability needs to be scale up or not;

- Automation: it includes the features of auto-discovery, auto-negotiation, and automatic update. These features are especially useful for the management of a large number of NSFs.

Based on the above principles, a set of abstract and vendor-neutral capabilities with standard interface is needed. The security controller can have a common capabilities base to choose the appropriate NSFs (by different vendors), as well as customize each NSF’s detailed function by setting the interface parameters, to meet the requirements requested by clients. The security controller does not need to be aware of any detailed implementation of each capability which each vendor differs. This category of security capability is more like a black box compared with the network security control.

Furthermore, when an unknown threat (e.g., zero-day exploits, unknown malwares, and APTs) is reported by a network security device, new capability may be created, or existing capability may need to update its intelligence (e.g., signature, and algorithm), to enable the device to acquire the new capability to handle the threat. The new capability and intelligence are provided from different vendors after their analysis on the new threats, and may be sent and stored in a centralized repository or stored separately in their local...
repository. In any case, a standard interface is needed during this automation process.

3.3. Attack Mitigation Control

This category of security capabilities is specially used to detect and mitigate various types of network attacks. Today’s common network attacks are mainly classified into the following sets, and each set further consists of a number of specific attacks:

- DDoS attacks:
  - Network layer DDoS attacks: SYN flood, UDP flood, ICMP flood, IP fragment flood, etc
  - Application layer DDoS attacks: http flood, https flood, dns flood, dns amplification, ssl DDoS, etc
- Single-packet attack:
  - Scanning and sniffing attacks: IP sweep, port scanning, etc
  - Malformed packet attacks: Ping of Death, Teardrop, etc
  - Special packet attacks: Oversized ICMP, Tracert, IP timestamp option packets, etc

Each type of network attack has its own network behaviors and packet/flow characteristics. It therefore needs a special security capability for detection and mitigation.

Overall, the implementation and management of this category of security capabilities of attack mitigation control is very similar to content security control. A standard interface is essential here through which the security controller can choose and customize the given security capabilities according to specific requirements.

4. Information Model Design

4.1. Overall Structure

The I2NSF capability interface is in charge of controlling and monitoring the NSFs. Based on the analysis above, the controlling part of its information model should at least consist of three blocks: network security control, content security control and attack mitigation control. It’s noted that the capability interface only cares about the specific security capabilities rather than to
which type of device that the NSF belongs. That is, it is assume that the user of the capability interface does not care about whether the network entity it is communicating with is a firewall or an IPS. Instead, the user cares about the capability that it has, such as packet filtering or deep packet inspection. The Overall structure is illustrated in the figure below:

![Information model for capability interface](image)

Figure 1. The overall structure of information model for I2NSF Capability Interface

As illustrated in Figure 1, the network security control is the key block of the whole system. It usually runs at the first step to handle traffic (i.e., packet/flow detection and filtering, etc) over the network layer. In the extreme condition, the system can still work without this block, which means network security control is not needed under that condition.

The other two blocks are both composed of a number of specific security capabilities, which are enforced either statically according to fixed configuration or dynamically over some given traffic based on the detecting results of network security control. The two enforcement ways have difference in the information model.

The more detailed descriptions of information model for each block are given in the following sections.

4.2. Information Model for Network Security Control

The information model for network security control specifies the content and structure of a rule. A rule is complied with by the NSFs
when they handle the network traffic over network layer. Its basic structure is shown in the following figure:
As illustrated in Figure 2, at the top level, policy is a container including a set of security rules according to certain logic, i.e.,
their similarity or mutual relations, etc. The network security
policy is able to apply over both the unidirectional and
bidirectional traffic across the NSF.

Each rule represents some specific security requirements or actions
with the classic "match & action" style supported in industry widely.

The logic relation among all the conditions in one match is "AND",
which means the match result is true only when the traffic matches
all the conditions in this match.

In summary, the detailed definitions of all match conditions are as
follows:

- L2/L3/L4 Packet header: all meaningful and useful attributes in
  L2/L3/L4 packet header, for example: src/dest address, or
  src/dest port;

- Packet payload: any segment of bytes in the packet payload;
- **User**: The user is a person who is authorized to access network resources. A user can be an internet access user who accesses Internet resources or intranet resources from inside the intranet through a FW, or a remote access user who connects to a FW in VPN, or PPPoE mode to access intranet resources. The NSFs need to know the IP address or other information (i.e., user’s tenant or VN-ID) of the user to identify the user’s traffic and perform the pre-defined actions. It can also define a group of users to match and perform actions to them together;

- **Schedule**: A schedule defines time range. A rule can reference a schedule to filter traffic that passes through the NSF within the schedule. A schedule can be a periodic schedule, or a one-time schedule;

- **Region**: The location information of the user traffic. The logical definition of the location can be pre-defined in the location signature database by the geographical information, or be manually defined by the user’s IP information.

- **Target**:  
  - **Service**: A service is an application identified by a protocol type and port number. It can be a service or a group of services. The network security control matches the service traffics based on the protocol types and port numbers and applies the security actions to them;  
  - **Application**: An application is a computer program for a specific task or purpose, and multiple applications constitute an application group. It provides a finer granularity than service in matching traffic. Even if different applications have the same service, they still can be distinguished by analyzing the data packets and comparing the signatures of each application. The hierarchy category method is appropriate for identifying applications. For example, the application of Gmail belongs to the category of business systems, and the subcategory of Email. Other key attributes that belong to and can be used to identify an application are data transmission model (e.g., client-server, browser-based, networking, or peer-to-peer), risk level (e.g., Exploitable, Evasive, Data-loss, or Bandwidth-consuming);  
  - **Device**: The attributes that can identify a device include type (i.e., router, switch, pc, ios, or android) and the device’s owner as well;
o State:

- Session state: any one specific state related to the user/operation sessions, such as authentication state, TCP/UDP session state, or bidirectional state;

- Access mode: the access mode of user or device.

o Direction: the direction of the network flow.

Following table lists the possible attributes with their possible values for all the match conditions:

<table>
<thead>
<tr>
<th>Match Condition</th>
<th>Attributes: Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet Frame Header</td>
<td>Source/Destination address s-VID/c-VID/EtherType</td>
</tr>
<tr>
<td>IPv4 Packet Header</td>
<td>src/dest address protocol src/dest port length flags ttl</td>
</tr>
<tr>
<td>IPv6 Packet Header</td>
<td>src/dest address protocol/nh src/dest port length traffic class hop limit flow label</td>
</tr>
<tr>
<td>TCP SCTP DCCP</td>
<td>Port syn ack fin rst psh urg window</td>
</tr>
</tbody>
</table>
Table 1. Match conditions details

<table>
<thead>
<tr>
<th></th>
<th>sockstress</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td></td>
</tr>
<tr>
<td>Schedule</td>
<td>time span</td>
</tr>
<tr>
<td></td>
<td>days, minutes, seconds,</td>
</tr>
<tr>
<td>Region</td>
<td>country, province, city</td>
</tr>
<tr>
<td></td>
<td>IP address, network section,</td>
</tr>
<tr>
<td></td>
<td>network domain</td>
</tr>
<tr>
<td>Target</td>
<td>service: TCP, UDP, ICMP, HTTP...</td>
</tr>
<tr>
<td></td>
<td>application: Gmail, QQ, MySQL...</td>
</tr>
<tr>
<td></td>
<td>device: mobile phone, tablet, PC...</td>
</tr>
<tr>
<td>State</td>
<td>session state: new, established, related</td>
</tr>
<tr>
<td></td>
<td>invalid, untracked</td>
</tr>
<tr>
<td></td>
<td>access mode: WIFI, 802.1x, PPPOE, SSL...</td>
</tr>
<tr>
<td>Direction</td>
<td>Direction: from_client, from_server,</td>
</tr>
<tr>
<td></td>
<td>bidirection, reversed</td>
</tr>
</tbody>
</table>

Note that match conditions are extensible, new one can be created any time according to the requirements.

Network security control policy consists of a number of rules complying with the information model described above. Then it enforces the rules in turn to process the passing traffic. Following is the basic operation process:

1. The NSF compares the attributes with the match conditions defined in the first rule. If all the conditions are met, the traffic matches the rule. If one or more conditions are not met, the NSF compares the attributes with the conditions defined in the next rule. If all rules are not met, the NSF denies the traffic by default;
2. If the traffic matches a rule, the NSF performs the defined actions over the traffic. If the action is "deny", the NSF blocks the traffic. If the basic action is permit/mirror, the NSF resumes checking whether certain other security capabilities are referenced in the rule. If yes, go to step 3. If no, the traffic is permitted;

3. If other security capabilities (e.g., Antivirus or IPS) are referenced in the rule and the action defined in the rule is permit/mirror, the NSF performs the called security capabilities.

One policy or rule can be applied multiple times on different places, i.e., links, devices, networks, vpns, etc. It not only guarantees the consistent policy enforcement in the whole network, but also decreases the configuration workload.

4.3. Information Model for Content Security Control

The block for content security control is composed of a number of security capabilities, while each one aims for protecting against a specific type of threat in the application layer.

Following figure shows a basic structure of the information model:

```
+-------------------------+
| Anti-Virus              |
| Intrusion Prevention    |
| URL Filtering           |
| File Blocking           |
| Data Filtering          |
| Application Behavior Control |
| Mail Filtering          |
| ...                     |

+-------------------------+
```

Figure 3. The basic structure of information model for content security control
The detailed description about the standard interface and the parameters for all the security capabilities of this category are TBD.

4.4. Information Model for Attack Mitigation Control

The block for attack mitigation control is composed of a number of security capabilities, while each one aims for mitigating a specific type of network attack.

Following figure shows a basic structure of the information model:

![Figure 4. The basic structure of information model for attack mitigation control](image)

The detailed description about the standard interface and the general shared parameters for all the security capabilities of this category are TBD.
5. IM Grammar of Security Policy

This section specifies the information model of security policy in Routing Backus-Naur Form [RFC5511]. This grammar is intended to help the reader better understand the English text description in order to derive a data model.

Firstly, several types of route are specified as follows:
- IPv4: Match on destination IP address in the IPv4 header
- IPv6: Match on destination IP address in the IPv6 header
- MPLS: Match on a MPLS label at the top of the MPLS label stack
- MAC: Match on MAC destination addresses in the ethernet header
- Interface: Match on incoming interface of the packet

Then, the I2NSF information model grammar of security policy is specified as follows:

```
<Policy> ::= <policy-name> <policy-id> (<Rule> ...)
<Rule> ::= <rule-name> <rule-id> <Match> <Action>

<Match> ::= [subject-based-match] [object-based-match]
subject-based-match ::= [L234-packet-header] ...]
    [packet-payload] ...
L234-packet-header ::= [address-scope] [layer-2-header]
    [layer-3-header] [layer-4-header]
address-scope ::= route-type (ipv4-route | ipv6-route |
    mpls-route | mac-route | interface-route)
route-type ::= IPv4 | IPv6 | MPLS | IEEE_MAC | INTERFACE
ipv4-route ::= <ip-route-type> (destination-ipv4-address |
```
<source-ipv4-address> | (<destination-ipv4-address>
<source-ipv4-address>))
<destination-ipv4-address> ::= <ipv4-prefix>
<source-ipv4-address> ::= <ipv4-prefix>
<ipv4-prefix> ::= <IPV4_ADDRESS> <IPV4_PREFIX_LENGTH>

<ipv6-route> ::= <ip-route-type> (<destination-ipv6-address> |
<source-ipv6-address> | (destination-ipv6-address>
<source-ipv6-address>))
<destination-ipv6-address> ::= <ipv6-prefix>
<source-ipv6-address> ::= <ipv6-prefix>
<ipv6-prefix> ::= <IPV6_ADDRESS> <IPV6_PREFIX_LENGTH>
<ip-route-type> ::= <SRC> | <DEST> | <DEST_SRC>

<layer-3-header> ::= <ipv4-header> | <ipv6-header>
<ipv4-header> ::= <SOURCE_IPv4_ADDRESS> <DESTINATION_IPv4_ADDRESS>
<PROTOCOL> [TTL] [DSCP]
<ipv6-header> ::= <SOURCE_IPV6_ADDRESS> <DESTINATION_IPV6ADDRESS>
<NEXT_HEADER> [TRAFFIC_CLASS]
[FLOW_LABEL] [HOP_LIMIT]

<object-based-match> ::= [<user> ...] [schedule] [region]
[target] [state]
:user ::= (<login-name> <group-name> <parent-group> <password>
<schedule> ::= <name> <type> <start-time> <end-time>

<weekly-validity-time>

<type> ::= <once> | <periodic>

<target> ::= [<service>] [<application>] [<device>]

<service> ::= <name> <id> <protocol> [<protocol-num>] [<src-port>]

[<dest-port>]

<protocol> ::= <TCP> | <UDP> | <ICMP> | <ICMPv6> | <IP>

<application> ::= <name> <id> <category> <subcategory>

<data-transmission-model> <risk-level>

<signature>

<category> ::= <business-system> | <Entertainment> | <internet>

<network> | <general>

subcategory> ::= <Finance> | <Email> | <Game> | <media-sharing> |

<brand=-network> | <web-posting> | <proxy> | ...

<data-transmission-model> ::= <client-server> | <browser-based> |

<networking> | <peer-to-peer> |

<unassigned>

<risk-level> ::= <Exploitable> | <Productivity-loss> | <Evasive> |

<Data-loss> | <Malware-vehicle> |

<Bandwidth-consuming> | <Tunneling>
<signature> ::= <server-address> <protocol> <dest-port-num>
   <flow-direction> <object> <keyword>
<flow-direction> ::= <request> | <response> | <bidirection>
<object> ::= <packet> | <flow>
<device> ::= <pc> | <mobile-phone> | <tablet>
<session-state> ::= <new> | <established> | <related> | <invalid> |
   <untracked>

<action> ::= <basic-action> [ <advanced-action> ]
<basic-action> ::= <pass> | <deny> | <mirror> | <call-function> |
   <encapsulation>
<advanced-action> ::= [ <profile-antivirus> ] [ <profile-IPS> ]
   [ <profile-url-filtering> ]
   [ <profile-file-blocking> ]
   [ <profile-data-filtering> ]
   [ <profile-application-control> ]

6. Security Considerations
   TBD

7. IANA Considerations

8. References

8.1. Normative References


8.2. Informative References


9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.
Authors’ Addresses

Liang Xia (Frank)
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: Frank.xialiang@huawei.com

DaCheng Zhang
Alibaba

Email: Dacheng.zdc@alibaba-inc.com

Edward Lopez
Fortinet
899 Kifer Road
Sunnyvale, CA 94086
Phone: +1 703 220 0988

EMail: elopez@fortinet.com

Nicolas BOUTHORS
Qosmos

Email: Nicolas.BOUTHORS@qosmos.com
User-group-based Security Policy for Service Layer

draft-you-i2nsf-user-group-based-policy-00

Abstract

This draft discusses the User-group Aware Policy Control (UAPC) framework that facilitates consistent enforcement of security policies based on user group identity. The framework calls for: (1) a user-group identifier derived from predefined policy criteria (e.g., source IP, time-of-day, device certificate, etc.) for security policy enforcement; (2) a logical policy server that maintains user-group identification policies as well as inter-user-group permission policies; and (3) a logical security controller responsible for managing Network Security Functions (NSFs), and implementing necessary policies on them. The document discusses key northbound APIs of the framework that fall within the scope of the I2NSF Service Layer.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

1. Introduction

In traditional networks, network access is typically controlled through a combination of mechanisms such as maintaining separate static VLAN/IP subnet assignments per organization; applying Access
Control Lists (ACLs) on VLANs and/or IP subnets; leveraging Network Access Control (NAC); etc. Common artifacts of these setups are:

- Network administrators typically assume that users access the network from their own static location--from their assigned switch, VLAN, IP subnet, etc.

- MAC or IP address of the users' device is often used as a proxy for the user's identity. As such, filtering (e.g., via ACLs) of the user is usually based on IP or MAC addresses.

- Authentication of the user by the network, if it exists at all, typically takes place only at the access switch in conjunction with an AAA (Authentication, Authorization, Accounting) server. Different authentication mechanisms could be used - from machine-based certificates to username/password challenges, to just "authenticating" on MAC addresses, etc.

- Network security functions such as firewalls often act only on IP addresses and ports - not on the user’s identity.

Traditional network access control mechanisms [I-D.dunbar-i2nsf-problem-statement] do not work well in newer network paradigms.

- First, both clients and servers can move and change their IP addresses on a regular basis. For example, Wi-Fi and VPN clients as well as backend Virtual Machine (VM)-based servers can move around; their IPs could change as a result. It means relying on well-known network fields like the 5-tuple is increasingly inadequate from a consistent security policy enforcement standpoint.

- Secondly, with more people working from non-traditional office setups like "working from home", there is now a need to be able to apply different security policy to the same set of users under different circumstances. Network access needs to be granted based on such criteria as users’ location, time-of-day, type of network device used (e.g., corporate issued device versus personal device), device’s security posture, etc. It means the network needs to recognize the users’ identity and their current posture, and map the users to their correct access entitlement to the network.

- Moreover, implementation of coherent security policy across several network and network security devices is almost impossible. NSFIs in operation could be sourced from different vendors, or could be different hardware models/software versions by the same
vendor. As a result, the capabilities as well as APIs of the NSFs may not be the same across the board. Finally, few enterprises, if any, have a complete view of all the application flows. It is not uncommon for administrators to update a policy on a firewall but only to find out later that they had missed to update ACLs or firewall policies elsewhere.

Today, addressing the above issues takes considerable time and effort. Most network administrators have to manually plan and implement necessary changes as little automation, if any, exists across diverse sets of network security platforms. In line with the I2NSF effort to standardize APIs so as to facilitate automation, this draft discusses the User-group Aware Policy Control (UAPC) that facilitates consistent enforcement of policies based on user-group identity, and how it operates in the I2NSF Service Layer [I-D.merged-i2nsf-framework]. If the approach is accepted, more detailed implementation mechanisms can be further explored.

2. Terminology

2.1. Abbreviations and acronyms

AAA: Authentication, Authorization, and Accounting
ACL: Access Control List
ADSL: Asymmetric Digital Subscriber Line
AP: Access Point
LTE: Long Term Evolution
NAC: Network Admission Control
NBI: Northbound Interface
NSF: Network Security Function
UAPC: User-group Aware Policy Control
VLAN: Virtual Local Area Network

2.2. Definitions

User: An individual or a group of individuals that act as a single entity.
User-group: A group of users that share one or more characteristics and/or behaviors in common. For example, sales employees are treated with equivalent service policy rules when accessing the network.

Profile: A set of capabilities, in terms of functions and behaviors, for a given entity or set of entities.

Role: A role defines a set of responsibilities of an object that it is attached to. This enables the functions and behavior of a complex object to be abstracted into just those that are required by a client in a particular context.

User-group Identifier (User-group ID): An identifier that represents the collective identity of a group of users, and is determined by a set of one or more matching criteria (e.g., roles, 4-, 5-, and 6-tuples, VLAN ID, etc.) that disambiguates an entity from other entities.

3. Use Cases for User-group Aware Policy Control

With the increased popularity of enterprise wireless networks and remote access technologies such as Virtual Private Networks (VPN), enterprise networks have become borderless, and employees’ locations can be anywhere. Enabling large-scale employee mobility across many access locations improves enterprise production efficiency but also introduces challenges related to enterprise network management and security. The IP address of the user can change frequently when the user is in motion. Consequently, IP address-based policies (such as forwarding, routing, QoS and security policies) may not be flexible enough to accommodate users in motion.

The User-group Aware Policy Control (UAPC) approach is intended to facilitate the consistent enforcement of policies. As shown in Figure 1, a multi-technology network (e.g., Wi-Fi, 3G/LTE, ADSL and fiber infrastructures) can connect different types of terminal devices (e.g., Smartphone, tablet, and laptop) which should be able to access networks in a secure manner. Whether these terminal devices connect to a wired or a wireless infrastructure, security policies should be enforced consistently based on their user-group identities.
4. User-group Aware Policy Control

4.1. Overview

At a high level, the UAPC framework is as follows: Users are authenticated and classified into different user-groups at the network ingress by the Security Controller in consultation with the Policy Server, and an authentication server. The user-group is an identifier that represents the collective identity of a group of users, and is determined by predefined policy criteria (e.g., source IP, geolocation, time of day, device certificate, etc.). Users may be moved to different user-groups if their composite security postures change.

The Security Controller, if necessary, pushes the required changes to all the Network Security Functions (NSFs) that need those changes. The policies are expressed in user-group (not IP or MAC address) IDs so as to decouple the user identity from the network addresses of the user’s device.
(Note that User-group IDs may be implemented in at least two ways: (1) the ingress switch inserts the user-group ID into the packets, and downstream NSFs match and act on the user-group ID, or (2) the Security Controller updates each NSF with the mapping between the user-group IDs and the packet tuples; NSFs map incoming packets to their rightful user-group IDs, and act on the user-group IDs. The two implementation methodologies are out of scope of this document.)

The security policy provisioning information can be derived from the user’s profile and credentials, as well as the group to which the user belongs; such information can also be derived from the outcomes of the dynamic security service parameter negotiation that could possibly take place between the user and the service provider or the network administrator (e.g., parameters like whether the user is entitled to access the enterprise network while in motion or not, the lease time associated to an IP address, whether the user can access the Internet or not, whether traffic needs to be encrypted or not, etc.). This information is transferred to the Network Security Functions (NSF) from the controller. Once an incoming packet matches a certain user group on the NSF, the corresponding security policy will be enforced on the said packet.

4.2. Functional Entities

The UAPC framework consists of three main components: (1) Policy Server, (2) Security Controller, (3) Network Security Functions:

- **Policy Server**
  - Contains (G)UI or APIs to enable policies to be created, modified, and deleted using command line and/or graphical tools
  - Contains logic to create, read, update, delete policies and policy components using one or more policy repositories
  - Contains logic to detect conflicts between policies
  - Contains logic to remediate conflicts between policies
  - Contains logic to broker and/or federate policies between domains

- **Security Controller**
The Security Controller coordinates various network security-related tasks on a set of NSFs under its administration.

- Authenticates the user at the ingress using an authentication service. While the authentication functionality is an integral part of the framework, the topics of defining and managing authentication rules are out of scope of this document.

- Asks policy server for decisions to security-related requests; takes these decisions and invokes the set of NSFs that are required to implement security for that particular packet.

- May perform additional actions as specified by the metadata associated with a policy rule (e.g., the "function(s)" to be executed after the actions in a policy rule are executed).

- Has an authoritative database of NSFs under its administration.

- Determines on which NSFs a given policy needs to be implemented.

- Presents a set of NBIs for applications, orchestration engines, etc.

- Interfaces with NSFs via (to-be-developed) I2NSF Capability Layer APIs.

- Network Security Functions

  - Packet classification: Depending on the implementation model, the NSF may match on User-group IDs in the packets; or it may match on common packet header fields such as the 5-tuple, and map the n-tuple to the appropriate User-group ID supplied out-of-band by the Security Controller.

  - Policy enforcement: Enforce the corresponding policy (or set of policies) if the packet matches a specified User-group ID or set of User-group IDs.

- Presents I2NSF Capability Layer APIs.

4.3. User Group

The user-group is an identifier that represents the collective identity of a group of users, and is determined by predefined policy criteria (e.g., source IP, geolocation, time of day, device certificate, etc.).
A given user is authenticated, and classified at the network ingress, and assigned to a user-group. (The term "user" refers to any user of the network. As such servers too are classified and assigned to their respective user-groups.) A user’s group membership may change as aspects of the user change. For example, if the user-group membership is determined solely by the source IP, then a given user’s user-group ID will change when the user moves to a new IP address that falls outside of the range of addresses of the previous user-group.

User-groups may share several common criteria. That is, user-group criteria are not mutually exclusive. For example, the policy criteria of user-groups R&D Regular and R&D-BYOD may share the same set of users that belong to the R&D organization, and differ only in the type of client (firm-issued clients versus users’ personal clients); likewise, the same user may be assigned to different user-groups depending on the time of day or the type of day (e.g., weekdays versus weekends); and so on.

Table 1 shows an example of how user-group definitions may be constructed.

<table>
<thead>
<tr>
<th>Group Name</th>
<th>Group ID</th>
<th>Group Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>R&amp;D</td>
<td>10</td>
<td>R&amp;D employees</td>
</tr>
<tr>
<td>R&amp;D BYOD</td>
<td>11</td>
<td>Personal devices of R&amp;D employees</td>
</tr>
<tr>
<td>Sales</td>
<td>20</td>
<td>Sales employees</td>
</tr>
<tr>
<td>VIP</td>
<td>30</td>
<td>VIP employees</td>
</tr>
<tr>
<td>Workflow</td>
<td>40</td>
<td>IP addresses of Workflow resource servers</td>
</tr>
<tr>
<td>R&amp;D Resource</td>
<td>50</td>
<td>IP addresses of R&amp;D resource servers</td>
</tr>
<tr>
<td>Sales Resource</td>
<td>54</td>
<td>IP addresses of Sales resource servers</td>
</tr>
</tbody>
</table>
4.4. Inter-group Policy Enforcement

Within the UAPC framework, inter-group policy enforcement requires two key components: (1) user-group-to-user-group access policies, and (2) sets of NSFs on which individual policies need to be applied.

First, the framework calls for an authoritative rule-base that lists all the destination user-groups to which all the source user-groups are entitled to access. The rule-base, hosted on the Policy Server, enables administrators to construct authorized inter-group access relationships. The simple example in Table 2 shows a policy matrix in which the row represents source user-groups and the column represents destination ones. The inter-group rule-base is similar to firewall rule-bases, which are mostly made up of the five-tuple. (Firewall rule-bases could and do include criteria other than the standard five-tuple. And the user-group rule-base could consist of other criteria. Actual implementation details are out of scope of this document.)

**Table 2: Inter-group Policy Example**

<table>
<thead>
<tr>
<th>Source Group</th>
<th>Workflow Group</th>
<th>R&amp;D Resource Group</th>
<th>Sales Resource Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>R&amp;D group</td>
<td>Permit</td>
<td>Permit</td>
<td>Deny</td>
</tr>
<tr>
<td>R&amp;D BYOD group</td>
<td>Permit</td>
<td>Deny</td>
<td>Deny</td>
</tr>
<tr>
<td>Sales group</td>
<td>Permit</td>
<td>Deny</td>
<td>Permit</td>
</tr>
<tr>
<td>VIP user group</td>
<td>Permit</td>
<td>Permit</td>
<td>Permit</td>
</tr>
</tbody>
</table>

The responsibility of implementing and managing the inter-group policies falls to the Security Controller. The controller first needs to determine, (or is told) the specific NSFs on which a given policy is to be implemented. The controller then communicates with each NSF via the I2NSF APIs to execute the required tasks.

Inter-user-group rules are configurable. Figure 2 illustrates how various user-groups and their entitlements may be structured. The example shows a "north-south" model in which how users may access internal network resources. Similar models can be developed for "east-west" intra-data center traffic flows.
Figure 2: Sample Authorization Rules for User-group Aware Policy Control

4.5. UAPC Implementation

The security policies are instantiated and maintained by the policy server. The associated computation logic (to instantiate such policies) may be dynamically fed with instructions coming from the application. The policy decisions could also be from the outcomes of dynamic security service parameter negotiations that typically take place at the management plane between the user and the service provider [RFC7297].

The NSFs receive group-based policy provisioning information from the security controller. The security policies will be enforced so that participating NSFs can process traffic accordingly.
Figure 3: Unified Policy Procedures

1. User-group identification policies and inter-user-group access polices on the Policy Server are managed by the authorized team(s).

2. The user-group-based policies are implemented on the NSF{s under the Security Controller’s management.

3. When a given user first comes up on the network, the user is authenticated at the ingress switch.

4. If the authentication is successful, the user is placed in a user-group, as determined by the Policy Server.

5. The user’s subsequent traffic is allowed or permitted based on the user-group ID by the NSF{s per the inter-user-group access policies. (It is beyond the scope of this document as to how user-group IDs may be delivered to non-ingress NSF{s. See Section 4.1 for a brief overview of possible implementation methods.)
5. Requirements for I2NSF

Key aspects of the UAPC framework falls within the Service Layer of the I2NSF charter. If the community adopts the approach as one possible framework for the Service Layer, the I2NSF Service Layer MUST support at least the following northbound APIs (NBIs):

- The user-group classification policy database on the Policy Server
- The inter-user-group access policy rule-base on the Policy Server
- The inventory of NSFs under management by the Security Controller
- The list of NSFs on which a given inter-user-group policy is to be implemented by the Security Controller.

The framework also assumes that the I2NSF Capability Layer APIs will be there for the NSFs.

6. Acknowledgement

The editors would like to thank Linda Dunbar and John Strassner for a thorough review and useful comments.

7. References

7.1. Normative References


7.2. Informative References


Authors’ Addresses

Jianjie You
Huawei
101 Software Avenue, Yuhuatai District
Nanjing,  210012
China

Email: youjianjie@huawei.com

Myo Zarny
Goldman Sachs
30 Hudson Street
Jersey City,  NJ 07302
USA

Email: myo.zarny@gs.com

Christian Jacquenet
France Telecom
Rennes 35000
France

Email: christian.jacquenet@orange.com

Mohamed Boucadair
France Telecom
Rennes 35000
France

Email: mohamed.boucadair@orange.com
Yizhou Li
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, 210012
China

Email: liyizhou@huawei.com

Sumandra Majee
F5 Networks
3545 N 1st St
San Jose, CA 95134

Email: S.Majee@f5.com
The Capability Interface for Monitoring Network Security Functions (NSF) in I2NSF
draft-zhou-i2nsf-capability-interface-monitoring-00

Abstract

This document focuses on the monitoring aspects of the flow-based Network Security Functions (NSFs). The NSF Capability interface between the Service Provider’s management system (or Security Controller) and the NSFs is meant to enforce the required monitoring capability.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 21, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
Rising security problems bring challenges for the security enterprises and organizations. There are already some software and hardware devices deployed for these problems, e.g., antivirus, firewall, intrusion detection systems (IDS), Web security gateway, and DPI, which build up many safety lines accordingly. However, one individual safety line only defends the security threats of only one aspect. And more seriously, these safety defense devices are generating large amount of logs and events in the operating procedure, which makes a lot of "information island". With large quantity and isolated security information, it brings low efficiency for the security managers who operate on their own control platform and warning window of various devices.

The network security mechanism would be more efficient if the Security Controller defined in [I-D.merged-i2nsf-framework] could monitor, analyze and investigate the abnormal events and finally produce security reports to the security administrators. The security controller should also be able to collect the traffic and session information from the NSF, in order to steer the suspected attack source or victim traffic to the cleaning center.

The data mining use case defined in [I-D.xia-dots-extended-use-cases] has provided a good example for distributed denial-of-service (DDoS) attack monitoring report. [I-D.reddy-dots-info-model] also describes the information model of flow monitoring to help identify DDOS...
attacks in a network. This document aims to cover more cases and more attack types in the network, e.g., botnets, spam, and spyware.

This document describes how to use the capability interface to collect the security information from the NSFs and which security information will be reported using this interface. The protocol and information model will be described for the monitoring aspects of the Capability Interface.

2. Terminology

3. The Capability Interface for Monitoring

3.1. Overview

The capability interface should be able to provide unified event format for the logs and warning information of the overall network, to facilitate the failure discovery and failure locating timely and accurately. With the unified event format, the security managers could run easily and quickly through various security events which guarantees the availability of service information system and service continuity. To achieve this, the Security Controller needs to collect the detailed information of the network status and traffic information from the NSF to make intelligent security decision and to dynamically adjust the sampling and steering policy.

![Diagram of Monitoring Interface]

Figure 1: Monitoring Interface
As described in Figure 1, the traffic monitoring procedure involves two network elements: Security Controller (SC) and NSF. The NSF reports the monitoring information to the SC, which provides the specific traffic information, e.g., abnormal flows, security logs, statistics or the suspicious attack sources or destinations. The SC is responsible for monitoring and collecting traffic information from NSFs. Based on the input from the NSF, the SC could make policy enforcement for the specific flows, e.g., traffic steering or adjusting the flow sampling policies.

3.2. Traffic Report

The traffic reported from the NSF may contain the information of IP addresses, security logs, statistics, warnings, and etc. The syslog protocol [RFC5424] could be used to send event notification messages to the SC for traffic collection. The IP Flow Information Export (IPFIX) protocol [RFC5101] may also be adopted for the flow sampling information collection. There are mainly three types of information reported using the capability interface:

- Traffic Statistics:
  - Normal traffic statistics based on the source and destination address, including byte per second (bps) and packet per second (pps).
  - Abnormal traffic statistics based on the source and destination address, including bps and pps.
  - Traffic statistics based on the network address range (including port usage), including bps and pps.

- Session Statistics:
  - Concurrent session statistics based on the source and destination address.
  - New session statistics based on the source and destination address.
  - Abnormal session statistics based on the source and destination address, including null link, retransmission session and slow-start link.

- Abnormal/Attack Event: analysis results of the relevant abnormal/attack event, e.g., time, type, level, detail description, threshold, source IP address and destination IP address.
The NSF could report the accurate source and destination of the attack to the security controller for which to make traffic steering policy, e.g., steering the bad "botnet" traffic to the cleaning center. The type, size and proportion of the abnormal traffic could also be reported to assist the security controller to determine the steering priority, e.g., priority traction, large flow cleaning.

3.3. Policy Enforcement

The Security Controller is responsible for making policy decisions after getting the security information from the NSF (and, typically, other instructions from the operator). It will provide the attack mitigation and defense strategy with the acquired sampling traffic information for attack detection by the way of dynamically adjusting the flow sampling policy, e.g., flow information, sampling ratio, sampling encapsulation method and/or sampling point information. The policies may include: traffic cleaning and sampling adjustment.

- Traffic cleaning: with the suspicious result of the analyzed sampling traffic, the security controller dynamically sends the steering policy to the related NSF or sends the flow blocking policy to the NSF which is nearest to the attack point, to block the attack traffic from the source. The traffic cleaning policy may include the following three ones:
  * Access Control List (ACL), to block the attack traffic.
  * Packet hash digest to block the attack traffic.
  * Traffic steering policy to clean the traffic.

- Flow sampling adjustment: after filtering the abnormal object of the sampling traffic, the security controller could dynamically adjust the sampling policy to improve the sampling rate of the TOPN abnormal object, in order to make more accurate attack detection. The abnormal object may include: Top N network address range of the abnormal session, Top N IP address of the abnormal session and the Top N of abnormal sessions.

4. Security Considerations

5. IANA Considerations

This document has no actions for IANA.
6. References

6.1. Normative References


6.2. Informative References


Authors' Addresses

Cathy Zhou
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: cathy.zhou@huawei.com

Liang Xia (Frank)
Huawei Technologies
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
P.R. China

Email: Frank.xialiang@huawei.com
Mohamed Boucadair  
France Telecom  
Rennes  35000  
France  
Email: mohamed.boucadair@orange.com

Jie Xiong  
Huawei Technologies  
101 Software Avenue, Yuhuatai District  
Nanjing, Jiangsu  210012  
P.R. China  
Email: emma.xiong@huawei.com