I2RS working group
Internet-Draft
Intended status: Standards Track
Expires: September 21, 2016

S. Hares
Huawei

March 20, 2016

I2NSF Management Traffic Flow Requirement
draft-hares-i2nsf-mgtflow-reqs-01.txt

Abstract

This document discuss the stresses on I2NSF management traffic during periods DDoS and network attacks, and how application layer tuning of I2NSF management traffic can improve the managementtraffic flow.

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 September 21, 2016.

Table of Contents

1. Introduction

The Interface to the Network Security Function (I2NSF) Working Group is chartered with providing architecture and mechanisms to inject into and retrieve information from network security devices. The I2NSF problem statement ([I-D.ietf-i2nsf-problem-and-use-cases] indicates that service providers lack a standard management interface which preserves:

- critical communications during DDoS attacks (DOTS),
- allows hosts to continue even during the DDoS attacks,
- aids reporting of these attacks the CERT (MILE),
- and manages network connnectivity of devices out of compliance (SACM).

This document describes the stress on I2NSF management traffic during DDoS and network attacks/incidents, and some mechanisms that help traffic flow during these periods. I2NSF considers two directions: I2NSF controller to NSF/vNSF, and I2NSF user to I2NSF controller.

2. Stresses on traffic between I2NSF and vNSF/NSF

During periods of DDoS attacks, I2NSF management traffic may encounter high error rates, congestion, restricted bandwidth caused by DDoS related traffic (ICMP spams, transport protocol SYN attacks, port spams, and others.), or attacks on specific network machines. Message integrity may be compromised by attacks on the transport

protocols, or by replay attacks on message sequence. However, during this same time period the I2NSF controller needs to send to NSFs/ vNSFs new filter policies or other configuration changes. IDS/IPS NSF functions may need to send I2NSF controller information to help detect the attack source or stop the attack.

During DDOS attacks or network security incidents, the client programs may want to receive status information from the I2NSF controller. This communication will also be impacted by the high error rates, congestion, and restricted bandwidth caused by DDoS related traffic or network security attacks.

This stress can be illustrated by examining two types of management traffic which need to be exchanged with the I2NSF controller: DDoS Open Threat Signaling (DOTS) traffic, and security incident (CERT) traffic reports.

2.1. DOTS (DDoS Open Threat Signaling) Management Traffic

Sending information about DDoS threats occurs during periods where the DDoS is congesting the network or causing large packet losses. I2NSF controllers may receive requests from DOTS controllers to configure new network security functions (NSFs) or reconfigure existing security functions on vNSF or NSF devices. I2NSF controllers may need to receive specific events from vNSF/NSF devices, and receive traffic monitoring data and logs regarding network security incidents.

The DOTS requirements for messages from devices with security functions (such as firewalls in routing devices) are specified in: [I-D.ietf-dots-requirements]. The following are DOTS descriptions of the resiliency needed by the management data:

- Resilence (DOTS-G-003) in the face of severally constrained severely constrained network conditions imposed by the attack traffic. The protocol SHOULD be resilient, that is, continue operating despite message loss and out-of-order or redundant signal delivery,

- Small message sizes (DOTS-G-005) to prevent fragmentation so that all of the message goes through in attack,

- Message integrity (G-006) and Message level replay protection (G-007) must exist for data streams even during periods of attack,

- Session-level Health monitoring (aka Heart beats) during attack (DOTS-OP-003), and

- o Abiilty to request/stop mitigation quickly (DOTS-OP-005)

2.2. MILE - Managed Incident Lightweight Exchange

Reporting and managing security incident traffic is being investigated by the MILE working group. The MILE related protocols ([RFC5070], [I-D.ietf-mile-rfc5070-bis]) provide data formats for reporting network security incidents during time periods of network attack. Similar to DOTS, the data passed by these protocols requires resilience, message integrity, message level replay protection, and session-level health monitoring. During these attacks, the use of small message sizes may be necessary.

3. Stresses on I2NSF controller to User traffic

The user application communicating with the network security controller uses the I2NSF protocol to:

- o give commands that direct the actions of the Network Security Controller during normal operation and during periods of security attack,
- o give commands to direct the creation of policy on the Network Security controller, or on the NSF or vNSF devices,
- o receive reports on the status of network security including DDoS attacks, outages, and devices operating outside the appropriate security software or actions, and
- o give commands to link the network security controller to additional resources (e.g. CERT for incident report or additional IDS/IPS services)/

The communication to perform security operations may encounter DDoS and network attack related outages, network congestion (bursts of congestion or time periods of congestion), and specific network attacks on messages protocols (E.g. TCP syn attacks, ICMP based attacks).

4. I2NSF Management Traffic Flow Needs

The I2NSF communication needs to support application layer services that handle the transport layer’s failure to support critical communication. These application services must provide the following to preserve the end-to-end communication between I2NSF controller to NSF/vNSF and between I2NSF controller and the user:

- o data flow resilence,

- o breaking the data traffic into appropriate sizes for pass through congestion (aka "chunking" the data) and re-assembly of data prior to handing to application,

- o message integrity and replay protection,

Each I2NSF agent and I2NSF client needs to provide this support at the application level since security attacks often attack the tranport connections. This is true whether the communication is between the I2NSF Controller to vNSF/NSF device, or between the user's client device and the I2NSF controller.

5. I2NSF Protocol with Session Layer Services

The diagram in figure 1 shows how a secure session service (SSE) at the application layer of the I2NSF protocol that could provide these

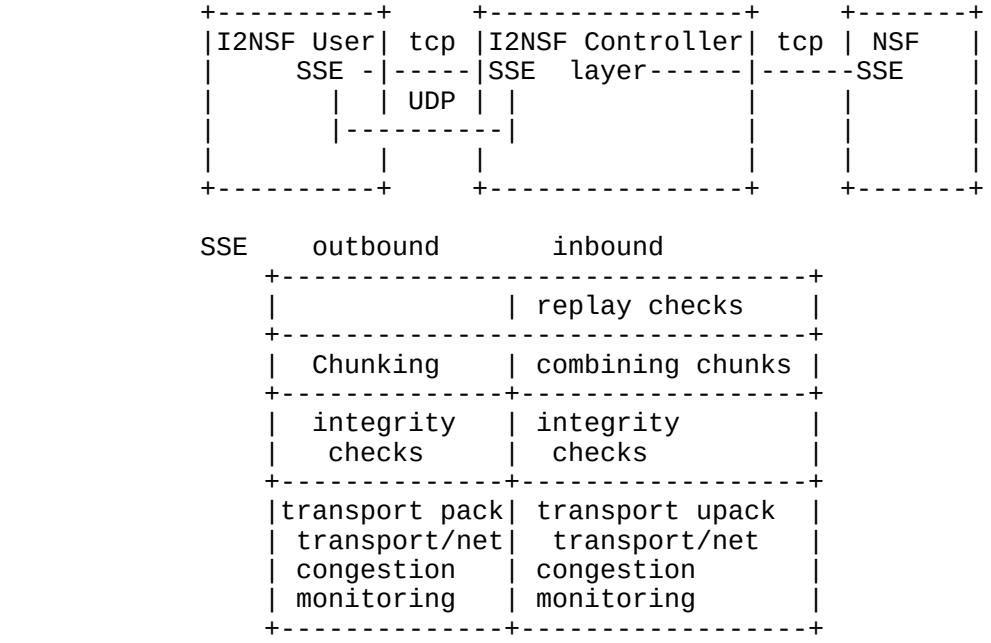


Figure 1

6. Impact of I2NSF potential use of I2RS protocol

I2NSF protocol may want to consider extending the I2RS protocol [I-D.hares-i2rs-protocol-strawman] for communication to routers/ switches that have onboard security functions. The first version of

the I2RS protocol will support communication by NETCONF [RFC6241] (with extensions), RESTCONF [I-D.ietf-netconf-restconf] (with extensions), and other protocols. The I2RS working group is seeking feedback on management traffic during network outages (security related or network connectivity related) in order to determine what protocols are needed beyond NETCONF and RESTCONF. This management traffic includes configuration, events, log information, alerts, traffic monitoring information, traffic statistics, and end-to-end performance information. I2NSF could help the I2RS working group determine the security management information needed to be passed to NSF or vNSF functions in routers.

7. IANA Considerations

There are no IANA requirements for this requirementdocument.

8. Security Considerations

TBD

9. Acknowledgements

The following people have aided in the discussion

- Russ White,and
- Robert Moskowitz.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

10.2. Informative References

[I-D.hares-i2rs-dataflow-req] Hares, S., "I2RS Data Flow Requirements", draft-hares-i2rs-dataflow-req-01 (work in progress), March 2016.

[I-D.hares-i2rs-protocol-strawman] Hares, S., "I2RS protocol strawman", draft-hares-i2rs-protocol-strawman-00 (work in progress), March 2016.

[I-D.ietf-dots-requirements]
Mortensen, A., Moskowitz, R., and T. Reddy, "DDoS Open Threat Signaling Requirements", draft-ietf-dots-requirements-00 (work in progress), October 2015.

[I-D.ietf-i2nsf-problem-and-use-cases]
Hares, S., Dunbar, L., Lopez, D., Zarny, M., and C. Jacquenet, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-00 (work in progress), February 2016.

[I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-13 (work in progress), February 2016.

[I-D.ietf-mile-rfc5070-bis]
Danyliw, R., "The Incident Object Description Exchange Format v2", draft-ietf-mile-rfc5070-bis-16 (work in progress), February 2016.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-10 (work in progress), March 2016.

[RFC5070] Danyliw, R., Meijer, J., and Y. Demchenko, "The Incident Object Description Exchange Format", RFC 5070, DOI 10.17487/RFC5070, December 2007, <http://www.rfc-editor.org/info/rfc5070>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

Author’s Address

Susan Hares
Huawei
Saline
US

Email: shares@ndzh.com

I2NSF
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

S. Hares
Huawei
R. Moskowitz
HTT Consulting

March 21, 2016

Secure Session Layer Services
draft-hares-i2nsf-ssls-00.txt

Abstract

Each I2NSF agent and I2NSF client needs to provide application level support for management traffic during periods of DDoS and network security attacks to deal with congestion (burst and/or continuous), high error rates and packet loss due to the attacks, and the inability to utilize a transport protocol (E.g. TCP) due to a specific protocol attack. This application level support needs to be able to select the key management system and provide "chunking" of data (in order to fit in reduced effective MTUs), compression of data (in order to fit into reduced bandwidth), small security envelope)in order to maximize room for mangement payload), and fragmentation and reassembly at the application layer for those protocols which do not support fragmentation/reassembly (E.g. UDP or SMS). The application layer needs to be able to turn off this features if the system detects these features are no longer needed.

This draft specifies a security session layer services(SSLs) which provide these features in terms of an API, and the component features (interface to key management systems, data compression, chunking of data, secure session envelope (SSE) to send data, and fragmentation and reassembly, and ability to detect existence of attack).

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 September 22, 2016.

Table of Contents

1. Introduction

Each I2NSF agent and I2NSF client needs to provide application level support for management traffic during periods of DDoS and network security attacks to deal with congestion (burst and/or continuous), high error rates and packet loss due to the attacks, and the

inability to utilize a transport protocol (E.g. TCP) due to a specific protocol attack. Some of the services the I2NSF controller must provide during these periods of DDoS or network security attacks are:

- receiving information regarding DDoS Threats from DOTS systems,
- Changing policy on vNSF and NSF devices during these periods,
- exchanging information with user security applications using I2NSF to obtain information from the controller,
- Aid the I2NSF reporting of attacks with the the CERT (MILE) either by providing data or sendign the report
- and manages network connnectivity of devices out of compliance (SACM).

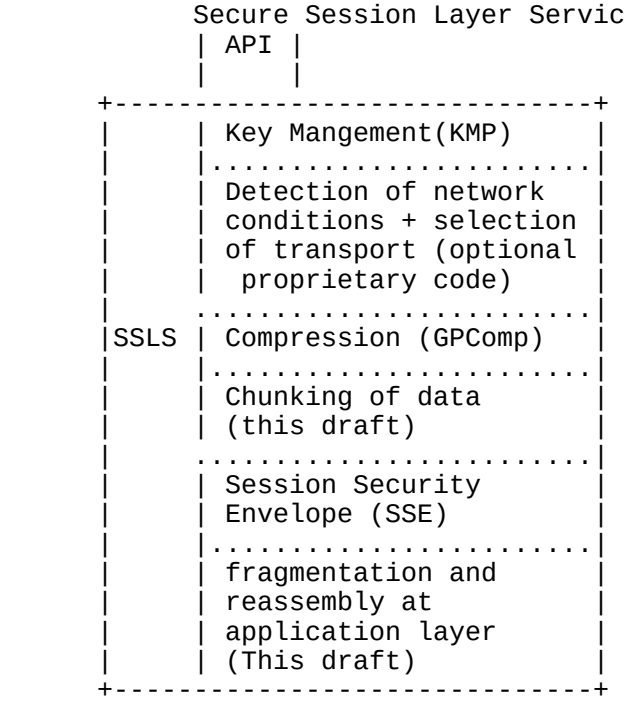
This application level support for I2NSF client-agent communication needs to be able to select the key management system and provide "chunking" of data (in order to fit in reduced effective MTUs), compression of data (in order to fit into reduced bandwidth), small security envelope)in order to maximize room for mangement payload), and fragmentation and reassembly at the application layer for those protocols which do not support fragmentation/reassembly (E.g. UDP or SMS). The application layer needs to be able to turn off this features if the system detects these features are no longer needed.

This draft specifies a security session layer (SSL) which provides these features in terms of:

- an API for the layer (section 2)
- interface to key management system (section 3),
- data compression (section 4)
- chunking of data (section 5)
- secure envelope (section 6),
- fragmentation and reassembly (section 7),
- detection of network conditions that require this service (section 8).

A diagram of the SSLS with these process is in figure 1.

The API for this SSLS allows the application to select the types of key management, and the different types of services (data compression, chunking of data, secure e)



2. API for SSLS

2.1. SSLS socket calls

The SSLS uses socket calls to set up the application session layer. The calls are shown below.

s = int socket(int domain, int type, int protocol)

where:

domain: AF_INET and AF_INET6 supported

type: SOCK_SSLS

desired protocol: Transport protocol (TCP (6), UDP (6), SCTP (132)), SMS (xx)

```
     int setsockopt(int sockfd, int level, int optname,
                const void *optval, socklen_t optlen);

    int getsocketopt(int sockfd, int level, int optname
                      const void *optval, socket
     where:
      sockfd:      # socket file descriptor
      optname:     # option name (see below)
      optval;          # points to *sse_transport structure;
      optlen;          # length of option

      optnam:
      SSLS_AUTH_PRIV ]1]
      SSLS_AES_MODE[2]
      SSLS_ALGS[3]
      SSLS_SSE [4]
```

Where the opt_val structure are define in the figure below.

Figure 2

2.1.1. KMP related options

```
    Security Keying structures for:
    SSLS_AUTH_PRV, SSLS_AES_MODE, SSLS_ALGS
    options of setsockopt, getsockopt

    #struture for SSL_AUTH-PRIV optval
       struct *ssls_auth-priv_opts  {
              *ssls-x509-auth [SSLS-X509-LIMIT]
          }

          #SSL-X509-limit
          typedef struct ssls-x509-auth {
             const char name;
             void *x509-cert;  #cert struture by API
          }


          #structure for SSL_AES_MODE optval
          struct *ssls_aes_mode_opts {
        ... IKEV2 options # openikev2 API
        ... HIPv2 options # HIPv2 API
                                              #[RFC6317 + HIPv2]
         struct ssls_algs_opts;
          }

          #compression options
          struct *ssls_algs_opts {
            boolean  gpcomp-kmp; # computed with keys
       enum gmcomp-type;    #
           }
```

figure 3: setsockopt structure
for KMP related optins

2.1.2. SSE Envelope related options

```
     Security Session Envelope Related options
            #structure for SSL_SSE optval
            # SPI - is generated by KMP
            # SSE - sequence number - by SSE
            # Flags = Fragment (5 bits [0-5],

       struct *ssls_sse_opts {
            int nt_sockfd;   # new transport socket
            int *protocol;   # transport protocol for SSLS SSE
                             # can choose from (1-n )
            int *known_ports # known ports
            int chunk-size;  # chunk size
            int frag-size;   # fragment size
                             # greater than 0 means fragment]
            int SSEs-at-once # number of SSEs sent at once
            enum SSE_size;   # (compact, large, extreme)
            enum SSE-FLAG;   # compression flags
            );
```

Figure 4

2.2. OpenSSL X.509 API calls used

TBD

2.3. HIPv2 API calls used

(API calls will be added later based on HIP [RFC6317] upgraded to HIPv2.

2.3.1. HIP Structures

```
           struct addrinfo {
              int       ai_flags;          /* e.g., AI_CANONNAME */
              int       ai_family;         /* e.g., AF_HIP */
              int       ai_socktype;       /* e.g., SOCK_STREAM */
              int       ai_protocol;       /* 0 or IPPROTO_HIP */
              socklen_t ai_addrlen;        /* size of *ai_addr  */
              struct    sockaddr *ai_addr; /* sockaddr_hip */
              char     *ai_canonname;      /* canon. name of the host */
              struct    addrinfo *ai_next; /* next endpoint */
              int       ai_eflags;         /* RFC 5014 extension */
           };
```

2.3.2. HIP KMP calls

```
  #HIP uses
  #   #include <netdb.h>
 int getaddrinfo(const char *nodename,
                 const char *servname,
                 const struct addrinfo *hints,
                  struct addrinfo **res)
    void free_addrinfo(struct addrinfo *res)
```

Figure 3

3. Data Compression

The first step in making the application data easier to send through the network is to compress the data. The data compression algorithm is defined in draft-moskowitz-gpcomp-00.txt. The result of the compressed data is handed to the chunking function.

The user can disable or enable the compression function by setting SSE-DATA types to be one of the following:

- SSLS compress only - set compression, [1]
- SSLS compression and fragmentation [3],

Setting this flag to:

- no compression or fragmentation [0],
- SSLS to fragmentation only [2]

will skip the data compression step.

4. SSLS Processes

4.1. Chunking of Data

The process that "chunks" data breaks down the application stream after the compression process. If the compression process has compressed the data, the chunking process will chunk compressed data. If the user has requested no compression, this chunking process will chunk uncompressed data. The size of chunks of data the SSLS process creates to encapsulate in the secure session envelope (SSE) is specified on SSL_SSE setsockopt call.

The secure session envelope must be bigger than the chunk.

If the SSE is using TCP or STCP, that assembles the application flow into a byte stream, then the SSE packages will contain a chunk within the secure session envelope.

If Transports that do not fragment and re-assembly are being specified, the SSL will support application layer fragmentation and reassembly. (see the fragmentation section below

4.2. Secure Session Envelope

The Secure Session Envelope (SSE) creates a secure envelope using the SPI created by the key management and running over the transport selected by the user. The SSE has three forms: compact, Large, Extreme. The SSE compact form is below in figure x. SSL defines 4 bytes of the reserved field in the FLAGS field. See [I-D.moskowitz-sse] for details on secure session envelope sizes and formats.

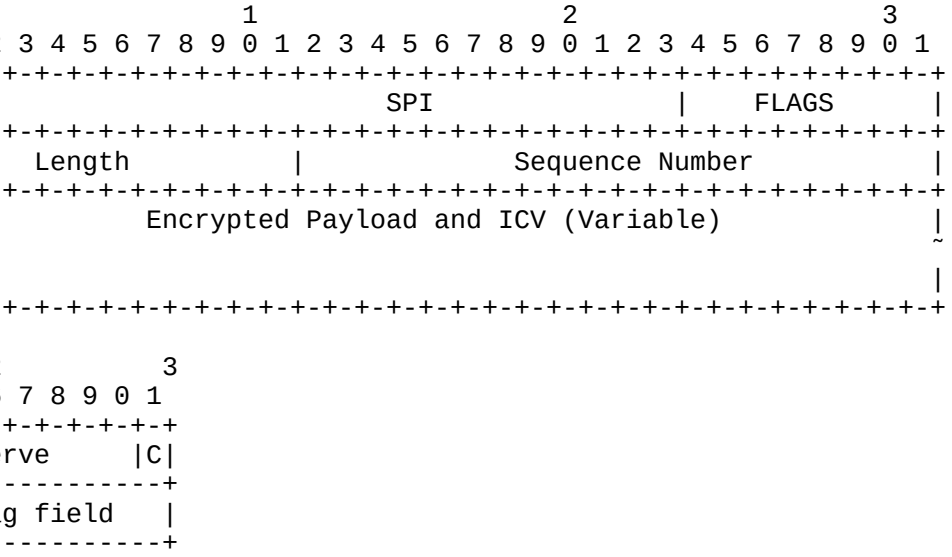
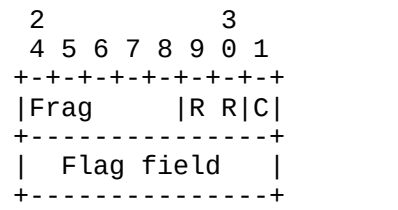


Figure 5 - Compact format of SSE

The SSLS utilizes 6 bits of the 8 bit flag in order to provide provide fragmentation and reassembly checks when the SSE gets fragmented into multiple transport packets. Each time the SSE fragments the packet to fit in the transport, it increments the fragment count in bits [24-28]. The bits for the flag word shown in figure 6.



Flag work in SSE header

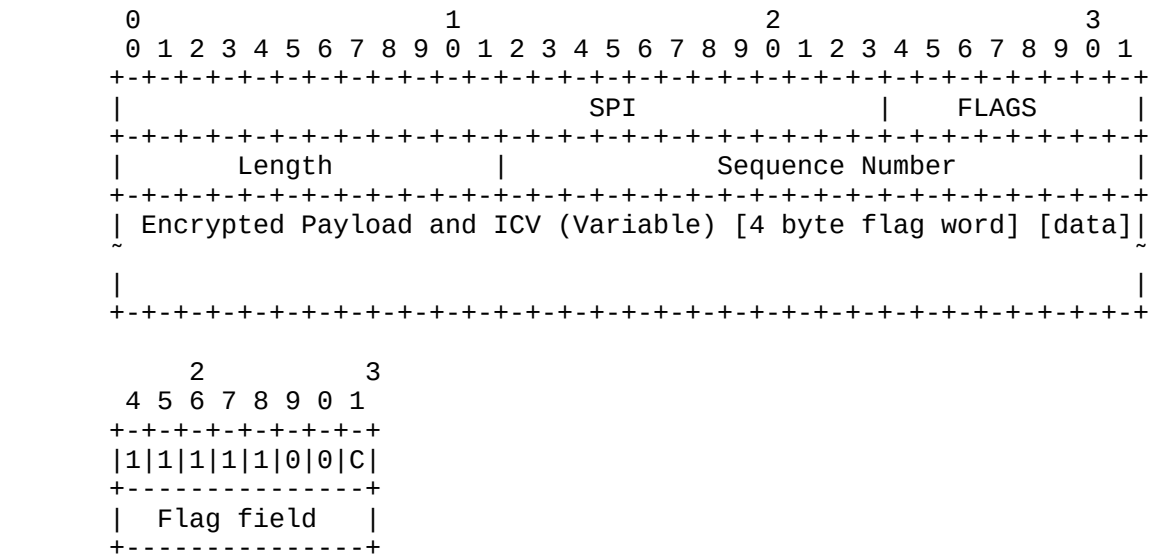
Bits [4-8] - 1-30 bit value for the fragment number
0 - no fragmentation
31 - indicates an fragmentation ACK response
Bits 5-6 - reserved
Bit 7 - compression

Figure 6 - SSLS redefined SSE Flag byte

4.3. Application Packet Fragmentation and Reassembly

SSE's secure envelope may be passed over UDP to avoid transport-level security attacks. Alternatively SSE's secure transport may go over the extremely limited SMS fabric so that some security management information gets through. In both cases, the user (or the "detection log") can select the transport and fragmentation.

If fragmentation is turned on, the individual SSE envelopes will track the IP messages the SSE envelope is broken into by placing the fragment number in the lowest 5 bits of the SSE Flag byte [24-28]. The SSE process receiving the traffic will send back an acknowledge SSE packet [Flag value in bits 0-4 is 0x1F or 31] within 30 bit map of sequences acked [1-30] in first 4 bits of SSE data. It is anticipate that the fragmentation process will attempt to bundle some acks.



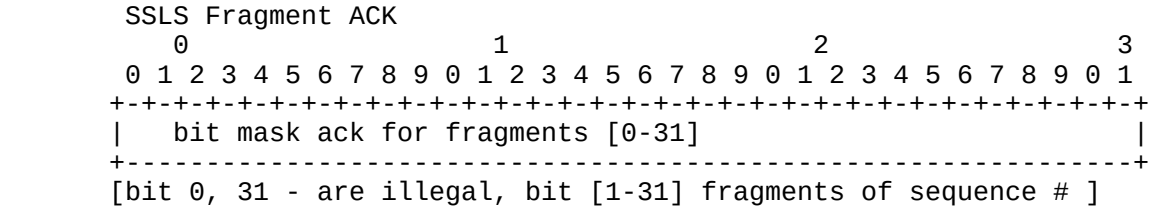


Figure 7 - SSLS ACK flag filed and first 4 bytes of payload

An example Fragmentation and ACK exchange

```
    SSLS-process-1--------IP/SMS---------SSLS Process-2
     [E.g. I2NSF Client -----------------I2NSF Agent]

    SSE-packet (SPI,(flags(fragment=1,C=1),
         length, seq 1, data )---->

     SSE-packet (SPI,(flags(fragment=2,C=1),
         length, seq 1, data )---->

     SSE-packet (SPI,(flags(fragment=3,C=1),
         length, seq 1, data )---->

     SSE-packet (SPI,(flags(fragment=1,C=1),
         length, seq 2, data )---->

      SSE-packet (SPI,(flags(fragment=2,C=1),
         length, seq 2, data )---->
                    <--SSE-packet (SPI)(flags fragment=31,C=1)
                length, seq1,[ack-fragment 1,2])
                           <--SSE-packet (SPI)(flags fragment=32,C=1)
                             length, seq2,[ack-fragment1,2]

     SSE-packet (SPI,(flags(fragment=3,C=1),
         length, seq 1, data )---->
                    <--SSE-packet (SPI)(flags fragment=31,C=1)
                length, seq1,[ack-fragment 3])
```

Below is a set of pseudo call for the calls to socket

```
     pseudo
    struct sse_opts = {};
    optlen=size(sse_opts);
    optname= SSLS_SSE; #4
    s = int socket(int domain, int type, int protocol)
    errno =  int setsockopt(sockfd,level,optname,
             void struct *sse_opts,optlen);

    Errors: (Exact ERNOS added later)
     - protocol not support
     - error in known ports
     - error in chunk_size
     - error in fragment size
     - error in SSE-at-once
     - error - unsupported SSE
     - error in compression flags

     [Add read-write to socket ]
```

The SSE window size for fragmentation is 30 IP fragments or 30 SMS fragments per SSE chunk. The SSE process SHOULD assign the SSE fragments in order if possible. The SSE process will send an error response to the SSE if the data chunk does not fit in 30 IP/SM fragments.

If the SSE transmitting process has not received an acknowlegement for all IP fragments for a particular SSE envelope (identified by sequence number) with a SSE-retransmit-time, it sill retransmit the unacknowledged fragments.

Several SSE envelopes may be sent with fragmentation at once. The user signals the number sent at once with multiple SSE with fragment variable on the options. If fragmentation is selected, each of these SSE envelopes may need to track up to 30 IP fragments.

4.4. Proprietary Plugins: Detect Conditions + Select Transport

The SSL process allows two properitary plugins:

1. Plugin to detect error conditions which require SSLS services which include:
    * High levels of end-to-end congestion,
    * High levels of error and loss,
    * Input from IDS/IPS that detects problems
    * Signals from other I2NSF applications
2. Proprietary actions may select transport based on input from other standardize security services (DOTS, CERT, MILE) or proprietary services.

Prototype code will provide instances to show plugin values.

5. IANA Considerations

TBD

6. Security Considerations

The SSLS shares the following security considerations with the SSE Technology:

- As SSE uses an AEAD block cipher, it is vulnerable to attack if a sequence number is reused for a given key. Thus implementations

of SSE MUST provide for rekeying prior to Sequence Number rollover. An implementation should never assume that for a given context, the sequence number space will never be exhausted. Key Management Protocols like IKEv2 [RFC7296] or HIP [RFC7401] could be used to provide for rekeying management. The KMP SHOULD not create a network layer fate-sharing limitation.

- o As any security protocol can be used for a resource exhaustion attack, implementations should consider methods to mitigate flooding attacks of messages with valid SPIs but invalid content. Even with the ICV check, resources are still consumed to validate the ICV.

- o SSE makes no attempt to recommend the ICV length. For constrained network implementations, other sources should guide the implementation as to ICV length selection. The ICV length selection SHOULD be the the responsibility of the KMP.

- o As with any layered security protocol, SSE makes no claims of protecting lower or higher processes in the communication stack. Each layer’s risks and liabilities need be addressed at that level.

7. Acknowledgements

The authos would like to thank Frank (Liang) Xia for his comments and suggestions on this draft.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

8.2. Informative References

[I-D.hares-i2nsf-mgtflow-reqs]
Hares, S., "I2NSF Data Flow Requirements", draft-hares-i2nsf-mgtflow-reqs-00 (work in progress), March 2016.

[I-D.moskowitz-sse]
Moskowitz, R., Faynberg, I., Lu, H., Hares, S., and P. Giacomin, "Session Security Envelope", draft-moskowitz-sse-02 (work in progress), February 2016.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6317] Komu, M. and T. Henderson, "Basic Socket Interface Extensions for the Host Identity Protocol (HIP)", RFC 6317, DOI 10.17487/RFC6317, July 2011, <http://www.rfc-editor.org/info/rfc6317>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

[RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <http://www.rfc-editor.org/info/rfc7401>.

Authors' Addresses

Susan Hares
Huawei
Saline
US

Email: shares@ndzh.com

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237
USA

Phone: +1-248-968-9809
Email: rgm@htt-consult.com

I2NSF
Internet-Draft
Intended status: Standards Track
Expires: September 4, 2016

S. Hares
J. Strassner
Huawei
D. Lopez
Telefonica I+D
L. Xia
Huawei
March 3, 2016

I2NSF Terminology
draft-hares-i2nsf-terminology-00.txt

Abstract

This document describes the terminology for I2NSF.

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 September 4, 2016.

Copyright Notice

Table of Contents

1. Introduction

This document describes the terminology for the work on the Interface to Security Functions (I2NSF). This section provides some background on I2NSF, but a problem statement can be found in [I-D.ietf-i2nsf-problem-and-use-cases]

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. 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 may consume network security services hosted by one or more providers, which may be their own enterprise, service providers, or a combination of both.

2. Terminology

AAA: Authentication, Authorization, and Accounting. See individual definitions.

Abstraction: An abstraction defines the salient characteristics and behavior of an object that distinguish it from all other types of objects. It manages complexity by exposing common properties between objects and processes while hiding detail that is not relevant.

Accounting: TBD

ACL: Access Control List. This is a mechanism for defining a set of permissions that are attached to an object.

Action: is a set of purposeful activities that have a set of associated behavior. (see I2NSF Action below.) (from [I-D.strassner-supa-generic-policy-info-model])

Authentication: TBD

Authorization: TBD

B2B: Business-to-Business

Bespoke: Something made to fit a particular person, client or company.

Bespoke security management: Security management systems which are make to fit a particular customer.

Boolean Clause: A logical statement that evaluates to either TRUE or FALSE. Also called Boolean Expression.

Capability: TBD

Capability Layer: TBD [Editorial comment from Strassner: the existing definition in use in documents is descriptive, not prescriptive.]

Condition: a set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to make a decision. Examples of an I2NSF Condition include matching attributes of a packet or flow, and comparing the internal state of a NSF to a desired state. A Condition, when used in the context of a Policy Rule, is used to determine whether or not the set of Actions in that Policy Rule can be executed or not. (from [I-D.strassner-supa-generic-policy-info-model])

Constraint: A constraint is a limitation or restriction. Constraints may be added to any type of object (e.g., events, conditions, and actions in Policy Rules).

Constraint Programming: a type of programming that uses constraints to define relations between variables in order to find a feasible (and not necessarily optimal) solution.

Context: The Context of an Entity is a collection of measured and/or inferred knowledge that describe the state and the environment in which an Entity exists or has existed. (from http://www.ietf.org/mail-archive/web/i2nsf/current/msg00762.html)

Controller: TBD [Editorial: The definition is lacking content ("used interchangeably with Service Provider Security Controller or management system throughout this document") and overloaded - the two terms should be split into two separate definitions in documents.]

DC: Data Center

Data Model: A data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol (typically one or more of these). (from [I-D.strassner-supa-generic-policy-info-model]).

Event: An Event is defined as any important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. Examples of an I2NSF EVent include time and user actioins (e.g. logon, logoff, and actions that violate and ACL.) An Event, when used in the context of a Policy Rule, is used to determine whether the condition clause of an imperative Policy Rule can be evaluated or not. (from [I-D.strassner-supa-generic-policy-info-model]).

ECA: Event - Condition - Action policy.

FW: Firewall

Flow-based NSF: A NSF 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, or NAT inspection or changes).

I2NSF Action: An I2NSF Action is a special type of Action that is used to control and monitor aspects of physical and virtual flow-based Network Security Functions. Examples of I2NSF Actions include providing intrusion detection and/or protection, web and flow filtering, and deep packet inspection for packets and flows. An I2NSF Action, when used in the context of a I2NSF Policy Rule, may be executed when both the event and the condition clauses of its owning I2NSF Policy Rule evaluate to true. The execution of this action may be influenced by applicable metadata. (see [I-D.strassner-supa-generic-policy-info-model]).

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

I2NSF client: A security client software component that utilizes the I2NSF protocol to read, write or change provisioning and operational aspects for the NSFs it attaches to by using the I2NSF protocol

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

I2NSF Policy: is a set of rules that are used to manage and control the changing or maintaining of the state of an security device.

I2NSF Policy Rule: is a policy rule that is adapted for I2NSF. The I2NSF Policy Rule is assumed to be in ECA form (i.e., an imperative structure). Other types of programming paradigms (e.g., declarative and functional) are currently out of scope. An example of an I2NSF Policy Rule is, in pseudo-code:

```
IF <event-clause> is TRUE

   IF <condition-clause> is TRUE

      THEN execute <action-clause>

   END-IF

END-IF
```

In the above example, the Event, Condition, and Action portions of a Policy Rule are all **Boolean Clauses**.

I2NSF Registry: a registry which contains I2NSF capability information that can be controlled by the controller. (An expansion of Registry definition below.)

IDS: Intrusion Detection System (see below).

IPS: Intrusion Protection System (see below).

Information Model: An information model is a representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol. (from [I-D.strassner-supa-generic-policy-info-model]).

Interface: is the set of operations one object knows it can invoke on another object. It is a subset of all operations that a given object implements. An example of multiple interfaces can be seen by considering the interfaces include a firewall uses. A firewall can have: a) multiple interfaces for data packets to traverse through and b) an interface for a controller to impose policy, or retrieve the results of execution of a policy rule. This illustrates that the same object may have multiple types of interfaces to serve different purposes.

Intrusion Detection System (IDS): a system which detects network intrusions via a variety of filters, monitors, and/or probes. An IDS may be stateful or stateless.

Intrusion Protection System (IPS): a system that protect against network intrusions. An IPS may be stateful or stateless.

Metadata: is data that provides information about other data. IETF network management protocols (e.g. NETCONF/RESTCONF/IPFix) or IETF routing interfaces (I2RS), and the I2NSF security interface may each utilize Metadata regarding the yang data models.

Middlebox: TBD

NSF: Network security function. An NSF is a function that that detects unwanted activity and blocks/mitigates the effect of such unwanted activity in order to support availability of a network. In addition, the NSF can help in supporting communication stream integrity and confidentiality.

OCL (the Object Constraint Language) is used to specify constraints in UML. (from http://www.ietf.org/mail-archive/web/i2nsf/current/msg00762.html)

OPNFV (Open Network Function Virtualization) TBD

Policy Rule: A Policy Rule is a set of rules that are used to manage and control the changing or maintaining of the state of one or more managed objects. Often this is shorterned to Rule or Policy. (from [I-D.strassner-supa-generic-policy-info-model]). An I2NSF Policy Rule is assumed to be in ECA form (i.e., an imperative structure). Other types of programming paradigms (e.g., declarative and functional) are currently out of scope. For the complete definition of an I2NSF Policy Rule please see above. (see above I2NSF policy rule).

Profile: A structured representation of information that characterizes the capabilities of an object. This may be used to simplify how this object interacts with other objects in its environment. [Editors note: John Strassner suggestse this is a simplified defintion from a variety of sources (UAProf and CC/PP). It does not mention the concept of preference, therefore John wonders if we need a different definition here.]

Registry: is a logically centralized location containing data of a particular type; it may optionally contain metadata, relationships, and other aspects of the registered data in order to use those data effectively. An I2NSF registry is used to contain capability information that can be controlled by the controller.

Registration Interface: is an interface dedicated to requesting, receiving, editing, and deleting information in a Registry.

Security Management System: TBD (Editorial: Placeholder fro split of definition betweeen controller (see above), and service provider security controller (see below) which existing I2NSF documents merge").

Server Layer: The Service Layer is called the Server Layer Interface in the I2NSF context.

Service Layer: The Service Layer (also called Client-Facing Interface) enables clients to manage security policies for their specific flows.

Service Provider Security Controller: TBD (Editorial: Place holder for a split between controller and security controller definition.)

Tenant: a tenant is a gorup of users that share common access proviliges to the same software. An I2NSF tenanat may be physical or virtual, and may run on a variety of systems or servers.

Vendor Facing Interface: The Vendor Facing Interface enables vendors to register their NSFs, along with the capabilities of their NSFs, with a logically centralized authority.

Virtual NSF: A NSF that is deployed as a distributed virtual device.

Virtual Network Function (VNF): A virtualized network component such as a router, switch, security box, or AAA Servier.

VNFM (VNF Manager): Manager of virtual network functions that creates, deletes, manages, and moves VNFs.

VNFPool: a collection of interchangeable VNFs (i.e., each VNF has the same set of capabilities).

Virtualization: Virtualization is a type of software that creates a non-physical version of an object. Examples include virtualized operating systems, storagte devices, and networking elements. [Editoris notes: Questions from John: Do we want or need to differentiate between different tyeps of virtualization? For example: full vs. partial vs. para-virtualization (all types of "hardware virtualization")? Do we need to introduce OS virtualization? What about application virtualization?]

3. IANA Considerations

No IANA considerations exist for this document.

4. Security Considerations

This is a terminology document with no security considerations.

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

5.2. Informative References

[I-D.ietf-i2nsf-gap-analysis]
Hares, S., Moskowitz, R., and D. Zhang, "Analysis of Existing work for I2NSF", draft-ietf-i2nsf-gap-analysis-00 (work in progress), February 2016.

[I-D.ietf-i2nsf-problem-and-use-cases]
Hares, S., Dunbar, L., Lopez, D., Zarny, M., and C. Jacquenet, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-00 (work in progress), February 2016.

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-06 (work in progress), December 2015.

[I-D.ietf-opsawg-firewalls]
Baker, F. and P. Hoffman, "On Firewalls in Internet Security", draft-ietf-opsawg-firewalls-01 (work in progress), October 2012.

[I-D.strassner-supa-generic-policy-info-model]
Strassner, J., Halpern, J., and J. Coleman, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-strassner-supa-generic-policy-info-model-04 (work in progress), February 2016.

[RFC4948] Andersson, L., Davies, E., and L. Zhang, "Report from the IAB workshop on Unwanted Traffic March 9-10, 2006", RFC 4948, DOI 10.17487/RFC4948, August 2007, <http://www.rfc-editor.org/info/rfc4948>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Phone: +1-734-604-0332
Email: shares@ndzh.com

John Strassner
Huawei
Santa Clara, CA
USA

Email: John.Strassner@huawei.com

Diego R. Lopex
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain

Email: diego.r.lopez@telefonica.com

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

Email: Frank.Xialiang@huawei.com

I2NSF WG
Internet-Draft
Intended status: Standards Track
Expires: August 5, 2016

S. Hares
R. Moskowitz
Huawei
D. Zhang

February 2, 2016

Analysis of Existing work for I2NSF
draft-ietf-i2nsf-gap-analysis-00.txt

Abstract

This document analyzes the status of the arts in industries and the existing IETF work/protocols that are 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.

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 August 5, 2016.

Table of Contents

1. Introduction

This documents provides a gap analysis for I2NSF.

1.1. What is I2NSF

The Network Security Function (NSF) in a network ensures integrity, confidentiality and availability of network communications, detects unwanted activity, and blocks out or at least mitigates the effects of unwanted activity. NSF devices are provided and consumed in increasingly diverse environments. For example, users of NSFs could consume network security services offered on multiple security products hosted one or more service provider,their own enterprises, or a combination of the two.

The lack of standard interfaces to control and monitor the behaviour of NSFs, makes it virtually impossible for security service providers to automate service offerings that utilize different security functions from multiple vendors.

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 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. The Service Layer is out of scope for this phase of I2NSF's work.

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).

1.2. Structure of this Document

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

IETF working groups (section 2)

ETSI Network Functions Virtualization Industry Specification Group (ETSI NFV ISG), (section 3)

OPNFV Open Source Group (section 4)

Open Stack - Firewall as a service (OpenStack Firewall FaaS) (section 5) (http://docs.openstack.org/admin-guide-cloud/content/install_neutron-fwaas-agent.html)

Cloud Security Alliance Security (CSA)as a Service (section 6) (https://cloudsecurityalliance.org/research/secaas/#_overview)

In-Depth Review of Some IETF Protocols (section 7)

1.3. Terms and Definitions

1.3.1. Requirements 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, BCP 14 [RFC2119] and indicate requirement levels for compliant CoAP.

1.3.2. Definitions

NSF: Network security function. An NSF is a function that that detects unwanted activity and blocks/mitigates the effect of such unwanted activity in order to support availability of a network. In addition, the NSF can help in supporting communication stream integrity and confidentiality.

Cloud Data Center (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.

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: is a security function that can be requested by one domain but may be owned or managed by another domain.

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.

2.1. Traffic Filters

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.

2.1.1.1. Data Flow Filters in NETMOD and I2RS

The current work on expanding these filters is focused oncombining 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.

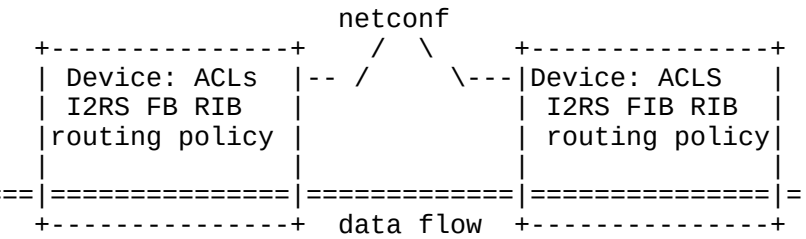


Figure 1

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].

One thing important to note is that NETCONF and RESTCONF manage device layer yang models. However, as figure 2 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).

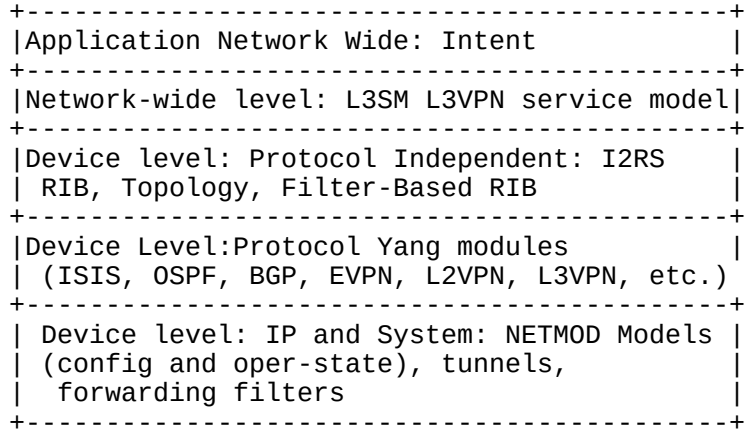


Figure 2 levels of Yang modules

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.

2.1.2. Middle-box Filters

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

2.1.3. Security Work

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 re-mediation. 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).

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 x.x.

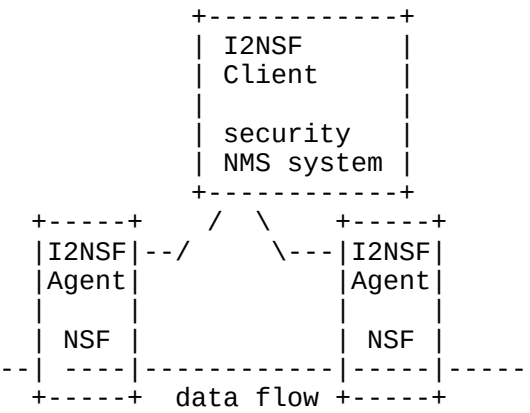


Figure 2

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

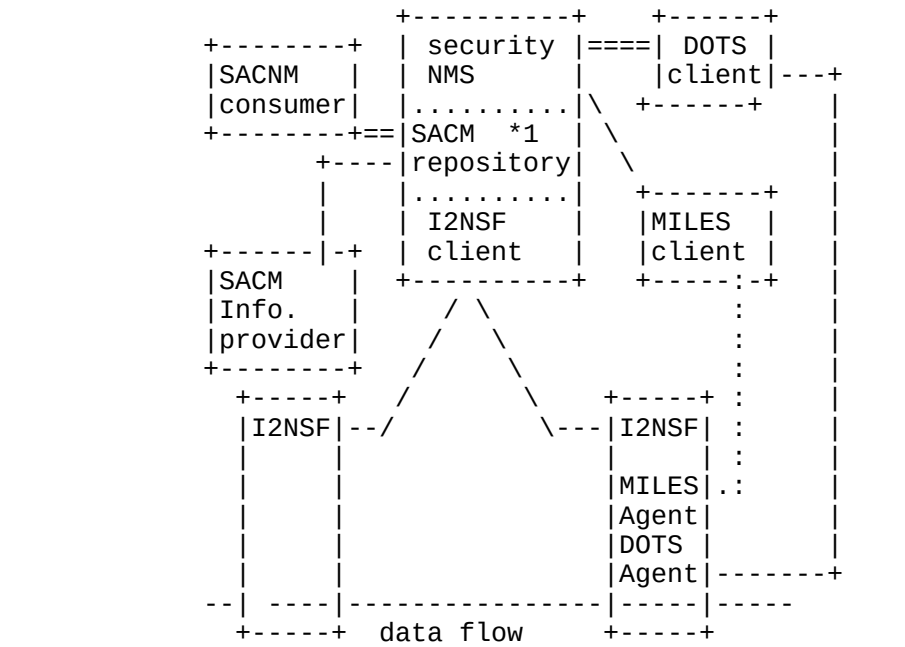
- SACM [I-D.ietf-sacm-architecture] describes an architecture which tries to determine if the end-point security policies and the reality (denoted as security posture) align. [I-D.ietf-sacm-terminology] defines posture as the configuration and/or status of hardware or software on an endpoint as it pertains 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.

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.



*1 - this is the SACM Controller (CR) with its broker/proxy/repository show as described in the SACM architecture.

Figure 3

Figure 3 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.

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.

3. ETSI NFV

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 along side 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 4 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.

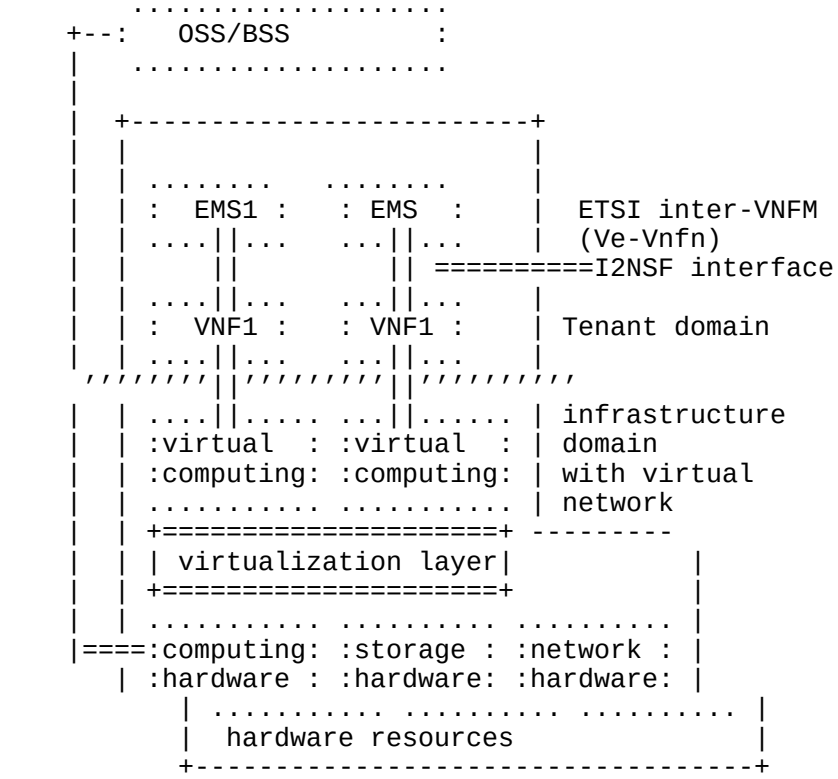


figure 4

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

o #16 - NFVIaas with Secure, SDN controlled WAN Gateway,

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.

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.

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 5 shows. This figure 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. Open Stack 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, the 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

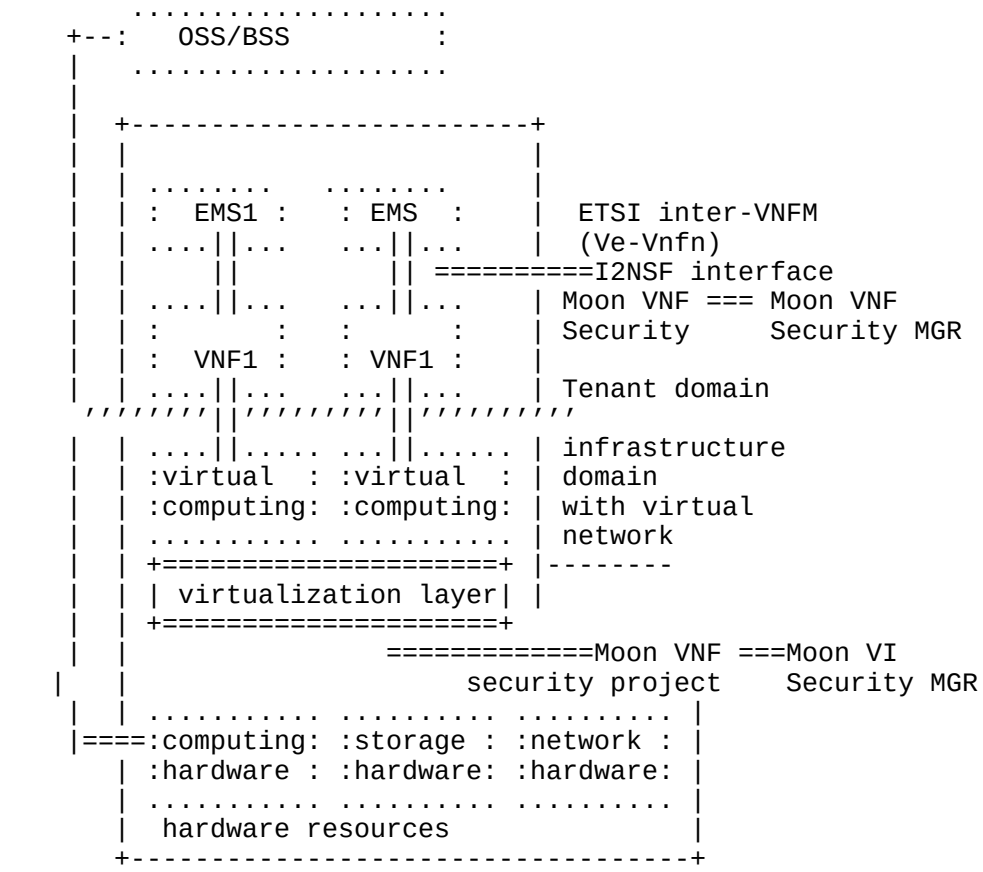


figure 5

4.2. Gap Analysis for OPNFV Moon Project

OpenStack congress does not provide vendor independent systems.

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

5.1. 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 OpenSTsack 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.

5.2. 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:

- identification of firewall (id, name, description),
- tenant associated with this firewall,
- administrative state (up/down),
- status (active, down, pending create, pending delete, pending update, pending error)
- firewall policy ID this firewall is associated with

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.

6. CSA Secure Cloud

6.1. CSA Overview

The Cloud Security Alliance (CSA)(www.cloudsecurityaliance.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

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)

2. Data Loss Prevention (DLP) (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_2_DLP_Implementation_Guidance.pdf)

3. Web Security (web) (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_3_Web_Security_Implementation_Guidance.pdf),

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

5. Security Assessments (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_5_Security_Assessments_Implementation_Guidance.pdf),

6. Intrusion Management (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_6_Intrusion_Management_Implementation_Guidance.pdf),

7. Security information and Event Management (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_7_SIEM_Implementation_Guidance.pdf),

8. Encryption (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_8_Encryption_Implementation_Guidance.pdf),

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

10. Network Security (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_10_Network_Security_Implementation_Guidance.pdf).

The sections below give an overview these implementation guidances

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 a problem. This 2012 implementation report confirms there is a gap with I2NSF

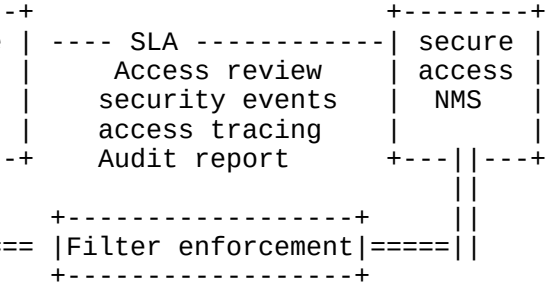


Figure 6

6.1.3. Data Loss Prevention (DLP)

Document:
(https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_2_DLP_Implementation_Guidance.pdf)

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

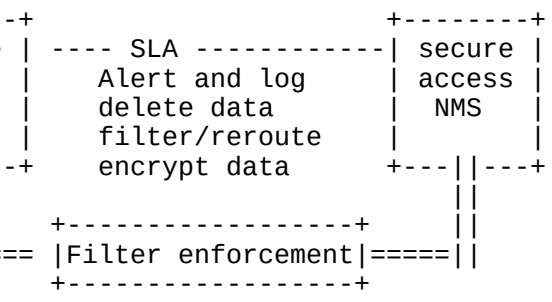


Figure 7

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.

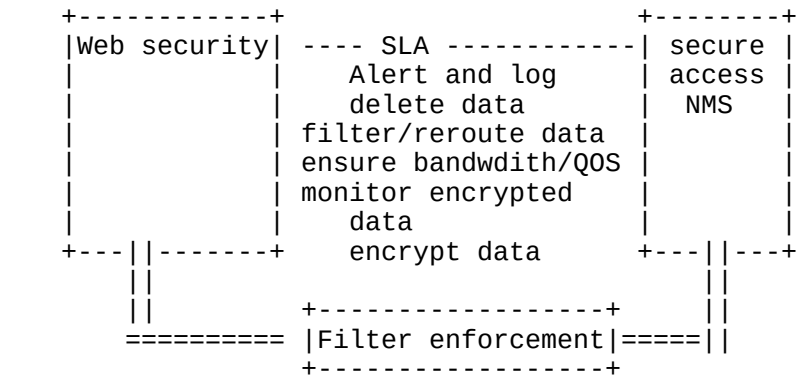


Figure 8

6.1.5. Email Security (email))

Document:
https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_4_Email_Security_Implementation_Guidance.pdf

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 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.

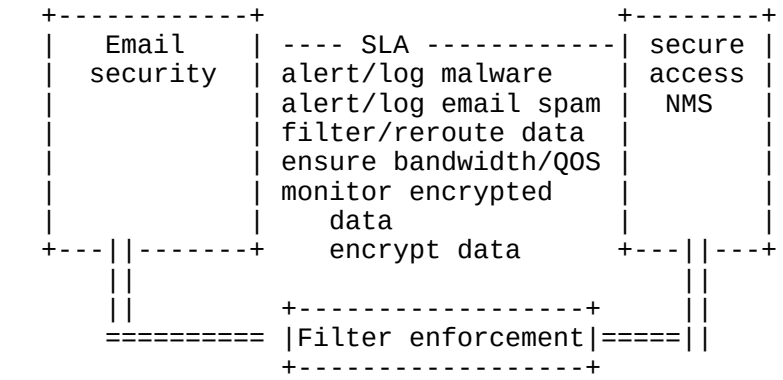


Figure 9

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:

o hypervisor infrastructure,

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

6.1.7. Intrusion Detection

Document: https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_6_Intrusion_Management_Implementation_Guidance.pdf)

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

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

Intrusion response includes both:

- Automatic, Manual, or Hybrid Mechanisms,
- Technical, Operational, and Process 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.

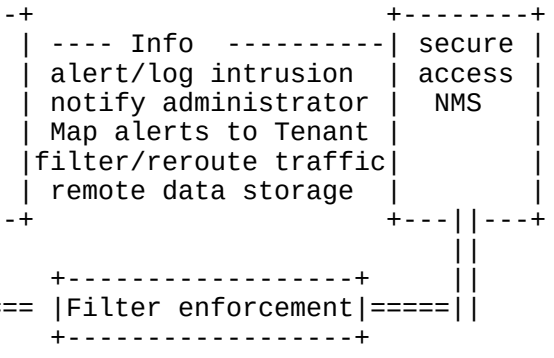


Figure 10

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/prevision (IDS/IPS)), encryption, 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.

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.

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 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

ISO/IEC 24762:2008, BS25999, ISO 27031, and FINRA Rule 4370

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 devices is extremely important to this application. Recovery as a Service (SaaS) indicates that these services need to be able to be remotely management.

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

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.

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.

7. In-depth Review of IETF protocols

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

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

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:

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

- o [I-D.ietf-i2rs-pub-sub-requirements] (Publication-Subscription notifications,

- o [I-D.ietf-i2rs-traceability]

- o [I-D.ietf-i2rs-protocol-security-requirements]

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.

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]).

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 1. The Policy decision point kept network-wide policy (E.g. ACLs) and sent it to Policy enforcements who 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.

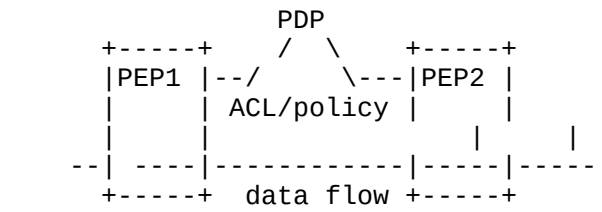


Figure 11

COPS had a design of Policy Enforcement Points (PEP), and policy Decision Points (PDP) as shown in figure 11. 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.

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]

[I-D.ietf-pcp-optimize-keepalives]

[I-D.ietf-pcp-proxy]

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

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 belief 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 is 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].

8. IANA Considerations

No IANA considerations exist for this document.

9. Security Considerations

No security considerations are involved with a gap analysis.

10. Contributors

The following people have contributed to this document: Hosnieh Rafiee.

11. References

11.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

11.2. Informative References

[I-D.hares-i2rs-bnp-eca-data-model]
Hares, S., Wu, Q., and R. White, "An Information Model for Basic Network Policy and Filter Rules", draft-hares-i2rs-bnp-eca-data-model-03 (work in progress), January 2016.

[I-D.hares-i2rs-info-model-service-topo]
Hares, S., Wu, W., Wang, Z., and J. You, "An Information model for service topology", draft-hares-i2rs-info-model-service-topo-03 (work in progress), January 2015.

[I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-11 (work in progress), December 2015.

[I-D.ietf-i2rs-ephemeral-state]
Haas, J. and S. Hares, "I2RS Ephemeral State Requirements", draft-ietf-i2rs-ephemeral-state-02 (work in progress), September 2015.

[I-D.ietf-i2rs-problem-statement]
Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-ietf-i2rs-problem-statement-09 (work in progress), January 2016.

[I-D.ietf-i2rs-protocol-security-requirements]
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", draft-ietf-i2rs-protocol-security-requirements-02 (work in progress), December 2015.

[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-04 (work in progress), January 2016.

[I-D.ietf-i2rs-rib-data-model]
Wang, L., Ananthakrishnan, H., Chen, M., amit.dass@ericsson.com, a., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-04 (work in progress), November 2015.

[I-D.ietf-i2rs-rib-info-model]
Bahadur, N., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-08 (work in progress), October 2015.

[I-D.ietf-i2rs-traceability]
Clarke, J., Salgueiro, G., and C. Pignataro, "Interface to the Routing System (I2RS) Traceability: Framework and Information Model", draft-ietf-i2rs-traceability-06 (work in progress), January 2016.

[I-D.ietf-i2rs-usecase-reqs-summary]
Hares, S. and M. Chen, "Summary of I2RS Use Case Requirements", draft-ietf-i2rs-usecase-reqs-summary-01 (work in progress), May 2015.

[I-D.ietf-i2rs-yang-l2-network-topology]
Dong, J. and X. Wei, "A YANG Data Model for Layer-2 Network Topologies", draft-ietf-i2rs-yang-l2-network-topology-02 (work in progress), December 2015.

[I-D.ietf-i2rs-yang-network-topo]
Clemm, A., Medved, J., Varga, R., Tkacik, T., Bahadur, N., and H. Ananthakrishnan, "A Data Model for Network Topologies", draft-ietf-i2rs-yang-network-topo-02 (work in progress), December 2015.

[I-D.ietf-isis-yang-isis-cfg]
Litkowski, S., Yeung, D., Lindem, A., Zhang, J., and L. Lhotka, "YANG Data Model for ISIS protocol", draft-ietf-isis-yang-isis-cfg-02 (work in progress), March 2015.

[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]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-04 (work in progress), January 2015.

[I-D.ietf-netconf-restconf-collection]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Collection Resource", draft-ietf-netconf-restconf-collection-00 (work in progress), January 2015.

[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]
Bogdanovic, D., Sreenivasa, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-02 (work in progress), March 2015.

[I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-19 (work in progress), May 2015.

[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]
Wasserman, M., Hartman, S., Zhang, D., and T. Reddy, "Port 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-02 (work in progress), October 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]
Shaikh, A., D'Souza, K., Bansal, D., and R. Shakir, "BGP Model for Service Provider Networks", draft-shaikh-idr-bgp-model-01 (work in progress), March 2015.

[I-D.shaikh-rtgwg-policy-model]
Shaikh, A., Shakir, R., D'Souza, K., and C. Chase, "Routing Policy Configuration Model for Service Provider Networks", draft-shaikh-rtgwg-policy-model-01 (work in progress), July 2015.

[I-D.tsingh-bess-pbb-evpn-yang-cfg]
Tiruveedhula, K., Singh, T., Sajassi, A., Kumar, D., and L. Jalil, "YANG Data Model for PBB EVPN protocol", draft-tsingh-bess-pbb-evpn-yang-cfg-00 (work in progress), March 2015.

[I-D.zhang-i2rs-l1-topo-yang-model]
Zhang, X., Rao, B., and X. Liu, "A YANG Data Model for Layer 1 Network Topology", draft-zhang-i2rs-l1-topo-yang-model-01 (work in progress), March 2015.

[I-D.zhdankin-idr-bgp-cfg]
Alex, A., Patel, K., Clemm, A., Hares, S., Jethanandani, M., and X. Liu, "Yang Data Model for BGP Protocol", draft-zhdankin-idr-bgp-cfg-00 (work in progress), January 2015.

[I-D.zhuang-bess-evpn-yang]
Zhuang, S. and Z. Li, "Yang Model for Ethernet VPN", draft-zhuang-bess-evpn-yang-00 (work in progress), December 2014.

[I-D.zhuang-bess-l3vpn-yang]
Zhuang, S. and Z. Li, "Yang Data Model for BGP/MPLS IP VPNs", draft-zhuang-bess-l3vpn-yang-00 (work in progress), December 2014.

[RFC2748] Durham, D., Ed., Boyle, J., Cohen, R., Herzog, S., Rajan, R., and A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, DOI 10.17487/RFC2748, January 2000, <http://www.rfc-editor.org/info/rfc2748>.

[RFC2940] Smith, A., Partain, D., and J. Seligson, "Definitions of Managed Objects for Common Open Policy Service (COPS) Protocol Clients", RFC 2940, DOI 10.17487/RFC2940, October 2000, <http://www.rfc-editor.org/info/rfc2940>.

[RFC3084] Chan, K., Seligson, J., Durham, D., Gai, S., McCloghrie, K., Herzog, S., Reichmeyer, F., Yavatkar, R., and A. Smith, "COPS Usage for Policy Provisioning (COPS-PR)", RFC 3084, DOI 10.17487/RFC3084, March 2001, <http://www.rfc-editor.org/info/rfc3084>.

[RFC3303] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, DOI 10.17487/RFC3303, August 2002, <http://www.rfc-editor.org/info/rfc3303>.

[RFC3304] Swale, R., Mart, P., Sijben, P., Brim, S., and M. Shore, "Middlebox Communications (midcom) Protocol Requirements", RFC 3304, DOI 10.17487/RFC3304, August 2002, <http://www.rfc-editor.org/info/rfc3304>.

[RFC3483] Rawlins, D., Kulkarni, A., Bokaemper, M., and K. Chan, "Framework for Policy Usage Feedback for Common Open Policy Service with Policy Provisioning (COPS-PR)", RFC 3483, DOI 10.17487/RFC3483, March 2003, <http://www.rfc-editor.org/info/rfc3483>.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, DOI 10.17487/RFC3484, February 2003, <http://www.rfc-editor.org/info/rfc3484>.

[RFC4080] Hancock, R., Karagiannis, G., Loughney, J., and S. Van den Bosch, "Next Steps in Signaling (NSIS): Framework", RFC 4080, DOI 10.17487/RFC4080, June 2005, <http://www.rfc-editor.org/info/rfc4080>.

[RFC4261] Walker, J. and A. Kulkarni, Ed., "Common Open Policy Service (COPS) Over Transport Layer Security (TLS)", RFC 4261, DOI 10.17487/RFC4261, December 2005, <http://www.rfc-editor.org/info/rfc4261>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <http://www.rfc-editor.org/info/rfc4949>.

[RFC5189] Stiemerling, M., Quittek, J., and T. Taylor, "Middlebox Communication (MIDCOM) Protocol Semantics", RFC 5189, DOI 10.17487/RFC5189, March 2008, <http://www.rfc-editor.org/info/rfc5189>.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <http://www.rfc-editor.org/info/rfc5277>.

[RFC5539] Badra, M., "NETCONF over Transport Layer Security (TLS)", RFC 5539, DOI 10.17487/RFC5539, May 2009, <http://www.rfc-editor.org/info/rfc5539>.

[RFC5973] Stiemerling, M., Tschofenig, H., Aoun, C., and E. Davies, "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", RFC 5973, DOI 10.17487/RFC5973, October 2010, <http://www.rfc-editor.org/info/rfc5973>.

[RFC6022] Scott, M. and M. Bjorklund, "YANG Module for NETCONF Monitoring", RFC 6022, DOI 10.17487/RFC6022, October 2010, <http://www.rfc-editor.org/info/rfc6022>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, DOI 10.17487/RFC6243, June 2011, <http://www.rfc-editor.org/info/rfc6243>.

[RFC6436] Amante, S., Carpenter, B., and S. Jiang, "Rationale for Update to the IPv6 Flow Label Specification", RFC 6436, DOI 10.17487/RFC6436, November 2011, <http://www.rfc-editor.org/info/rfc6436>.

[RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <http://www.rfc-editor.org/info/rfc6470>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC6639] King, D., Ed. and M. Venkatesan, Ed., "Multiprotocol Label Switching Transport Profile (MPLS-TP) MIB-Based Management Overview", RFC 6639, DOI 10.17487/RFC6639, June 2012, <http://www.rfc-editor.org/info/rfc6639>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <http://www.rfc-editor.org/info/rfc6887>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[RFC7225] Boucadair, M., "Discovering NAT64 IPv6 Prefixes Using the Port Control Protocol (PCP)", RFC 7225, DOI 10.17487/RFC7225, May 2014, <http://www.rfc-editor.org/info/rfc7225>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <http://www.rfc-editor.org/info/rfc7317>.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Bob Moskowitz
Huawei
Oak Park, MI 48237

Email: rgm@labs.htt-consult.com

Dacheng Zhang
Beijing
China

Email: dacheng.zdc@aliabab-inc.com

I2NSF
Internet-Draft
Intended status: Standards Track
Expires: August 5, 2016

S. Hares
L. Dunbar
Huawei
D. Lopez
Telefonica I+D
M. Zarny
Goldman Sachs
C. Jacquenet
France Telecom

February 2, 2016

I2NSF Problem Statement and Use cases
draft-ietf-i2nsf-problem-and-use-cases-00.txt

Abstract

This document describes the problem statement for Interface to Network Security Functions (I2NSF) and summary of the I2NSF use cases.

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 August 5, 2016.

Copyright Notice

Table of Contents

1. Introduction

This document describes the problem statement for Interface to Network Security Functions (I2NSF) and summary of the I2NSF use cases. A summary of the I2NSF state of the art in the industries and IETF which is relevant to I2NSF work is contained in [I-D.hares-i2nsf-gap-analysis].

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 may 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 also briefly describes the following use cases summarized by [I-D.pastor-i2nsf-merged-use-cases]:

- [I-D.pastor-i2nsf-access-usecases] (I2NSF-Access),
- [I-D.zarny-i2nsf-data-center-use-cases](I2NSF-DC), and
- [I-D.qi-i2nsf-access-network-usecase] (I2NSF-Mobile).

2. Terminology

ACL: Access Control List

B2B: Business-to-Business

Bespoke: Something made to fit a particular person, client or company.

Bespoke security management: Security management which is make to fit a particular customer.

DC: Data Center

FW: Firewall

IDS: Intrusion Detection System

IPS: Intrusion Protection System

NSF: Network security function. An NSF is a function that that detects unwanted activity and blocks/mitigates the effect of such unwanted activity in order to support availability of a network. In addition, the NSF can help in supporting communication stream integrity and confidentiality.

Flow-based NSF: A NSF which 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, or NAT inspection or changes).

Virtual NSF: A NSF which is deployed as a distributed virtual device.

VNFPool: Pool of Virtual Network Functions.

3. Problem Space

The following sub-section describe the problems and challenges facing customers and security service providers when some or all of the security functions are no longer physical hosted by the customer’s administrative domain.

Security service providers can be internal to the company or external security service providers. For example, an internal IT Security group within a large enterprise could act as a security service

provider for the enterprise. In contrast, an enterprise could outsource all security services to an external security service provider in a global service provider. In document, the security service provider function whether it is internal or external, will be denoted as "service provider".

The "Customer-Provider" relationship may be between any two parties. The parties can be in 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.

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

3.1. 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 and Attack Protection: Examples of this function are firewall/ACL authentication, IPS, IDS, and endpoint protection.

Security Functions in a DMZ: Examples of this function are firewall/ACLs, IDS/IPS, authentication and authorization services, NAT, forwarding proxies, application, and AAA services. These functions may be physically on-premise in a server provider's network at the DMZ spots or at "virtual" DMZ.

Internal Security Analysis and Reporting: Examples of this function are security logs, event correlation, and forensic analysis.

Internal Data and Content Protection: Examples of this function are encryption, authorization, and public/private key management for internal database.

Given the diversity of security functions, the contexts in which these functions can be deployed, and the constant evolution of these

functions, standardizing all aspects of security functions is challenging, and most probably not feasible. Fortunately, it is not necessary to standardize all aspects. 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 a standardized interface to control and monitor the rule sets that NSFs use to treat packets traversing through. And standardizing interfaces will provide an impetuous for standardizing established security functions.

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.

Since no widely accepted industry standard security interfaces exists today, management of NSFs (device and policy provisioning, monitoring, etc.) tends to be bespoke security management offered by product vendors. As a result, automation of such services, if it exists at all, is also bespoke. Thus, even in the traditional way of deploying security features, there is a gap to coordinate among implementations from distinct vendors. This is the main reason why mono-vendor security functions are often deployed and enable in a particular network segment.

A challenge for monitoring is that an NSF cannot monitor what it cannot view. Therefore, 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 and provide execution status of NSFs to security and compliance management tools. There exist various network security monitoring vendor specific interfaces for forensics and troubleshooting.

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. Therefore, 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 which 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, and network security functions (vNSF).

A 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 (see [I-D.jeong-i2nsf-sdn-security-services]), 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 in NSFs or vNSFs manufactured by multiple vendors. Even within one product category (e.g., firewall), security functions provided by different vendors can have different features and capabilities. For example, filters that can be designed and activated by a firewall may or may not support IPv6 depending on the firewall technology.

The service provider’s management system (or controller) needs a way to retrieve the capabilities of service functions by different vendors so that it could build an effective security solution. These service function capabilities can be documented in a static manner (e.g. a file) or via an interface which access a repository of security function capabilities which the NSF vendors dynamically update.

A 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 cannot 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, DOTS filters). Different policies might need different signatures or profiles. Today, the construction and use of black databases 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.

3.1.9. Lack of Mechanisms to accept external alerts to trigger automatic configuration changes

NSF can ask the I2NSF security controller to alter network policy. For example, a DDoS alert could trigger a change to routing system to send traffic to a traffic scrubbing service to mitigate the DDoS.

The DDoS protection has the following two parts: a) the configuration of signaling of open threats and b) DDoS mitigation. DOTS controller manages the signaling part of DDoS. I2NSF controller(s) would manage the changing to the network policy. By monitoring the network alerts from DDoS, I2NSF can feed a alerts analytics engine that could recognize attacks and the I2NSF can implement the needed new policies.

DDoS mitigation is enhanced if the provider's network security controller can monitor, analyze, and investigate the abnormal events and provide information to the client or change the network configuration (see section x) for details on the interfaces.

[I-D.zhou-i2nsf-capability-interface-monitoring] provides details on how monitoring aspects of the flow-based Network Security Functions (NSFs) can use the I2NSF interfaces to receive traffic reports and enforce policy.

3.1.10. Lack of mechanism for dynamic key distribution to NSFs

There is a need for controller to distribute various keys to distributed NSFs. To distribute various keys, the keys must be created and managed. While there is many key management methods and key derivation functions (KDF), there is a lack of standard interface to provision and manage keys.

The keys may be used for message authentication and integrity in order to protect data flow. In addition, keys may be used to secure the protocol and messages in the core routing infrastructure.

As of now there is no much focus on an abstraction for keying information that describes the interface between protocols, operators, and automated key management.

The keys may be used for message authentication and integrity in order to protect data flow. In addition, keys may be used to secure the protocol and messages in the core routing infrastructure.

The ability to utilize keys when routing protocols send or receive messages will be enhanced by having an abstract key table maintained by a security services. Conceptually, there must be an interface defined for routing/signaling protocols to make requests of automated key management when it is being used, to notify the protocols when keys become available in the key table.

An abstract key service needs to have three things:

1. I2NSF need to design the key table abstraction, the interface between key management protocols and routing/other protocols, and possibly security protocols at other layers.

2. For each routing/other protocol, I2NSF need to define the mapping between how the protocol represents key material and the protocol-independent key table abstraction. (If protocols share common mechanisms for authentication (e.g. TCP Authentication Option), then the same mapping may be reused.)

3. Automated Key management must support both symmetric keys and group keys via the service provided by items 1 and 2.

3.1.10.1. Background on Core Routing Security

A recommendation from a workshop held by the Internet Architecture Board (IAB) held a workshop on the topic of "Unwanted Internet Traffic" [RFC4948] suggest since a "simple risk analysis" suggests an "ideal attack target of minimal cost but maximal disruption is the core routing infrastructure", it is important to "tightening the security of the core routing infrastructure". One of the ways to tighten security of the core routing infrastructure is to tighten the security of protocol packets on the wire is by protecting the messages by use of keys.

Conceptually, when routing protocols send or receive messages, they might need to look up the key to use in this abstract key table.

Conceptually, there must be an interface defined for a protocols to make requests of automated key management when it is being used; when keys become available, they might be made available in the key table.

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 the security skills to express sufficiently precise requirements or security policies. Usually these customers express the expectations of their security requirements or the intent of their security policies. These expectations can be considered customer level security expectations. Customers may also desire to express guidelines for security management. Examples of such guidelines are the following:

- Which critical communications are to be preserved during critical events (DOTS),
- Which hosts are to continue service even during severe security attacks (DOTS),
- Reporting of attacks to CERT (MILE),
- Managing network connectivity of systems out of compliance (SACM),

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 deploy. These enterprises want to combine 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 will consist of NSFs on customer networks and NSFs on service provider networks.

3.2.2. Today’s Control Requests are Vendor Specific

Customers may consume NSFs by multiple service providers. Customers need to express their security requirements, guidelines, and expectations to the service providers. In turn, the service providers must translate this customer information into customer security policies and associated configuration sets for the set of security functions in their network. Without a standard technical

characterizations or a standard interface, the service provide faces many challenges.

Due the lack of standard technical characterizations and a standard interfaces, the following problems exists:

No standard technical characterization and/or APIs : Even the most common security services there is no standard technical characterization or APIs. Most security services are accessible only through disparate, proprietary interfaces (e.g., portals or APIs) in whatever format vendors choose to offer. The service provider must the customer’s input to these widely varying interfaces.

No standard interface: Without standard interfaces it is complex for customers to update security policies or integrate the security functions in their enterprise with the security services provided by the security service providers. This complexity is induced by the diversity of the configuration models, policy models, and supported management interfaces. Without a standard interface, new innovative security products find a large barrier to entry into the market

Managing by scripts de-jour: The current practices rely on the use of scripts which generate other scripts which the automatically run to upload or download configuration changes, log information and other things. These scripts have to be adjusted each time an implementation from a different vendor is enabled in a provider side.

Lack of immediate Feedback: Customers may also require a mechanism to easily update/modify their security requirements with immediate effect in the underlying involved NSFs.

Lack of explicit invocation request: 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.

To see 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.

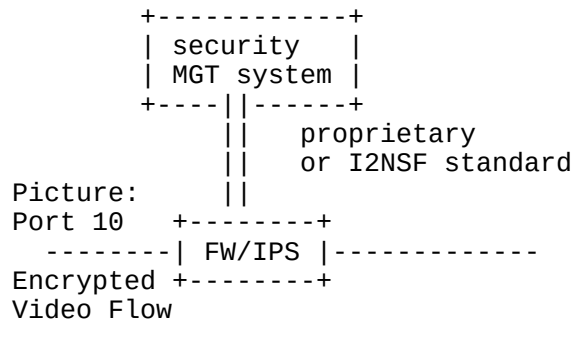


Figure 2: Example of non-standard vs. standard interface

In contrast, if a firewall/IPS interface standard exists, the customer would be able to send the request, without having to do the extensive preliminary legwork. A standard interface also helps service providers since they could now offer the same firewall/IPS interface to represent firewall/IPS services for utilizing products from many vendors. The result is that the service provider has now abstracted the firewall/IPS services. The standard interface also helps the firewall/IPS vendors to focus on their core security functions or extended features rather than the standard building blocks of a management interface.

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 results from the execution of their desired security requirements, guidelines and expectations.

Today, there is no standard way for customers to get security service assurance of their specified security policies properly enforced by the security functions in the provider domain. The customer also lacks the ability to perform "what-if" scenarios to assess the efficiency of the delivered security service.

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 the ability to express, monitor and verify security policies that combine several distributed security functions. It is crucial to an effective service to be able to take these actions via a standard interface. This standard interface becomes more crucial to the hosted security service when NSFs are instantiated in Virtual Machines which are sometimes widely

distributed in the network and sometimes are combined together in one device to perform a set of asks in a service.

Without standard interfaces and security policy data models, the enforcement of a customer-driven security policy remains challenging because of the inherent complexity created by the combining the invocation of several vendor-specific security functions into a multi-vendor, heterogeneous environment. Each vendor specific function may require specific configuration procedures and operational tasks.

Ensuring the consistent enforcement of the policies at various domains is also challenging. Standard data models are likely to contribute to ameliorating that issue.

3.4. Lack of Standard Interface to Inject Feedback to NSF

Today, many security functions, such as IPS, IDS, DDoS 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. Cyper Threat Alliance (CA, http://cyberthreatalliance.org/) is 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 cannot perform the policies requested by 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. Use Cases

Standard interfaces for monitoring and controlling the behavior of NSFs are essential building blocks for Security Service Providers and enterprises to automate the use of different NSFs from multiple vendors by their Security management entities. I2NSF may be invoked by any (authorized) client. Examples of authorized clients are upstream applications (controllers), orchestration systems, and security portals.

4.1. General Use Cases

User request security services through specific clients (e.g. a customer application, the NSP BSS/OSS or management platform) and the appropriate NSP network entity will invoke the (v)NSFs 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:

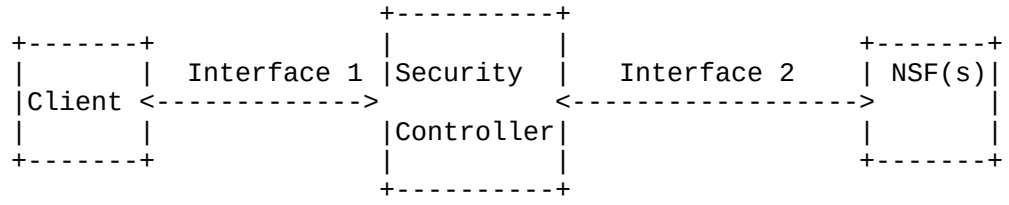


Figure 2: Interaction between Entities

Interface 1 is used for receiving security requirements from client and translating them into commands that NSFs can understand and execute. The security controller also passes back NSF security reports (e.g. statistics) to the client which the control has gathered from NSFs. Interface 2 is used for interacting with NSFs according to commands, and collect status information about NSFs.

Client devices or applications can require the security controller to add, delete or update rules in the security service function for their specific traffic.

When users want to get the executing status of security service, they can request the information of NSFs from the client. The security controller will collect NSF 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.

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

Integrity of the NSF: by ensuring that the NSF is not compromised;

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

In order to achieve this, the security controller may collect security measurements and share them with an independent and trusted third party (via the interface 1) in order to allow for attestation of NSF functions using the third party added information.

4.2. Access Networks

This scenario describes use cases for users (e.g. enterprise user, network administrator, and 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 requires a model of access virtualization that includes mobile and residential access where the operator may offload security services from the customer local environment (E.g. device or 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.

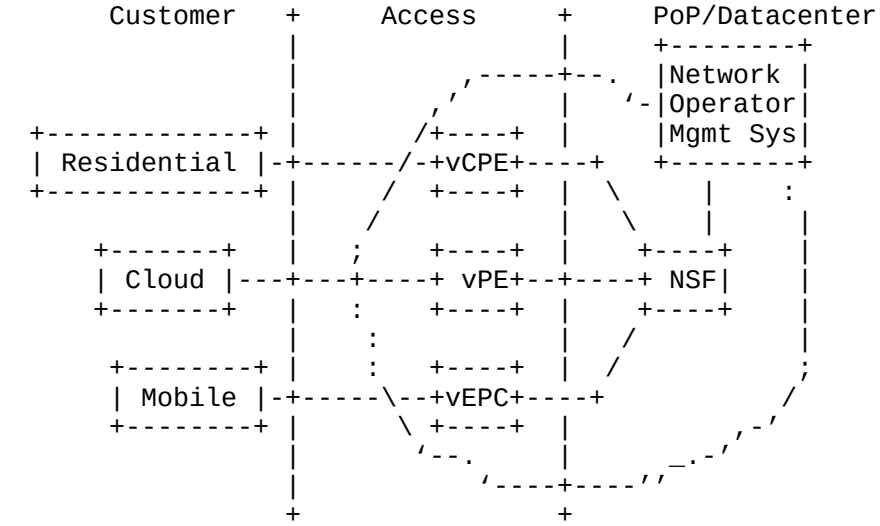


Figure 3: NSF and actors

The following are actions required for this access use case:

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.

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

* 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. These changes 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 on existing machines or provisioning a completely new firewall instance in a different machine.

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 requirement 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 in the right domain is of paramount importance, but also in 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. Another example is that IPS/IDS services for investment banking and non-banking traffic may be need to separated 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. Often it is 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.

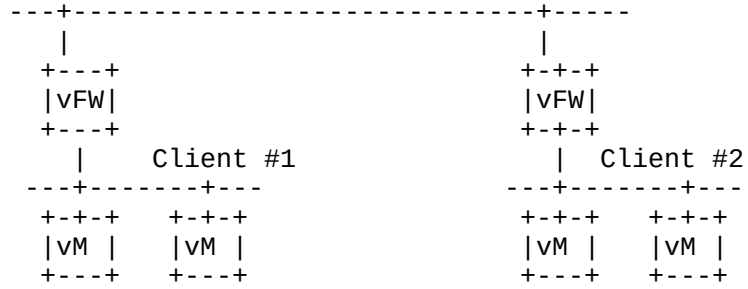


Figure 4: NSF in Data Center

4.3.2. Firewall Policy Deployment Automation

Firewall rules setting 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.

Firewall rules today are highly tied with ports and addresses of the traffic. This makes it very difficult for clients of cloud data center to construct rules for their own traffic as the clients only see the virtual networks and the virtual addresses. The customer-visible virtual networks and addresses may be different from the actual packets traversing the FWs.

Even though most vendors support similar firewall features, the actual rule configuration key words are different from vendors to

vendors, making it difficult for automation. Automation works best when it can leverage a common set of standards that will work across NSFs by multiple vendors. Without automation, it is virtually impossible for clients to dynamically specify their desired rules for their traffic.

4.3.3. 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 communication within the clients' own virtual networks as well as communication 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.

4.3.4. Internal network monitoring

There are many types of internal traffic monitors that may be managed by a security controller. This includes a new class of services referred to as DLP, Data Loss Prevention, or Reputation Protection Services. Depending on the class of event, alerts may go to internal administrators, or external services.

5. Management Considerations

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

6. IANA Considerations

No IANA considerations exist for this document.

7. Security Considerations

Having a secure access to control and monitor NSFs is crucial for hosted security service. The new NSF security controller introduces a new attack surface. It needs to be resilient to attack and recovery from attack needs to be quick and trivial (thus making attacking it 'uninteresting'). 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).

8. Contributors

I2NSF is a group effort. The following people contributed actively to the initial use case text: Xiaojun Zhuang (China Mobile), Sumandra Majee (F5), Ed Lopez (Fortinet), and Robert Moskowitz (Huawei).

9. Contributing Authors

I2NSF has had a number of contributing authors. The following are contributing authors:

- Antonio Pastur (Telefonica I+D),
- Mohamed Boucadair (France Telecom),
- Michael Georgiades (Prime Tel),
- Minpeng Qi (China Mobile),
- Shaibal Chakrabarty (US Ignite), and
- Nic Leymann (Deutsche Telekom).

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

10.2. Informative References

[Gartner-2013] Messmer, E., "Gartner: Cloud-based security as a service set to take off", October 2013.

[I-D.hares-i2nsf-gap-analysis] Hares, S., Zhang, D., Moskowitz, R., and H. Rafiee, "Analysis of Existing work for I2NSF", draft-hares-i2nsf-gap-analysis-01 (work in progress), December 2015.

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-06 (work in progress), December 2015.

[I-D.ietf-opsawg-firewalls]
Baker, F. and P. Hoffman, "On Firewalls in Internet Security", draft-ietf-opsawg-firewalls-01 (work in progress), October 2012.

[I-D.jeong-i2nsf-sdn-security-services]
Jeong, J., Kim, H., and P. Jung-Soo, "Requirements for Security Services based on Software-Defined Networking", draft-jeong-i2nsf-sdn-security-services-01 (work in progress), March 2015.

[I-D.lopez-i2nsf-packet]
Ed, E., "Packet-Based Paradigm For Interfaces To NSFs", draft-lopez-i2nsf-packet-00 (work in progress), March 2015.

[I-D.pastor-i2nsf-access-usecases]
Pastor, A. and D. Lopez, "Access Use Cases for an Open OAM Interface to Virtualized Security Services", draft-pastor-i2nsf-access-usecases-00 (work in progress), October 2014.

[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.

[I-D.qi-i2nsf-access-network-usecase]
Wang, K. and X. Zhuang, "Integrated Security with Access Network Use Case", draft-qi-i2nsf-access-network-usecase-02 (work in progress), March 2015.

[I-D.zarny-i2nsf-data-center-use-cases]
Zarny, M., Leymann, N., and L. Dunbar, "I2NSF Data Center Use Cases", draft-zarny-i2nsf-data-center-use-cases-00 (work in progress), October 2014.

[I-D.zhou-i2nsf-capability-interface-monitoring]
Zhou, C., Xia, L., Boucadair, M., and J. Xiong, "The Capability Interface for Monitoring Network Security Functions (NSF) in I2NSF", draft-zhou-i2nsf-capability-interface-monitoring-00 (work in progress), October 2015.

[RFC4948] Andersson, L., Davies, E., and L. Zhang, "Report from the IAB workshop on Unwanted Traffic March 9-10, 2006", RFC 4948, DOI 10.17487/RFC4948, August 2007, <http://www.rfc-editor.org/info/rfc4948>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Phone: +1-734-604-0332
Email: shares@ndzh.com

Linda Dunbar
Huawei
5340 Legacy Drive, Suite 175
Plano, TX 75024
USA

Phone: +1-734-604-0332
Email: ldunbar@huawei.com

Diego R. Lopex
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain

Email: diego.r.lopez@telefonica.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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 17, 2016

J. Jeong
H. Kim
Sungkyunkwan University
J. Park
ETRI
T. Ahn
S. Lee
Korea Telecom

March 16, 2016

Software-Defined Networking Based Security Services using Interface to Network Security Functions
draft-jeong-i2nsf-sdn-security-services-04

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 three use cases, such as a centralized firewall system, centralized DDoS-attack mitigation system, and centralized VoIP/VoLTE security system.

Table of Contents

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-OpenFlow][ONF-SDN-Architecture].

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 exisiting systems for security services. As feasible solutions to handle these challenges, this document proposes three use cases of the security services, such as a centralized firewall system, centralized DDoS-attack mitigation system, and centralized VoIP/VoLTE security system.

For the centralized firewall system, this document raises limitations in the 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 the 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.

For the centralized VoIP/VoLTE security system, this documents raises challenges in the legacy VoIP/VoLTE security system in terms of provisioning time, the granularity of security, cost, and the establishment of policy. This document shows a use case of SDN-based VoIP/VoLTE security system to resolve these challenges along in the I2NSF framework.

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-OpenFlow], [ONF-SDN-Architecture], [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].

- Access Control Policy Rules: Security policy rules concerning the provision of the access control service [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.

- Centralized VoIP/VoLTE Security System: A centralized security system that handles the security issues related to VoIP and VoLTE services. SDN can work as a network-based security system through a standard interface between VoIP/VoLTE security 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.

As shown in Figure 1, security functions as security services (e.g., firewall, DDoS-attack mitigation, VoIP/VoLTE, web filter, and 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.

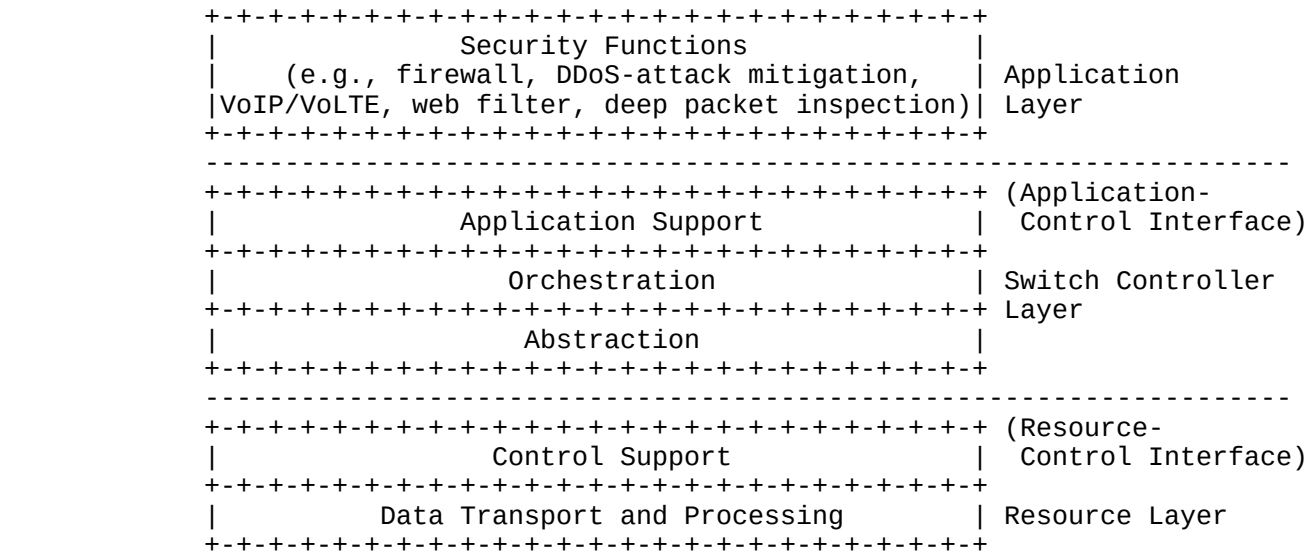


Figure 1: High-level Architecture for SDN-based Security Services

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]. Securiy 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.

o 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.

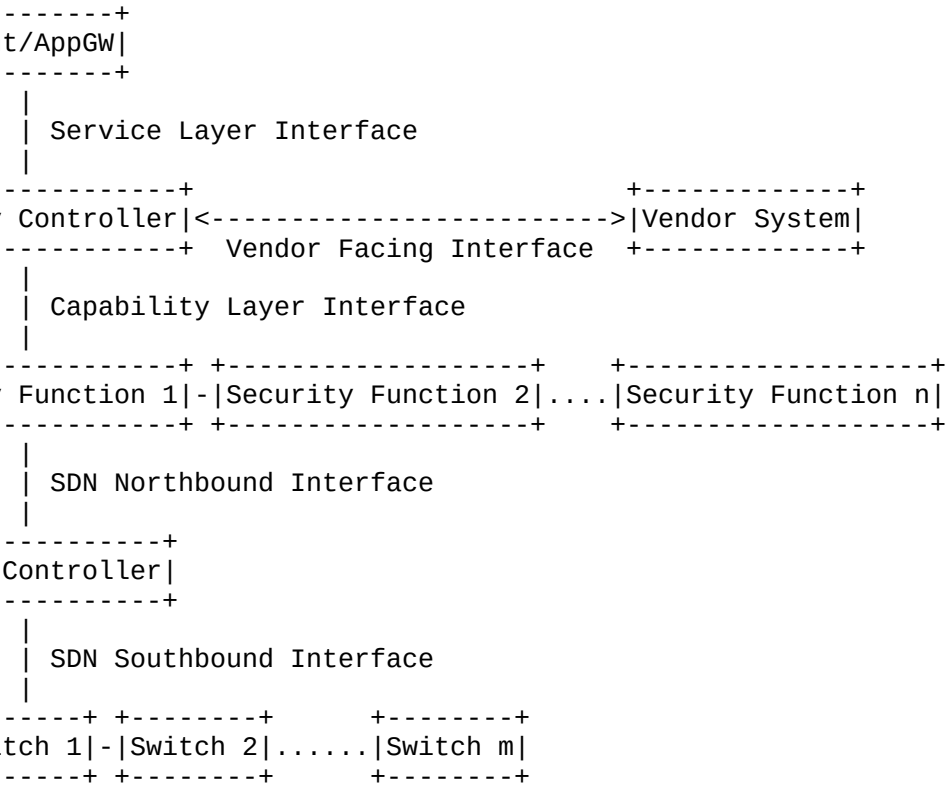


Figure 2: A Framework for SDN-based Security Services using I2NSF

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:

o SDN-based security services are required to support the programmability of network resources to mitigate network attacks.

o 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.

- SDN-based security services are required to support the seamless services to mitigate network attacks.

- SDN-based security services are required to provide the dynamic control of network resources to mitigate network attacks.

7. Use Cases

This section introduces three use cases for security services based on SDN: (i) centralized firewall system, (ii) centralized DDoS-attack mitigation system, and (iii) centralized VoIP/VoLTE security 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 Switch Controller.

2. Switch 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 Switch Controller.

5. Switch 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-OpenFlow] [ONF-SDN-Architecture].

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 managenent 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 Switch Controller.

2. Switch 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 Switch Controller.

5. Switch 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-OpenFlow][ONF-SDN-Architecture].

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.

7.3. Centralized VoIP/VoLTE Security System

For the centralized VoIP/VoLTE security system, a centralized VoIP/VoLTE security system can monitor each VoIP/VoLTE flow and manage VoIP/VoLTE security rules controlled by a centralized server for VoIP/VoLTE security service (called VoIP IPS). The VoIP/VoLTE security system controls each switch for the VoIP/VoLTE call flow management by manipulating the rules that can be added, deleted or modified dynamically.

The procedure of VoIP/VoLTE security operations in the centralized VoIP/VoLTE security system is as follows:

1. A switch forwards an unknown call flow’s signal packet (e.g., SIP packet) to Switch Controller. Also, if the packet belongs to a

matched flow's packet related to SIP (called matched SIP packet), Switch forwards the packet to Switch Controller so that the packet can be checked by a security function for VoIP (i.e., VoIP IPS) via Switch Controller, which monitors the behavior of its SIP call.

2. Switch Controller forwards the unknown flow's packet or the matched SIP packet to an appropriate security service function, such as VoIP IPS.

3. VoIP IPS analyzes the headers and contents of the signal packet, such as IP address, calling number, and session description [RFC4566].

4. If VoIP IPS regards the packet as a spoofed packet by hackers or a scanning packet searching for VoIP/VoLTE devices, it requests the Switch Controller to block that packet and the subsequent packets that have the same call-id.

5. Switch Controller installs new rules (e.g., drop packets) into switches.

6. The illegal packets are dropped by switches.

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

Legacy hardware based VoIP IPSes have some challenges, such as provisioning time, the granularity of security, expensive cost, and the establishment of policy. The proposed framework can resolve these challenges through the above centralized VoIP/VoLTE security system based on SDN as follows:

- Provisioning: The provisioning time of setting up a legacy VoIP IPS to network is substantial because it takes from some hours to some days. By managing the network resources centrally, VoIP IPS can provide more agility in provisioning both virtual and physical network resources from a central location.

- The granularity of security: The security rules of a legacy VoIP IPS are compounded considering the granularity of security. The proposed framework can provide more granular security by centralizing security control into a switch controller. The VoIP IPS can effectively manage security rules throughout the network.

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

- 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 VoIP IPS 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 document was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) [10041244, Smart TV 2.0 Software Platform] and by MSIP/IITP [R0166-15-1041, Standard Development of Network Security based SDN].

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

10. References

10.1. Normative References

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

[i2nsf-framework] Lopez, E., Lopez, D., Dunbar, L., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-merged-i2nsf-framework-02 , June 2015.

10.2. Informative References

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, March 2014.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[ITU-T.Y.3300] Recommendation ITU-T Y.3300, "Framework of Software-Defined Networking", June 2014.

[ONF-OpenFlow] ONF, "OpenFlow Switch Specification (Version 1.4.0)", October 2013.

[ONF-SDN-Architecture] ONF, "SDN Architecture", June 2014.

[ITU-T.X.1252] Recommendation ITU-T X.1252, "Baseline Identity Management Terms and Definitions", April 2010.

[ITU-T.X.800] Recommendation ITU-T X.800, "Security Architecture for Open Systems Interconnection for CCITT Applications", March 1991.

[AVANT-GUARD] Shin, S., Yegneswaran, V., Porras, P., and G. Gu, "AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-Defined Networks", ACM CCS, November 2013.

[ETSI-NFV] ETSI GS NFV 002 V1.1.1, "Network Functions Virtualisation (NFV); Architectural Framework", October 2013.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

Appendix A. Changes from draft-jeong-i2nsf-sdn-security-services-03

The following changes were made from draft-jeong-i2nsf-sdn-security-services-03:

- A new use case is added for a centralized VoIP/VoLTE security system.
- For the use case of the centralized VoIP/VoLTE security system, two new requirements are added. First, SDN-based security services are required to support the seamless services to mitigate

network attacks. Second, SDN-based security services are required to provide the dynamic control of network resources to mitigate network attacks.

Authors' Addresses

Jaehoon Paul Jeong
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4957
Fax: +82 31 290 7996
EMail: pauljeong@skku.edu
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

Hyoungshick Kim
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
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

Tae-Jin Ahn
Korea Telecom
70 Yuseong-Ro, Yuseong-Gu
Daejeon 305-811
Republic of Korea

Phone: +82 42 870 8409
EMail: taejin.ahn@kt.com

Se-Hui Lee
Korea Telecom
70 Yuseong-Ro, Yuseong-Gu
Daejeon 305-811
Republic of Korea

Phone: +82 42 870 8162
EMail: sehuilee@kt.com

Network Working Group
Internet Draft
Intended status: Informational
Expires: September 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

March 16, 2016

Framework for Interface to Network Security Functions
draft-merged-i2nsf-framework-05.txt

Status of this Memo

This Internet-Draft will expire on September 16, 2016.

Copyright Notice

Abstract

This document defines the framework for guiding the functionality provided by I2NSF. Network security functions (NSFs) are packet-processing engines that inspect and optionally modify packets traversing networks, either directly or in the context of sessions in which the packet is associated. This document provides an overview of how NSFs are used, and describes how NSF software interfaces are controlled and monitored using rulesets. The design of these software interfaces must prevent the creation of implied constraints on NSF capability and functionality.

Table of Contents

1. Introduction

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

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. I2NSF realizes this as a set of security rules for monitoring and controlling the behavior of their specific traffic. It also provides standard interfaces 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 have 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 defined by other protocols, such as SYSLOG and DOTS.

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- or flow-based paradigm. A common trait of NSFs is in the processing of packets based on the content (header/payload) and/or context (session state, authentication state, etc) of the received packets.

An important concept underlying this framework 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, as this enables 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/flow-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 standardized by using Event - Condition - Action (ECA) policy rulesets.

An Event, when used in the context of policy rules for a flow-based NSF, is used to determine whether the condition clause of the Policy Rule can be evaluated or not. Here are some examples of I2NSF Events:

- defining a clause, of the canonical form {variable, operator, value}, to represent an Event (e.g., time == 08:00)
- using an Event object as the variable or the value in the above clause (e.g., use one or more attributes from one or more Event objects in the comparison clause)
- using a Collection object to collect Events for aggregation, filtering, and/or correlation operations as part of the Event clause processing
- encoding the entire Event expression into an attribute

A Condition, when used in the context of policy rules for flow-based NSFs, is used to determine whether or not the set of Actions in that Policy Rule can be executed or not. A condition can be based on various combinations of the content (header/payload) and/or the context (session state, authentication state, etc) of the received packets:

- Packet content values are based on one or more packet headers, data from the packet payload, bits in the packet, or something derived from the packet;
- Context values are based on measured and inferred knowledge that define the state and environment in which a managed entity exists or has existed. In addition to state data, this includes data from sessions, direction of the traffic, time, and geo-location information. State refers to the behavior of a managed entity at a particular point in time. Hence, it may refer to situations in which multiple pieces of information that are not available at the same time must be analyzed. For example, tracking established TCP connections (connections that have gone through the initial three-way handshake).

Actions for flow-based NSFs include:

- Action ingress processing, such as pass, drop, mirroring, etc;
- Action egress processing, such as invoke signaling, tunnel encapsulation, packet forwarding and/or transformation;
- Applying a specific Functional Profile or signature - e.g., an IPS Profile, a signature file, an anti-virus file, or a URL filtering file. Many flow-based NSFs utilize profile and/or signature files to achieve more effective threat detection and prevention. It is not uncommon for a NSF to apply different profiles and/or signatures for different flows. Some profiles/signatures do not require any knowledge of past or future activities, while others are stateful, and may need to maintain state for a specific length of time.

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. The rulesets and software interfaces of I2NSF aim to standardize the form and function of profile and signature files while supporting vendor-specific functions of each.

4. Reference Models in Managing Flow-based NSFs

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

The following figure shows various interfaces for managing the provisioning & monitoring aspects of flow-based NSFs.

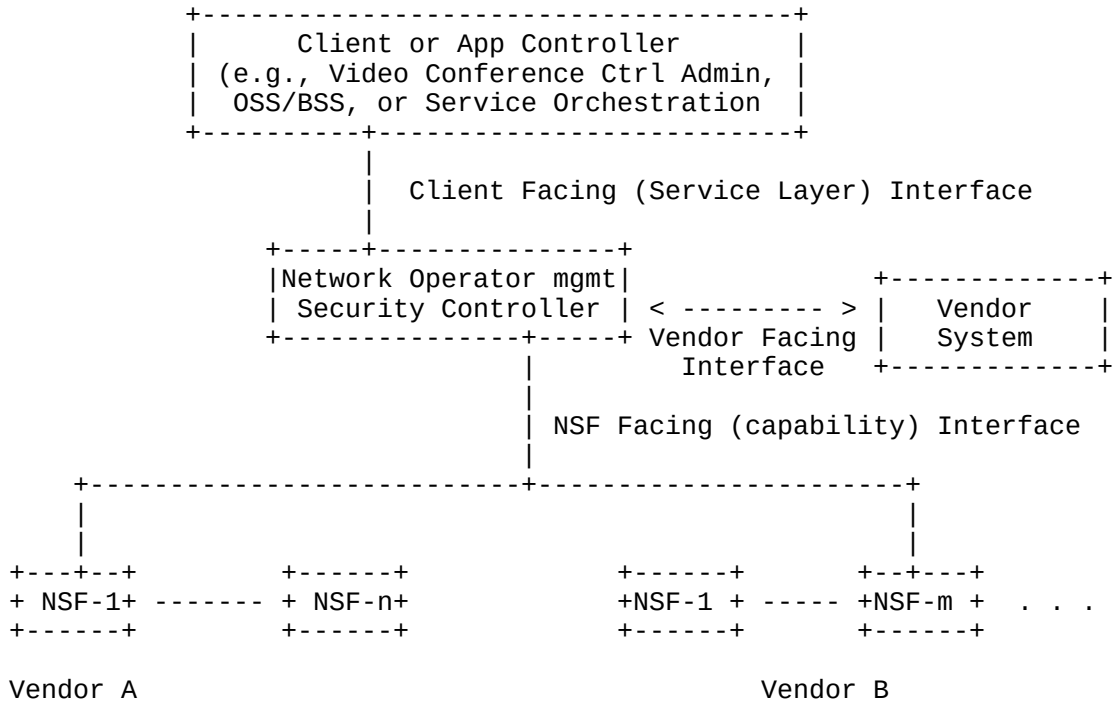


Figure 1: Multiple Interfaces

4.1. NSF Facing (Capability Layer) Interface

This is the interface between the Service Provider's management system (or Security Controller) and the set of NSFs that are selected to enforce the desired network security. This interface defines the features available for each NSF that the management system can choose to invoke for a particular packet or flow. Note that the management system does not need to use all features for a given NSF, nor does it need to use all available NSFs. Hence, this abstraction enables the same relative features from diverse NSFs from different vendors to be selected.

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 Service Layer Interface in the I2NSF 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 multiple instantiations of the same NSF) to achieve the desired enforcement.

4.3. Vendor Facing Interface

NSFs provided by different vendors have different capabilities. In order to automate the process of utilizing 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 a new functionality that is exposed to the user is added to an NSF, the vendor MUST notify the network operator’s management system or security controller of its updated functionality via the Registration 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 network connection between the 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 secure external network domains. This requires more restrictive security control to be placed over the I2NSF interface. Not only must the information over the I2NSF interfaces use trusted channels, such as TLS, SASL (RFC4422), or the combination of the two, but also require proper authentication as described in [Remote-Attestation].

  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 that has been distributed across a network. When different instantiations are visible to the Security Controller, different

policies may be applied to different instantiations of an individual 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.

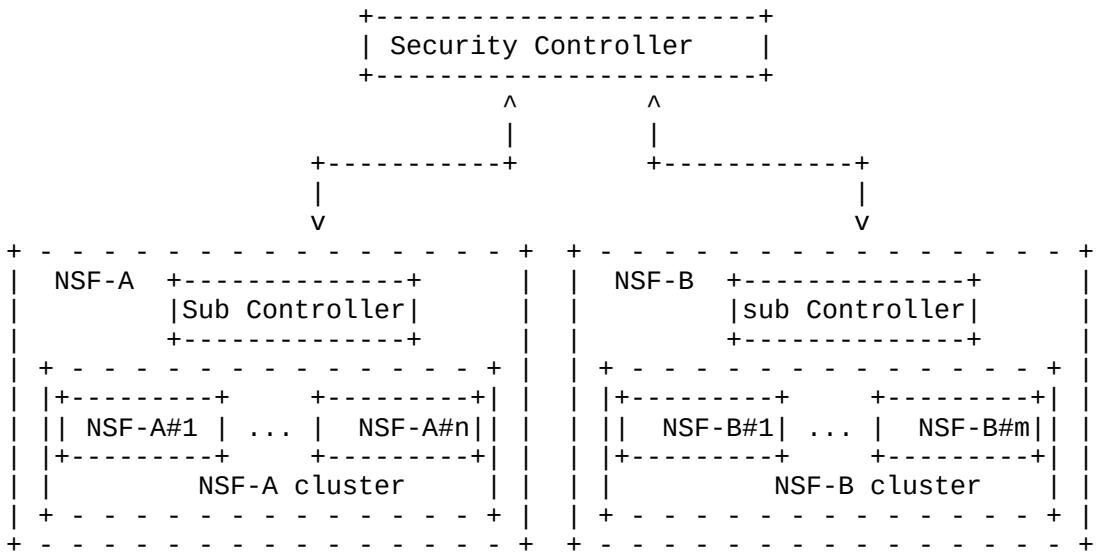


Figure 2: Cluster of NSF Instantiations Management

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 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.
. IPS (Intrusion Prevention System) - A device or function that analyzes 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 in 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 Packet Content Match categories. The IDR Flow Specification [RFC5575] has specified 12 different packet header matching types. More packet content matching types have been proposed in the IDR WG. I2NSF should re-use the packet matching types being specified as much as possible. More matching types might be added for Flow-based NSFS. Tables 1-4 below list the applicable packet content categories that can be potentially used as packet matching types by Flow-based NSFs:

Packet Content Matching Capability Index	
Layer 2 Header	Layer 2 header fields: Source/Destination/s-VID/c-VID/EtherType/.
Layer 3	Layer header fields:
IPv4 Header	protocol dest port src port src address dest address dscp length flags ttl
IPv6 Header	addr protocol/nh src port dest port src address dest address length traffic class hop limit flow label dscp
TCP SCTP DCCP	Port syn ack fin rst ? psh

		? urg
		? window
		sockstress
		Note: bitmap could be used to
		represent all the fields
UDP		
		flood abuse
		fragment abuse
		Port
HTTP layer		
		hash collision
		http - get flood
		http - post flood
		http - random/invalid url
		http - slowloris
		http - slow read
		http - r-u-dead-yet (rudy)
		http - malformed request
		http - xss
		https - ssl session exhaustion
IETF PCP	Configurable Ports	
IETF TRAM	profile	

Table 1: Subject Capability Index

context	matching Capability Index
Session	Session state, bidirectional state
Time	time span, time occurrence
Events	Event URL, variables
Location	Text string, GPS coords, URL
Connection Type	Internet (unsecured), Internet (secured by VPN, etc.), Intranet, ...

Direction	Inbound, Outbound
State	Authentication State
	Authorization State
	Accounting State
	Session State

Table 2: Object Capability Index

Action Capability Index	
Ingress port	SFC header termination, VxLAN header termination
Actions	Pass Deny Mirror Simple Statistics: Count (X min; Day;..) Client specified Functions: URL
Egress	Encap SFC, VxLAN, or other header

Table 3: Action Capability Index

Functional profile Index	
Profile types Signature	Name, type, or Flexible Profile/signature URL Command for Controller to enable/disable

Table 4: Function Capability Index

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 "packet content matching" 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.

Some Flow-based NSFs need matching criteria that include the context associated with the packets.

The I2NSF "actions" should extend the actions specified by [ACL-MODEL] to include applying statistics functions, threat profiles, or signature files that clients provide.

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

6.2. Service Layer Policy

This layer is for clients or an Application Controller to express and 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 instead express expectations or intent of the functionality desired by their security policies. Customers may also express guidelines such as which certain types of destinations are not allowed for certain groups. As a result, there could be different 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

- o enable "school protection control"
- o allow Internet traffic from 8:30 to 20:00
- o scan email for malware detection protect traffic to corporate network with integrity and confidentiality
- o remove tracking data from Facebook [website = *.facebook.com]
- o 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 clients or Application controllers when clients' service requirements have been changed. Some service layer policies may not be granted because the carrier or Enterprises imposes additional constraints on what a client 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 the I2NSF capability layer, but with more of a client-oriented expression for the packet content, context, and other parts of an ECA policy rule. This enables the client to construct an ECA policy rule without having to know its detailed structure or syntax.

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, Event, 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 Event-Condition-Action paradigm.

The I2NSF simple service layer can have the following entities:

- I2NSF-Groups: This is a collection of users, applications, virtual networks, or traffic patterns to which a service

layer policy can be applied. An I2NSF-Group 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-Group can have one or more Policy Rules applied to it.

- Target. This is used by the application client to identify the set of objects to be affected by the policy rules. A Target can be mapped to a physical/logical ingress port, a set of destinations, or a physical/logical egress port.
- Policy Rule. A Policy Rule consists of a set of Policy Events, Policy Conditions, and Policy Actions. Policy Rules are triggered by matching Events. If the Event portion of the Policy Rule evaluates to true, then the Condition portion is evaluated (otherwise, the Policy Rule terminates and no action is taken). If the Condition portion of the Policy Rule evaluates to true, then the set of Actions MAY be executed and applied to the traffic (otherwise, the Policy Rule terminates and no action is taken).
- Policy Event. This triggers a determination of whether the condition portion of a Policy Rule should be evaluated or not.
- Policy Condition. This determines when the Policy Actions contained in a Policy Rule are 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 (and was triggered by the Events). 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.

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 requirements (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 users and applications

  - 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 NSFs
- 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

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

[RFC3060] Moore, B, et al, "Policy Core Information Model (PCIM)", RFC 3060, Feb 2001.

[RFC3460] Moore, B. "Policy Core Information Model (PCIM) Extensions", RFC3460, Jan 2003.

[RFC5575] Marques, P, et al, "Dissemination of Flow Specification Rules", RFC 5575, Aug 2009.

[RFC7297] Boucadair, M., "IP Connectivity Provisioning Profile", RFC7297, April 2014.

12.2. Informative References

[I2NSF-ACCESS] A. Pastor, et al, "Access Use Cases for an Open OAM Interface to Virtualized Security Services", <draft-pastor-i2nsf-access-usecases-00>, Oct 2014.

[I2NSF-DC] M. Zarny, et al, "I2NSF Data Center Use Cases", <draft-zarny-i2nsf-data-center-use-cases-00>, Oct 2014.

[I2NSF-MOBILE] M. Qi, et al, "Integrated Security with Access Network Use Case", <draft-qi-i2nsf-access-network-usecase-00>, Oct 2014

[I2NSF-Problem] L. Dunbar, et al "Interface to Network Security Functions Problem Statement", <draft-dunbar-i2nsf-problem-statement-01>, Jan 2015

[ACL-MODEL] D. Bogdanovic, et al, "Network Access Control List (ACL) YANG Data Model", <draft-ietf-net-acl-model-00>, Nov 2014.

[gs_NFV] ETSI NFV Group Specification, Network Functions Virtualizsation (NFV) Use Cases. ETSI GS NFV 001v1.1.1, 2013.

[NW-2011] J. Burke, "The Pros and Cons of a Cloud-Based Firewall", Network World, 11 November 2011

[SC-MobileNetwork] W. Haeffner, N. Leymann, "Network Based Services in Mobile Network", IETF87 Berlin, July 29, 2013.

[I2NSF-Demo] Y. Xie, et al, "Interface to Network Security Functions Demo Outline Design", <draft-xie-i2nsf-demo-outline-design-00>, April 2015.

[ITU-T-X1036] ITU-T Recommendation X.1036, "Framework for creation, storage, distribution and enforcement of policies for network security", Nov 2007.

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
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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

R. Moskowitz
HTT Consulting
S. Hares
L. Xia
Huawei
March 21, 2016

Security alerts over the first MILE
draft-moskowitz-firstmile-00.txt

Abstract

This document describes a pub/sub styled protocol to send security alerts to a security monitor that can feed into MILE and other management platforms. It uses data structures from NETCONF, MILE, and IPFIX to manage the reporting and report security alerts.

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 September 22, 2016.

Copyright Notice

Table of Contents

1. Introduction

This document proposes a set of protocols to automate the reporting of security alerts to the various monitoring systems. The intent is primarily to automate the input of security events to the MILE environment (RID [RFC6545] and IODEF [I-D.ietf-mile-rfc5070-bis]). Any authorized monitoring system can subscribe to any of the security alerts reports.

An Internet security defense device first registers with a security alert monitoring system. At this point the content and protocol used has not been identified. Since such a registration is normally at 'quiet time', the registration does not occur during a network congested time and can use some HTTPS-based service. At this time both systems exchange their X.509 identifiers to be used for the sub/pub security and identification.

Once a defense device is registered, the monitoring system can subscribe to it for those alerts in needs to receive. The subscription protocol should use NETCONF [RFC6536] with the publication/subscription push service [I-D.ietf-netconf-yang-push]. If the system needs a "pull" service, the NETCONF and I2RS subscription service could be expanded to support a pull service.

Any secure NETCONF transport that this pub/sub service support can be used.

The defense device publishes security alerts to subscribed monitors using IODEF or IPFIX [RFC7011] data structures. The protocol(s) for these reports are discussed within this document.

2. Terms and Definitions

2.1. Requirements 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].

3. Problem Space

At the time of developing this document, there is no IETF defined set of standardized security alert messages and protocols. Administrators of systems which provide MILE service currently use "cut-and-past" where they cut selected messages from proprietary monitoring systems and past these messages into their MILE environment. The intent here is to standardize and automate this process. It is recognized that many of these alerts are too detailed to be actionable. Some implementations of the alert monitor will include analytic tools to select the actionable information from the alerts. Alerts which are too detailed to be actionable or alerts which include analytical tools are outside of any standardizing process.

Many of the needed alerts are scattered throughout the various standards like IPFIX and IODEF, but are not collected together as recognized security alerts that should be aggregated into a reporting framework.

4. The first mile of security alerts

There are three components to the first MILE process

- Register
- Subscribe
- Publish

4.1. Register

An Internet security defense device first registers with a security alert monitoring system. This is typically done at the time the device is installed, but may occur later as the device is registered to more monitoring systems. There is no theoretical limit on the

number of monitors a device is registered to. The limit within a system are practical limits based on internal limits within the device.

Most monitors will be commercial and the registration will be based on existing business relationships. One such example is the ISP's security monitor. It is possible that a CERT may accept direct registration without a business relationship. However this may require more study to ensure that this will not introduce potential attacks of false reporting to CERTs.

The actual content of the registration has not been determined. Minimally it needs to include

- Identifiers (e.g. X.509 certificates)
- Reports available from device (i.e. what to subscribe to)
- Subscription protocols(s)
- Publication protocols(s)

A device can alter any of its registered information at any time as well as cancel a registration.

4.2. Subscribe

Once a defense device is registered, the monitoring system can subscribe to it for those alerts in needs to receive. This is typically done via NETCONF, but is controlled by what the device registered as supported subscript protocols.

A monitor can subscribe or unsubscribe for reports at any time. With the first subscription, a secure communication transport will be enabled from the device to the monitor. See Section 4.3 for more on the this secure transport.

4.3. Publish

The defense device publishes security alerts to subscribed monitors. The reports will be sent over the subscribed protocol using the subscribed data model, either IODEF or IPFIX.

Since these alerts may be reported during an attack that degrades communications, many of the DOTS requirements [I-D.ietf-dots-requirements] apply here. One that doesn't is the bi-directional requirement. Even so, the same security and transport design used for DOTS should be used here.

5. first MILE data model

The data model will support the constraints of the NETCONF publication/subscription model [I-D.ietf-netconf-yang-push], and the NETCONF module library function [I-D.ietf-netconf-yang-library] which indicates pub/sub support within a model. If the MILE service which to utilize non-persistent (aka ephemeral) data that disappears on reboot, the netconf publication/subscription model will support non-persistent configuration.

Work on the data model is an open item.

6. IANA Considerations

No IANA considerations exist for this document at this time.

7. Security Considerations

An attacker that can disable first MILE may be able to attack a device at will as those monitoring it expect these attacks to show up on their monitor. As such each part of the firstMILE system will need the complete security services that are defined or referenced here.

8. Contributors

TBD

9. References

9.1. Normative References

[I-D.ietf-netconf-yang-library]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", draft-ietf-netconf-yang-library-04 (work in progress), February 2016.

[I-D.ietf-netconf-yang-push]
Clemm, A., Prieto, A., Voit, E., Tripathy, A., and E. Einar, "Subscribing to YANG datastore push updates", draft-ietf-netconf-yang-push-01 (work in progress), February 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

9.2. Informative References

[I-D.ietf-dots-requirements]
Mortensen, A., Moskowitz, R., and T. Reddy, "DDoS Open Threat Signaling Requirements", draft-ietf-dots-requirements-00 (work in progress), October 2015.

[I-D.ietf-mile-rfc5070-bis]
Danyliw, R., "The Incident Object Description Exchange Format v2", draft-ietf-mile-rfc5070-bis-17 (work in progress), March 2016.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC6545] Moriarty, K., "Real-time Inter-network Defense (RID)", RFC 6545, DOI 10.17487/RFC6545, April 2012, <http://www.rfc-editor.org/info/rfc6545>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <http://www.rfc-editor.org/info/rfc7011>.

Authors' Addresses

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237

Email: rgm@labs.htt-consult.com

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Liang Xia
Huawei
No. 101, Software Avenue, Yuhuatai District
Nanjing
China

Email: Frank.xialiang@huawei.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 5, 2017

A. Pastor
D. Lopez
Telefonica I+D
A. Shaw
Hewlett Packard Labs
July 4, 2016

Remote Attestation Procedures for Network Security Functions (NSFs) through the I2NSF Security Controller draft-pastor-i2nsf-vnsf-attestation-03

Abstract

This document describes the procedures a client can follow to assess the trust on an external NSF platform and its client-defined configuration through the I2NSF Security Controller. The procedure to assess trustworthiness is based on a remote attestation of the platform and the NSFs running on it performed through a Trusted Platform Module (TPM) invoked by the Security Controller.

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 January 5, 2017.

Table of Contents

1. Introduction

As described in [I-D.pastor-i2nsf-merged-use-cases], the use of externally provided NSF implies several additional concerns in security. The most relevant threats associated with a externalized virtual platform are detailed in [I-D.ietf-i2nsf-framework]. As stated there, mutual authentication between the user and the NSF environment and, what is more important, the attestation of the elements in this environment by clients could address these threats to an acceptable level of risk. In particular:

- Any impersonation attempt (of the client or the NSF environment) will be minimized by mutual authentication, and since appropriate records of such authentications will be made available, events will be suitable for auditing in the case of an incident.

- Attestation of the NSF environment, especially when performed periodically, will allow clients to detect the alteration of the processing elements, or the installation of malformed elements, and mutual authentication will provide again an audit trail.

- Attestation relying on independent Trusted Third Parties will alleviate the impact of malicious activity on the side of the provider by issuing the appropriate alarms in the event of any NSF environment manipulation.

- While it is true that any environment is vulnerable to malicious activity with full physical access (and this is obviously beyond the scope of this document), the application of attestation mechanisms raises the degree of physical control necessary to perform an untraceable malicious modification of the environment.

The client can have a proof that their NSFs and policies are correctly (from the client point of view) enforced by the Security Controller. Taking into account the threats identified in [I-D.ietf-i2nsf-framework], 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. 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. Establishing Client Trust

From a high-level standpoint, in any I2NSF platform, the client connects and authenticates to the Security Controller, which then initialises the client's NSFs and policies. Afterwards, user traffic from the client domain goes through the NSF platform which hosts the corresponding NSFs. The user's expectations of the platform behavior are thus twofold:

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

We will refer to the attestation of these two expectations as the "client-agnostic attestation" and the "client-specific attestation". Trusted Computing techniques play a key role in addressing this expectations.

3.1. First Step: Client-Agnostic Attestation

This is the first interaction between a client and a Security Controller: the client wants an attestation that proves it is connected to a genuine Security Controller before continuing 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 client credentials and set up the client NSFs and policies properly.

Once these two properties are proven to the client, the client knows that their credentials will only be used by the Security Controller to set up the execution of their NSFs.

3.2. Second Step: Client-Specific Attestation

From the security enforcement point of view, the client agnostic attestation focuses on the initialization of the execution platform

for the vNSFs. This second step aims to prove to clients that their security is enforced accordingly with their choices (i.e. NSFs and policies). The attestation can be performed at the initialization of the NSFs, before any user traffic is processed by the NSFs, or during the execution of the NSFs.

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

Additionally to the NSF attestation at the moment of their instantiation, a continuous attestation of the NSF execution (based on the generation of periodic TPM integrity measurements) MAY be required by a client to ensure their security.

3.3. 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 transitive 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.

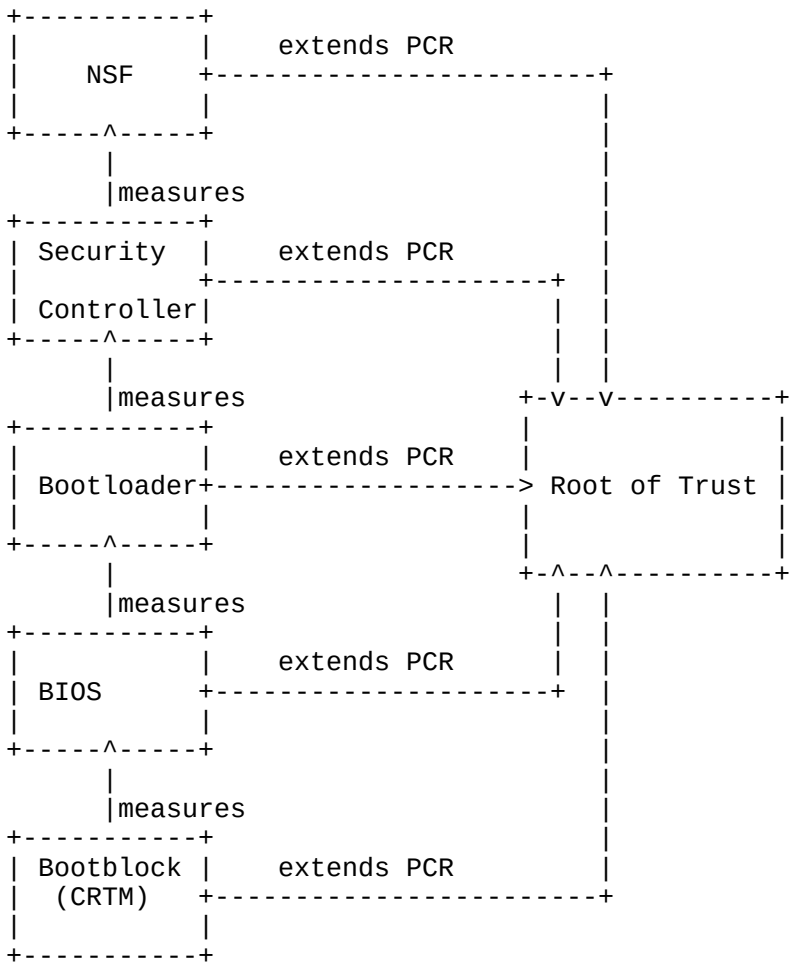


Figure 1: Applying Trusted Computing

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), which can be checked to prove the platform identity and attest the state of the system. The base element for the extension of the Chain of Trust is called the Core Root of Trust.

The TCG has created a standard for the 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). 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. 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.

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 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.

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. NSF Attestation Principles

Following the general requirements described in [I-D.ietf-i2nsf-framework] the Security Controller will become the essential element to implement the measurements described above, relaying on a TPM for the Root of Trust.

A mutual authentication of clients and the Security Controller MUST be performed, establishing the desired level of assurance. This level of assurance will determine how stringent are the requirements

for authentication (in both directions), and how detailed the collected measurements and their verification will be. Furthermore, the NSF platform MUST run a TPM, able to collect measurements of the platform itself, the Security Controller, and the NSFs being executed. The Security Controller MUST make the attestation measurements available to the client, directly or by means of a Trusted Third Party.

As described in [I-D.ietf-i2nsf-framework], a trusted connection between the client and the Security Controller MUST be established and all traffic to and from the NSF environment MUST flow through this connection

NOTE: The reference to results from WGs such as NEA and SACM is currently under consideration and will be included here.

4.1. Requirements for a Trusted NSF 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

NOTE: This section is derived from the original version of the document, focused on virtual NSFs. Although it seems to be applicable to any modern physical appliance, we must be sure all these considerations are 100% applicable to physical NSFs as well, and provide exceptions when that is not the case. Support from expert in physical node attestation is required here.

All clients 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 NSF 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 general usage of PCRs by each software component SHOULD conform to open standards, in order to make verifying attestation reports interoperable, as it is the case of TCG Generic Server Specification [TCGGSS].

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 root of trust match the particular state.

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.

4.1.2. Remote Attestation Service

A service MUST be present for providing signed attestation report (e.g. the measurements) from the Root of Trust (RoT) to the client. In case of failure to communicate with the service, the client MUST assume the service cannot be trusted and seek an alternative Security Controller.

Since some forms of RoT require serialised access (i.e. due to slow access to hardware), latency of getting an attestation report could increase with simultaneous requests. Simultaneous requests could occur if multiple Trusted Third Parties (TTP) request for attestation reports at the same time. This MAY be improved through batching of requests, in a special manner. In a typical remote attestation protocol, the client sends a random number ("nonce") to the RoT in order to detect any replay attacks. Therefore, caching of an attestation report does not work, since there is the possibility that it may not be a fresh report. The solution is to batch the nonce for each requestor until the RoT is ready for creating the attestation report. The report will be signed by the embedded identity of the RoT to provide data integrity and authenticity, and the report will

include all the nonces of the requestors. Regardless of the number of the number of nonces included, the requestor verifying the attestation report MUST check to see if the requestor's nonce was included in order to detect replay attacks. In addition to the attestation report containing PCRs, an additional report known as an SML (Secure Measurement Log) can be returned to the requestor to provide more information on how to verify the report (e.g. how to reproduce the PCR values). The integrity of the SML is protected by a PCR measurement in the RoT. An example of an open standard for responses is [TCGIRSS]. Further details are discussed in Section 5.2.

As part of initial contact, the Security Controller MAY present a list of external TTPs that the client can use to verify it. However, the client MUST assess whether these external verifiers can be trusted. The client can also choose to ignore or discard the presented verifiers.

Finally, to prevent malicious relaying of attestation reports from a different host, the authentication material of the secure channel (e.g. TLS, IPSec, etc.) SHOULD be bound to the RoT and verified by the connected client, unless the lowest levels of assurance have been chosen and an explicit warning issued. This is also addressed in Section 5.1.

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 NSF 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 NSF platform consists of three main phases, which need to be coordinated by the client:

1. Trusted channel with the Security Controller. During this phase, the client securely connects to the Security Controller to avoid that any 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 client verifies that the Security Controller components responsible for handling the credentials and for the isolation with respect to other potential clients 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 connection is terminated, the Security Controller verifies the integrity of the elements composing the NSF platform. The components responsible for this task have been already attested during the previous phase.

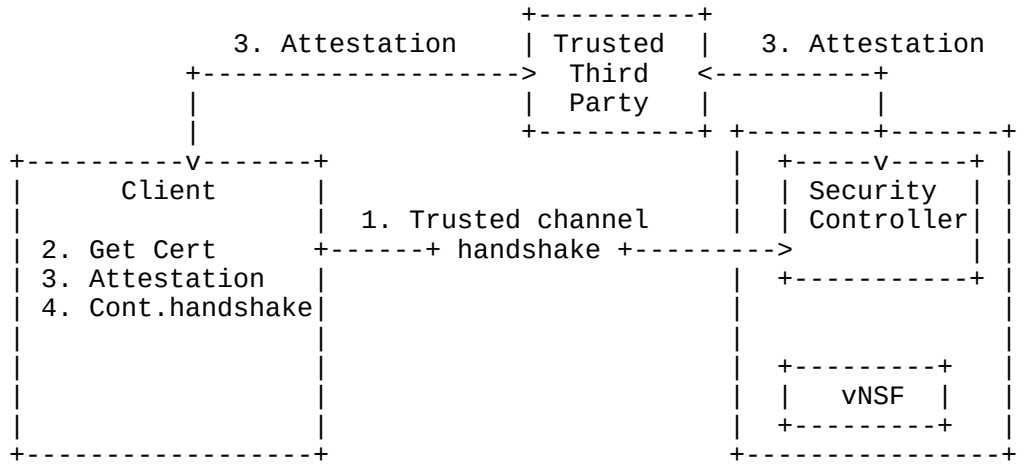


Figure 2: Steps for remote attestation

In the following each step, as depicted in the above figure, is discussed in more detail.

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 is redundant. This allows for the use of self-signed certificates, what would simplify operational procedures in many 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 client 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 client performs the remote attestation protocol with the Security Controller, either directly or with the help of a Trusted Third Party. The Trusted Third Party MAY perform the verification of attestation quotes on behalf of multiple clients.

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 client 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 of the vNSF platform, in order to build the chain of trust.

Since a client may not have enough capabilities to perform the integrity verification of a Security Controller the client 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 client 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 [TCGIRSS].

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 client 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 client.

4. The client 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 clients connected to it is enforced. As part of the verification, the client also checks that the digest of the certificate, received during the trusted channel handshake, is present among measurements.

If the client has limited computation resources, or requires an independent external element whom he can trust the measurements from, it may contact a TTP it may contact a TTP which, in turn, attests the Security Controller and returns the result of the integrity evaluation to the client, 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 operational environment for NSFs to be managed by the connecting client (the NSF platform, or just platform) in a way that any user traffic is processed only by these NSFs within the platform. Platform attestation, instead, evaluates the integrity of the NSFs running within the platform.

Platform attestation does not imply a validation of the mechanisms the Security Controller can apply to select the appropriate NSFs to enforce the Service Policies applicable to specific flows. The selection of these NSFs is supposed to happen independently of the attestation procedures, and trust on the selection process and the translation of policies into function capabilities has to be based on the trust clients have on the Security Controller being attested as the one it was intended to be used. An attestation of the selection and policy mapping procedures constitute an interesting research matter, but it is out of the scope of this document.

The procedures are essentially similar to the ones described in the previous section. This step MAY be applied periodically if the level of assurance selected by the user requires it.

Attesting NSFs, especially if they are 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.ietf-i2nsf-framework]
elopez@fortinet.com, e., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-02 (work in progress), July 2016.

[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, <http://www.rfc-editor.org/info/rfc2119>.

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

[TCGGSS] "TCG Generic Server Specification, Version 1.0", <http://www.trustedcomputinggroup.org/>.

[TCGIRSS] "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] Schiffman, J., Vijayakumar, H., and T. Jaeger, "Verifying System Integrity by Proxy", <http://dl.acm.org/citation.cfm?id=2368379>.

[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

Adrian L. Shaw
Hewlett Packard Labs
Long Down Avenue
Bristol, BS34 8QZ
UK

Phone: +44 117 316 2877
Email: als@hpe.com

I2NSF
Internet Draft
Intended status: Standard Track

L. Xia
Huawei
D. Zhang
Alibaba
E. Lopez
Fortinet
N. BOUTHORS
Qosmos

Expires: April 2016 October 19, 2015

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

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

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:

- o 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;

- o 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

AD - Active Directory

ANSI - American National Standards Institute

DDoS - Distributed Deny of Services

FW - Firewall

I2NSF - Interface to Network Security Functions

INCITS - International Committee for Information Technology Standards

IoT - Internet of Things

IPS - Intrusion Prevention System

LDAP - Lightweight Directory Access Protocol

NAT - Network Address Translation

NBI - North-bound Interface

NIST - National Institute of Standard Technology

NSF - Network Security Function

RBAC - Role Based Access Control

UE - User Equipment

URL - Uniform/Universal Resource Locator

VM - Virtual Machine

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:

- o 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;

- o 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;

- o 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;

- o 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:

o 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

o 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:

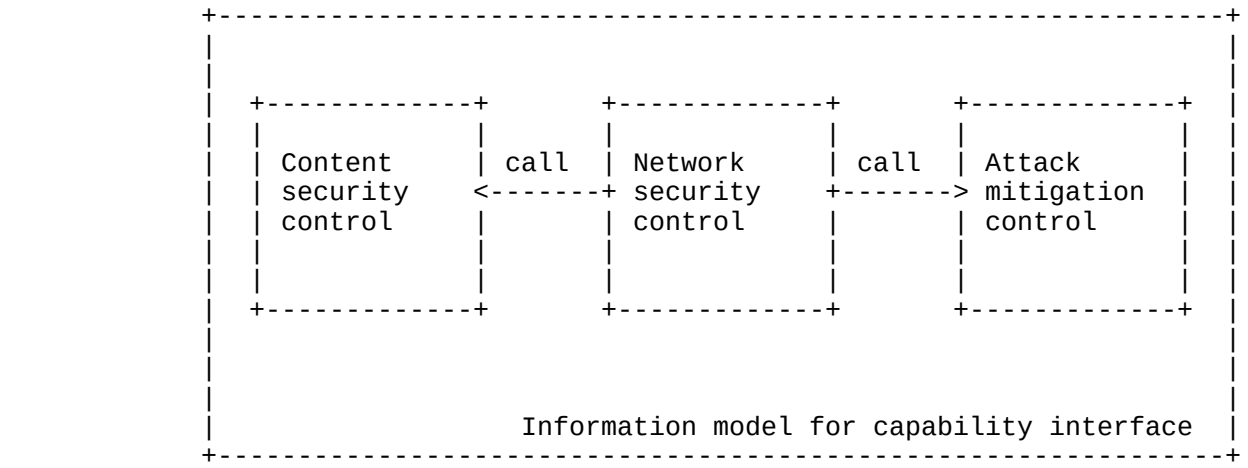


Figure 1. The overall structure of information model for I2NSF Capability Interface

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:

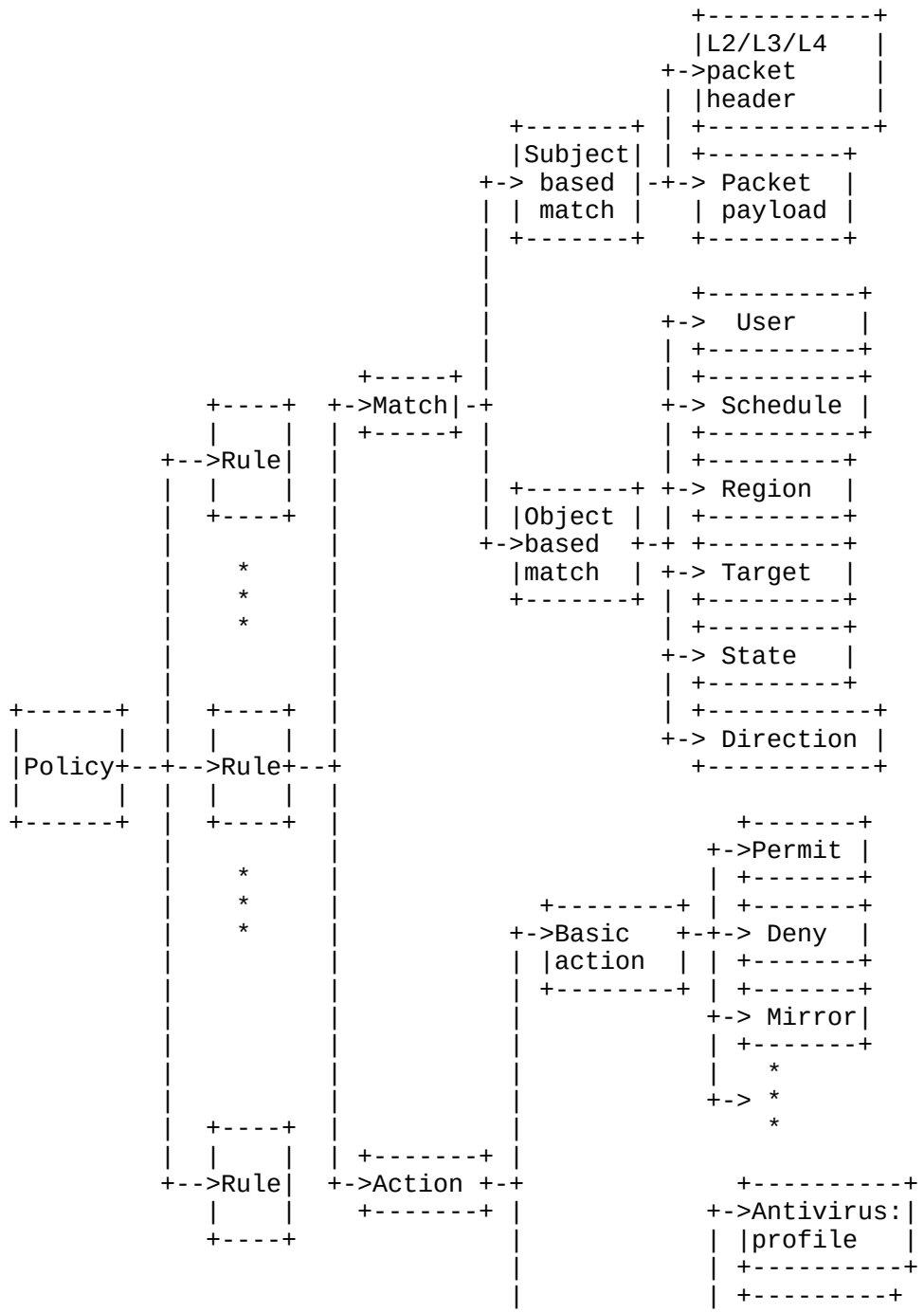

L2/L3/L4
packet
header
Subject
based
match
Packet
payload
User
Schedule
Region
Object
based
match
Target
State
Direction
Match
Rule
Policy
Rule
Permit
Basic
action
Deny
Mirror
Rule
Action
Antivirus:
profile

Figure 2. The basic structure of information model for network security control

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:

o 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;

o Packet payload: any segment of bytes in the packet payload;

- o 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;

- o 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;

- o 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.

- o 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 belongs 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:

Match Condition	Attributes: Values
Ethernet Frame Header	Source/Destination address s-VID/c-VID/EtherType
IPv4 Packet Header	src/dest address protocol src/dest port length flags ttl
IPv6 Packet Header	src/dest address protocol/nh src/dest port length traffic class hop limit flow label
TCP SCTP DCCP	Port syn ack fin rst psh urg window

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

Table 1. Match conditions details

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:

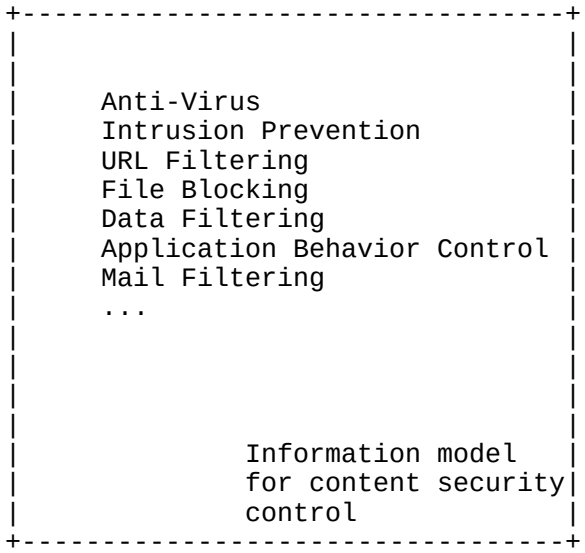


Figure 3. The basic structure of information model for content security control

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:

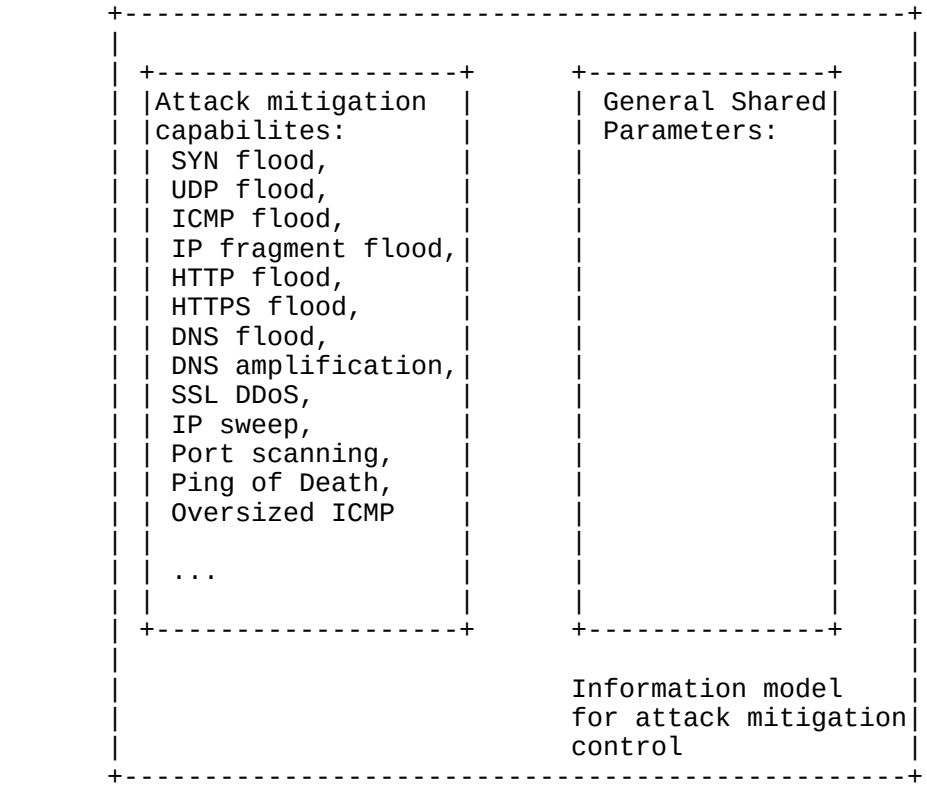


Figure 4. The basic structure of information model for attack mitigation control

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:

o IPv4: Match on destination IP address in the IPv4 header

o IPv6: Match on destination IP address in the IPv6 header

o MPLS: Match on a MPLS label at the top of the MPLS label stack

o MAC: Match on MAC destination addresses in the ethernet header

o 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_IPV6_ADDRESS>
                     <NEXT_HEADER> [<TRAFFIC_CLASS>]
                     [<FLOW_LABEL>] [<HOP_LIMIT>]

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

```
                <expired-date> <allow-multi-account-login>
                <address-binding>) | <tenant> | <VN-id>
   <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> |
                     <social-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

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

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, April 2009.

8.2. Informative References

[INCITS359 RBAC] NIST/INCITS, "American National Standard for Information Technology - Role Based Access Control", INCITS 359, April, 2003

[I-D.draft-pastor-i2nsf-merged-use-cases] Pastor, A., et.al., "Use Cases and Requirements for an Interface to Network Security Functions", Work in Progress, June, 2015.

[I-D.draft-merged-i2nsf-framework] Lopez, E., et.al., "Framework for Interface to Network Security Functions", Work in Progress, June, 2015.

[I-D.xia-i2nsf-service-interface-DM] Xia, L., et.al., "Data Model of Interface to Network Security Functions Service Interface", February, 2015.

[I-D.lopez-i2nsf-packet] Lopez, E., "Packet-Based Paradigm For Interfaces To NSFs", March, 2015.

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

I2nsf Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2016

J. You
Huawei
M. Zarny
Goldman Sachs
C. Jacquenet
M. Boucadair
France Telecom
Y. Li
J. Strassner
Huawei
S. Majee
F5 Networks

March 9, 2016

User-group-based Security Policy for Service Layer
draft-you-i2nsf-user-group-based-policy-01

Abstract

This draft defines the User-group Aware Policy Control (UAPC) framework, which facilitates consistent enforcement of security policies based on user group identity. Policies are used to control security policy enforcement using a policy server and a security controller. Northbound APIs are also discussed.

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/.

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 September 10, 2016.

Table of Contents

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). Common side effects are:

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

- o 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.

- o 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.

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

These are all symptoms of a system not using actual user identification information, but rather, one or more attributes that attempt to represent a user identity.

Traditional network access control mechanisms [I-D.ietf-i2nsf-problem-and-use-cases] do not work well in newer network paradigms.

- o 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 back-end Virtual Machine (VM)-based servers, can move; their IP addresses could change as a result. This means relying on well-known network fields (e.g., the 5-tuple) is increasingly inadequate to ensure consistent security policy enforcement.

- o 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 policies 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. This means the network needs to recognize the users' identity and their current context, and map the users to their correct access entitlement to the network.

- o Moreover, implementation of coherent security policy across several network and network security devices is almost impossible. NSFs 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 throughout the environment. 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, only to later find out that related ACLs, firewall policies, and other related mechanisms were not updated.

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 defines User-group Aware Policy Control (UAPC), which facilitates consistent enforcement of policies based on user-group identity, and discusses how it operates in the I2NSF Service Layer [I-D.merged-i2nsf-framework].

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, which allows each user in the user-group to be assigned the same access control permissions. 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 this user-group 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. Security policies should be consistently enforced based on their user-group identities, regardless of whether these terminal devices connect to a wired or a wireless infrastructure.

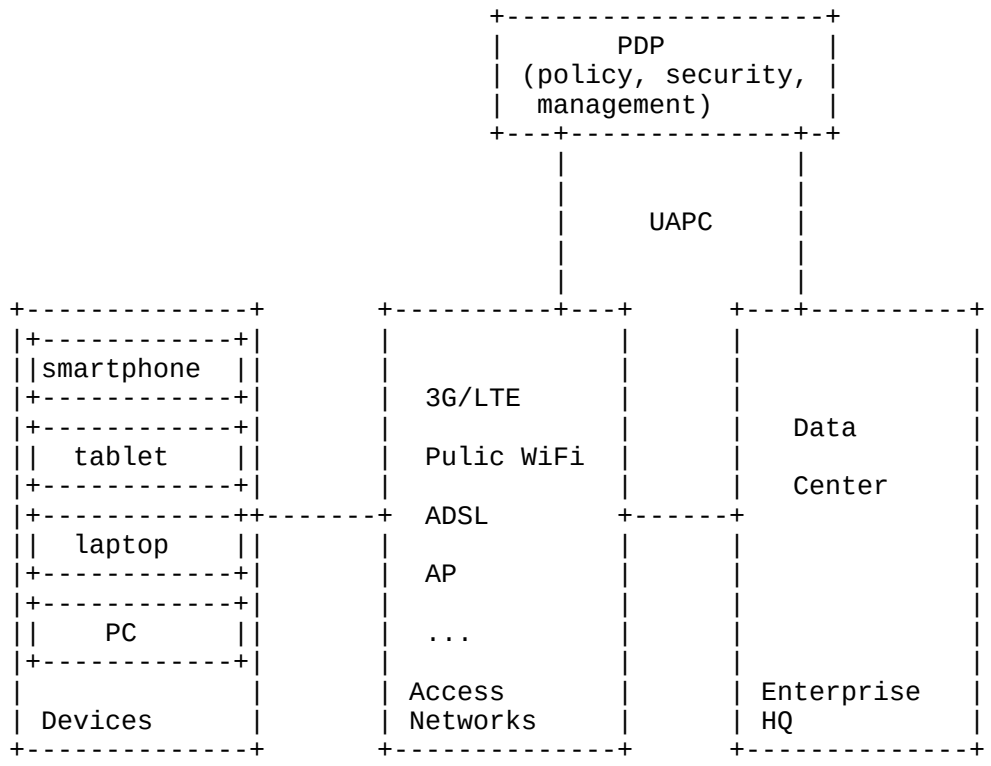


Figure 1: UAPC Framework Example

4. User-group Aware Policy Control

4.1. Overview

The UAPC framework is as follows enables users to be authenticated and classified into different user-groups at the network ingress by the Security Controller; this may require obtaining information from the Policy Server and an AAA server. The user-group is an identifier that represents the collective identity of a group of users, and is determined by a set of pre-defined policy criteria (e.g., source IP address, geo-location data, time of day, or device certificate). Users may be moved to different user-groups if their composite security context and/or environment change.

The Security Controller, if necessary, pushes the required user-group policies to all Network Security Functions (NSFs) that need them. The policies are expressed as 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. These and other 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, and whether traffic needs to be encrypted or not). 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 that packet.

4.2. Functional Entities

The UAPC framework consists of four main components: (1) Policy Server, (2) Authentication Server, (3) Security Controller, (4) Network Security Functions:

o Policy Server

The Policy Server houses two policy databases: (1) the user-group criteria, which assigns users to their user-group, and (2) the rule base of what each user group has access to.

- Contains (G)UI and/or APIs to enable policies to be created, modified, and deleted using command line, graphical tools, and/or programming logic

- Contains logic to create, read, update, and delete policies and policy components, and apply policies to user-groups from one or more policy repositories

- Contains logic to detect conflicts between policies

- Contains logic to resolve conflicts between policies

- Contains logic to broker and/or federate policies between domains

The above subjects are beyond the scope of this document.

o AAA Server

The AAA Server authenticates users, and then performs associated authorization and accounting functions. The AAA server classifies users into different user-groups at the network ingress. AAA server implementation details are out of scope for this document.

o Security Controller

The Security Controller coordinates various network security-related tasks on a set of NSFs under its administration. In general, there may be multiple security domains, where each domain has its own security controller. The detailed architecture is beyond the scope of this document.

- 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. The security controller may cache policies.

- 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 enforced

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

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

o 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, whose definition is controlled by one or more policy rules (e.g., source IP, geo-location, time of day, and device certificate).

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, terminals and other devices are also 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 address, 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.

Table 1 shows an example of how user-group definitions may be constructed. 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: User-Group Example

Group Name	Group ID	Group Definition
R&D	10	R&D employees
R&D BYOD	11	Personal devices of R&D employees
Sales	20	Sales employees
VIP	30	VIP employees
Workflow	40	IP addresses of Workflow resource servers
R&D Resource	50	IP addresses of R&D resource servers
Sales Resource	54	IP addresses of Sales resource servers

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 that are managed by sets of policies.

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 5-tuples. (Firewall rule-bases could and do include criteria other than the standard 5-tuple. Also, the user-group rule-base could consist of other criteria. Actual implementation details are out of scope of this document.)

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.

Table 2: Inter-group Policy Example

	Destination Group		
Source Group	Workflow Group	R&D Resource Group	Sales Resource Group
R&D group	Permit	Permit	Deny
R&D BYOD group	Permit	Deny	Deny
Sales group	Permit	Deny	Permit
VIP user group	Permit	Permit	Permit

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 that shows how users may access internal network resources. Similar models can be developed for "east-west" intra-data center traffic flows.

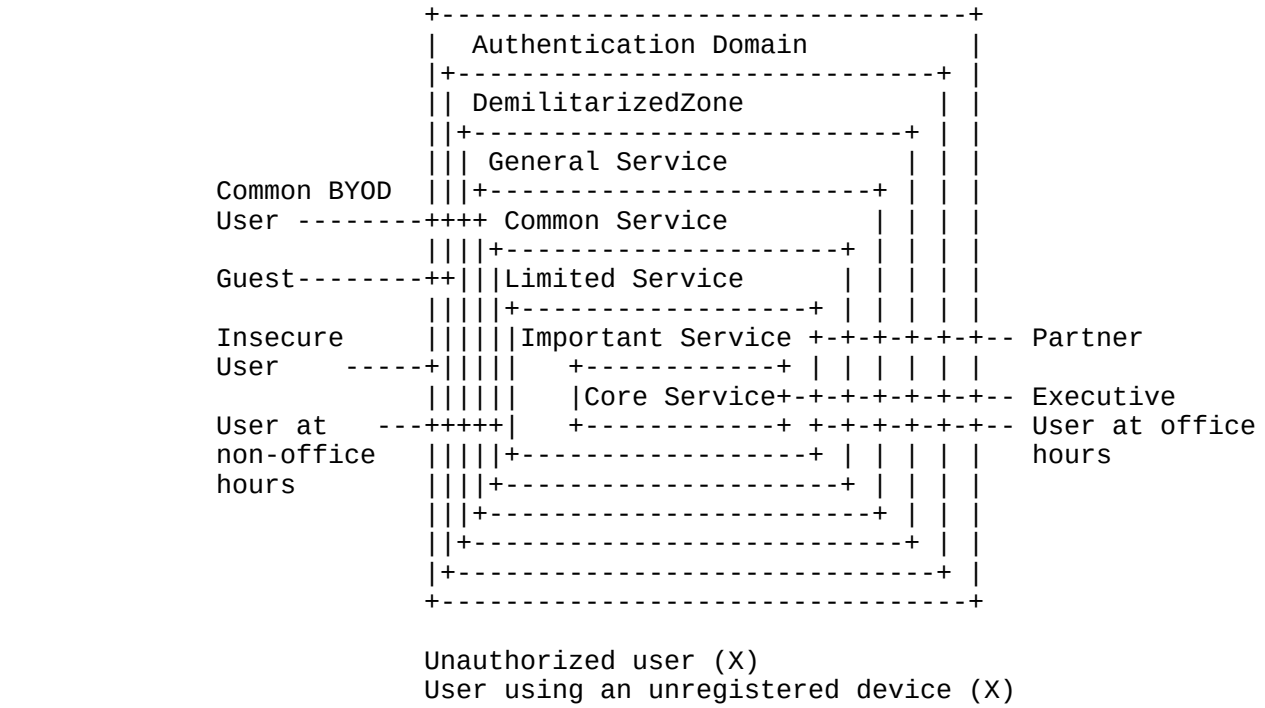


Figure 2: Sample Authorization Rules for User-group Aware Policy Control

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. There are five steps for implementing the UAPC framework, which are shown in Figure 3.

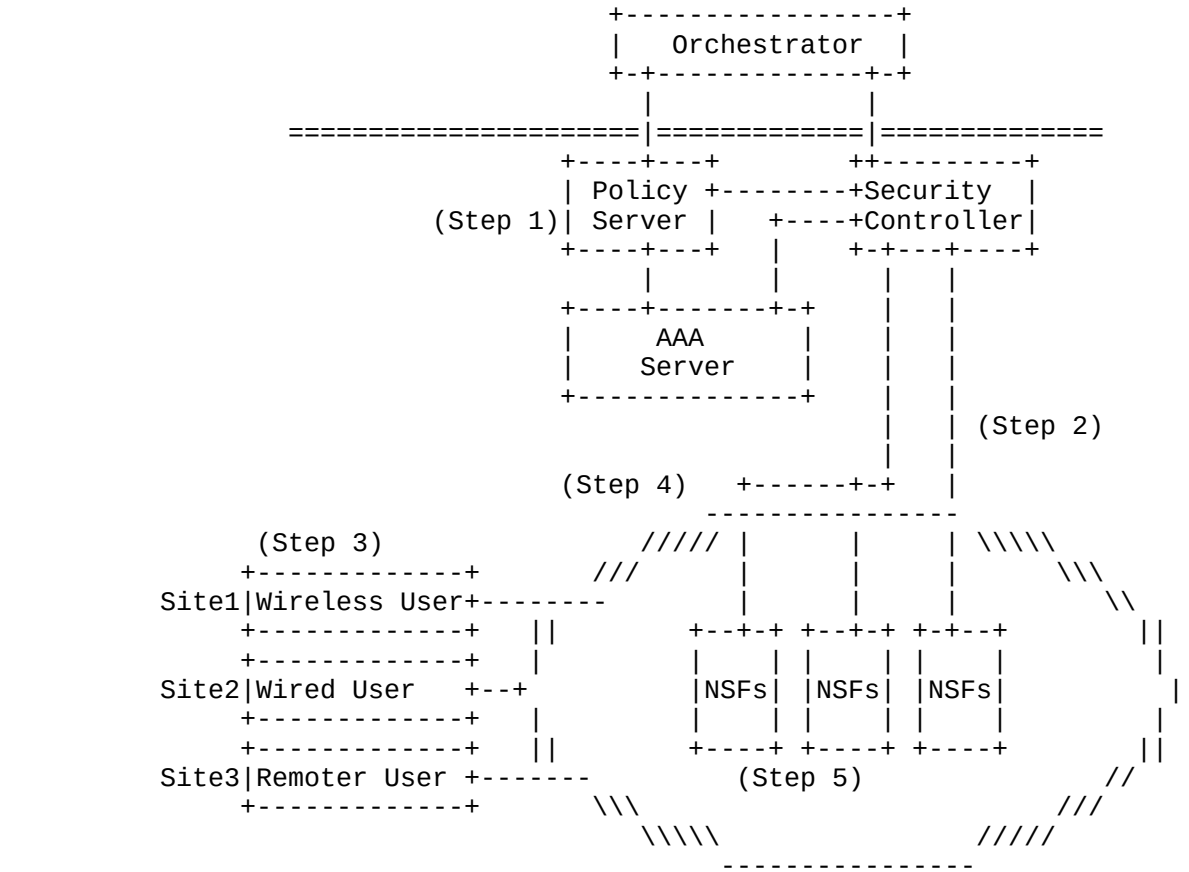


Figure 3: Unified Policy Procedures

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

2. The user-group-based policies are implemented on the NSFs under the Security Controller's management.

3. When a given user first comes logs onto 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. If the authentication is not successful, then the user is not assigned a user-group, which means that the user has no access permissions for the network.

5. The user's subsequent traffic is allowed or permitted based on the user-group ID by the NSFs 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 NSFs. See Section 4.1 for a brief overview of possible implementation methods.)

5. Requirements for I2NSF

Key aspects of the UAPC framework fall 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. Security Considerations

TBD

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

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

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014, <http://www.rfc-editor.org/info/rfc7297>.

9.2. Informative References

[I-D.ietf-i2nsf-problem-and-use-cases] Hares, S., Dunbar, L., Lopez, D., Zarny, M., and C. Jacquenet, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-00 (work in progress), February 2016.

[I-D.merged-i2nsf-framework] Ed, E., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-merged-i2nsf-framework-04 (work in progress), October 2015.

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

John Strassner
Huawei
2330 Central Expressway
San Jose, CA
USA

Email: john.sc.strassner@huawei.com

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

Email: S.Majee@f5.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 18, 2016

D. Zhang
Y. Wu
Aliababa Group
L. Xia
Huawei
March 17, 2016

An Information Model for the Monitoring of Network Security Functions (NSF)
draft-zhang-i2nsf-info-model-monitoring-00

Abstract

In this draft, an information model for the capability interface monitoring network security functions (NSF) is proposed.

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].

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 September 14, 2016.

Copyright Notice

Table of Contents

1. Introduction

According to [I-D.merged-i2nsf-framework], the interface provided by a NSF (e.g., FW, AAA, IPS, Anti- DDOS, or Anti-Virus) for other network entities (e.g., I2NSF client, or network/security controller) to control and monitor the NSF is referred to as a 'capability interface'. In practice, the NSF can communicate with the network entities though the capability interface in either a 'push' or a 'pull' way. The NSF can send out a report about its status or about certain network event proactively or send out message as a reply to network entities who control or monitor it. This memo will not go into the design details of capability interface. Instead, this draft is engaged in specifying the information that a NSF needs to provided in the monitoring part of the capability interface.

In this memo, the following types of reports that a NSF may need to provide are considerred:

- The alarm triggered by a certain status in NSF
- The alarm triggered by a certain security event
- The report about the security events occured in a certain period
- The report about the status of NSF in a certain period

2. Common Information

There are some information that should be provided in each message sent from a NSF to the network entity monitoring it.

- Time Stampe: indicate the time when the message is generated.
- NSF name: the name (or IP) of the device generatign the message.
- Vendor name: the name of the NSF vendor
- Type of NSF: inidcate what type the NSF is (e.g., firewall, WAF, IPS)
- NSF model
- Vesion: The version of the interface
- NSF Version: The software version of the NSF
- Type of report: Alarm, periodical report, etc.

3. Alarm

An alarm is the message generated by a NSF. An alarm could be triggered by certain abnormal conditions occurred in a NSF (referred to as a System Alarm) or a detected network abnormal conditions (referred to as a Security Event Alarm).

3.1. System Alarm

3.1.1. Memory Alarm

The following information should be included in a Memory Alarm:

- event _Name: MEM_USAGE_HIGH
- usage:the usage rate of memory
- threshold: the threshold triggering the event
- message: The memory usage exceeded the threshold

3.1.2. CPU Alarm

The following information should be included in a CPU Alarm:

- event _Name: 'MEM_USAGE_HIGH'
- usage:the usage rate of CPU
- threshold: the threshold triggering the event
- message: 'The CPU usage exceeded the threshold'

3.1.3. DISK Alarm

The following information should be included in a Disk Alarm:

- event _Name: 'MEM_USAGE_HIGH'
- usage:the usage rate of disk
- threshold: the threshold triggering the event
- message: 'The disk usage exceeded the threshold'

3.1.4. Session Table Alarm

The following information should be included in a Session Table Alarm:

- event _Name: 'SESSION_USAGE_HIGH'
- current: the number of concurrent sessions
- Max:the maximum number of sessions that the session table can support
- threshold: the threshold triggering the event
- message: 'The number of session table exceeded the threshold'

3.1.5. Interface Alarm

The following information should be included in a Interface Alarm:

- event _Name: 'IFNET_State'
- interface_Name:the name of interface
- state: 'UP' or'DOWN'
- message: 'The disk usage exceeded the threshold'

3.2. Security Event Alarm

3.2.1. DDoS Alarm

The following information should be included in a DDoS Alarm:

- event_Name: 'SEC_EVENT_DDoS'
- sub_attack_type: any one of Syn flood, ACK flood, SYN-ACK flood, FIN/RST flood, TCP Connection flood, UDP flood, Icmp flood, HTTPS flood, HTTP flood, DNS query flood, DNS reply flood, SIP flood, and etc.
- dst_ip: the IP address of a victum under attack
- dst_port: the port numbers that the attrack traffic aims at.
- start_time: The time stampe indicating when the attack started

- end_time:The time stampe indicating when the attack ended. If the attack is still undergoing when sending out the alarm, this field can be empty.
- attack_rate: the PPS of attack traffic
- attack_speed: the bps of attack traffic

3.2.2. Virus Alarm

The following information should be included in a Virus Alarm:

- event_Name: 'SEC_EVENT_Virus'
- virus_type: type of the virus, e.g., trojan，worm，macro Virus type
- virus_name
- dst_ip: The destination IP address of the packet where the virus is found
- src_ip: The source IP address of the packet where the virus is found
- src_port: The source port of the packet where the virus is found
- dst_port: The destination port of the packet where the virus is found
- src_zone: The source security zone of the packet where the virus is found
- dst_zone: The destination security zone of the packet where the virus is found
- file_type: The type of the file where the virus is hided within
- file_name: The name of the file where the virus is hided within
- virus_info: The brief introduction of virus
- raw_info: The information describing the packet triggering the event.
-
-

3.2.3. Intrusion Alarm

The following information should be included in a Intrustion Alarm:

- event_name: the name of event：SEC_EVENT_Intrusion
- sub_attack_type: attack type，e.g., brutal force、buffer overflow
- src_ip: The source IP address of the packet
- dst_ip: The destination IP address of the packet
- src_port:The source port number of the packet
- dst_port; The destination port number of the packet
- src_zone :the source security zone of the packet
- dst_zone: the destination security zone of the packet
- protocol: The employed transport layer protocol，e.g.,TCP、UDP
- app: The employed application layer protocol，e.g.,HTTP、FTP
- rule_id: The ID of the rule being triggered
- rule_name: The name of the rule being triggered
- intrusion_info: Simple description of intrusion
- raw_info: The information describing the packet triggering the event.

3.2.4. Botnet Alarm

The following information should be included in a Botnet Alarm:

- event_name: the name of event：SEC_EVENT_Botnet
- botnet_name: The name of the detected botnet
- src_ip: The source IP address of the packet
- dst_ip: The destination IP address of the packet

- src_port: The source port number of the packet
- dst_port: The destination port number of the packet
- src_zone: the source security zone of the packet
- dst_zone: the destination security zone of the packet
- protocol: The employed transport layer protocol，e.g.,TCP、UDP
- app: The employed application layer protocol，e.g.,HTTP、FTP
- role: The role of the communicating parties within the botnet:
  1. the packet from zombie host to the attacker
  2. The packet from the attacker to the zombie host
  3. The packet from the IRC/WEB server to the zombie host
  4. The packet from the zombie host to the IRC/WEB server
  5. The packet from the attacker to the IRC/WEB server
  6. The packet from the IRC/WEB server to the attacker
  7. The packet from the zombie host to the victim
- botnet_info: Simple description of Botnet
- raw_info: The information describing the packet triggering the event.

3.2.5. Web Attack Alarm

The following information should be included in a Web Attack Alarm:

- event_name: the name of event：SEC_EVENT_WebAttack
- sub_attack_type: Concret web attack type，e.g.，sql injection 、command injection 、XSS、CSRF
- src_ip: The source IP address of the packet
- dst_ip: The destination IP address of the packet

- src_port: The source port number of the packet
- dst_port: The destination port number of the packet
- src_zone: the source security zone of the packet
- dst_zone: the destination security zone of the packet
- req_method: the method of requirement. For instance, 'PUT' or 'GET' in HTTP.
- req_url: Requested URL

4. Reports

Different from Alarms, a report normally triggered by a timmer or a request from the NE monitoring the device. So compared to alarms, a report contains more statical information.

4.1. Attack Report

4.1.1. DDoS Report

Besides the fields in an DDoS Alarm, the following information should be included in a DDoS Report:

- attack_type: attack type： DDoS
- attack_ave_rate: The average pps of the attack traffic within the recorded time
- attack_ave_speed: The average bps of the attack traffic within the recorded time
- attack_pkt_ num: The number attack packets within the recorded time
- rule_id: The ID of the rule being triggered
- rule_name: The name of the rule being triggered
- attack_ src_ip: The source IP addresses of attack traffics. If there are a large amount of IP addresses, then pick a certain number of resources according to different rules.

4.1.2. Virus Report

Besides the fields in an Virus Alarm, the following information should be included in a Virus Report:

- attack_type: attack type： Virus

  *

- protocol The transport layer protocol

- app The name of the application layer protocol

- times The time of detecting the virus

- action The actions dealing with the virus, e.g., alert，block os The OS that the virus will affect, e.g., all，android，ios，unix，windows

4.1.3. Intrusion Report

Besides the fields in an Intrusion Alarm, the following information should be included in a Intrusion Report:

- attack_type: attack type： Intrusion

- times: The times of intrusions happened in the recorded time action STRING The actions dealing with the intrusions, e.g., alert，block

- os: The OS that the intrusion will affect, e.g., all, android, ios, unix, windows

- action: The actions dealing with the intrusions, e.g., alert, block

- attack_rate: NUM the pps of attack traffic

- attack_speed: NUM the bps of attack traffic

4.1.4. Botnet Report

Besides the fields in an Botnet Alarm, the following information should be included in a Botnet Report:

- attack_type: attack type： Botnet

- botnet_pkt_num:The number of the packets sent to or from the detected botnet
- action: The actions dealing with the detected packets, e.g., alert，block
- os: The OS that the attack aiming at, e.g., all，android，ios，unix，windows等。

4.1.5. Web Attack Report

Besides the fields in an Web Attack Alarm, the following information should be included in a Web Attack Report:

- attack_type: attack type： Web Attack
- rsp_code: response code
- req_clientapp: the client application
- req_cookies: Cookies
- req_host: The domain name of the requested host
- raw_info: The information describing the packet triggering the event.

4.2. Service Report

4.2.1. Traffic Report

TBD

4.2.2. Policy HIT Report

TBD

4.2.3. DPI Report

4.3. System Report

4.4. Operation Report

4.5. Running Report

5. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

6. Security Considerations

TBD

7. Acknowledgements

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

8.2. Informative References

[I-D.merged-i2nsf-framework] Ed, E., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-merged-i2nsf-framework-04 (work in progress), October 2015.

Authors' Addresses

Dacheng Zhang
Aliababa Group

Email: dacheng.zdc@alibaba-inc.com

Yi Wu
Aliababa Group

Email: anren.wy@alibaba-inc.com

Liang Xia
Huawei

Email: frank.xialiang@huawei.com