     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.

Copyright Notice

to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction

   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:

   o 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.

   o 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.

   o 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.

   o 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:

o 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).

o 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.

```
        +----------+
        |          |      extends PCR
        |   NSF    +-----------------------+
        |          |                       |
        +-----^----+                       |
              |                            |
              |measures                    |
        +----------+                       |
        | Security |     extends PCR        |
        |          +-------------------+    |
        | Controller|                  |   |
        +-----^----+                   |   |
              |                        |   |
              |measures          +-v--v----------+
        +----------+             |               |
        |          |      extends PCR    |        |
        | Bootloader+-------------------> Root of Trust |
        |          |             |        |
        +-----^----+             |        |
              |              +-^--^----------+
              |measures          |   |
        +----------+             |   |
        |          |     extends PCR    |   |
        | BIOS     +-------------------+   |
        |          |                       |
        +-----^----+                       |
              |                            |
              |measures                    |
        +----------+                       |
        | Bootblock |    extends PCR        |
        |   (CRTM)  +-----------------------+
        |          |
        +----------+
```
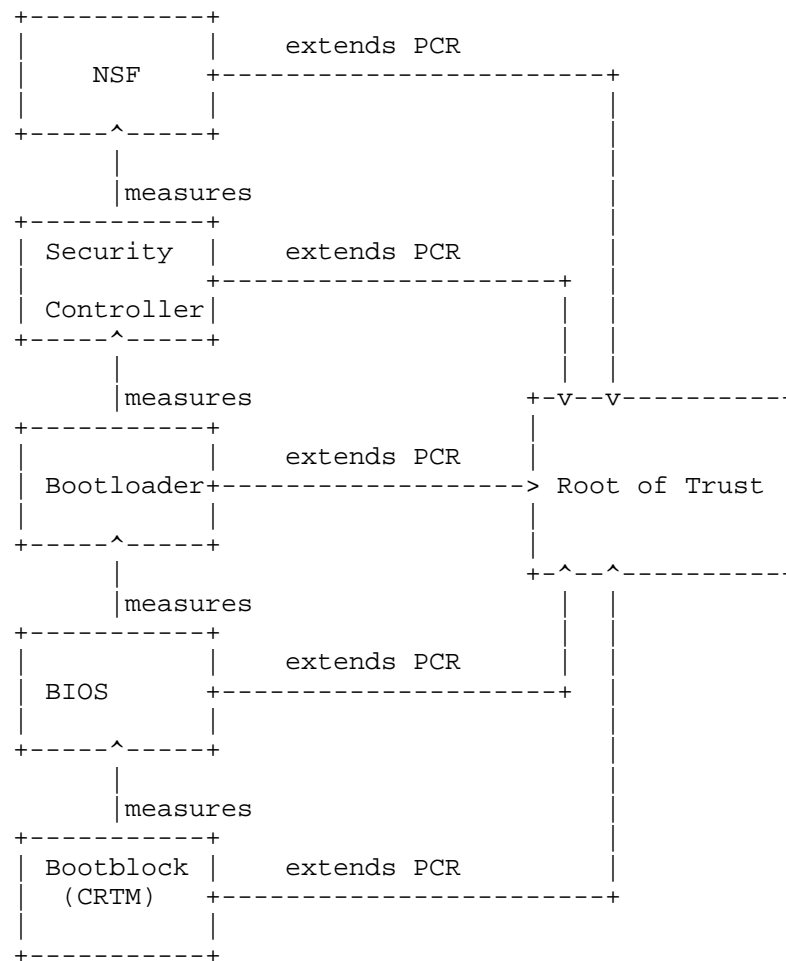
Figure 1: Applying Trusted Computing

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
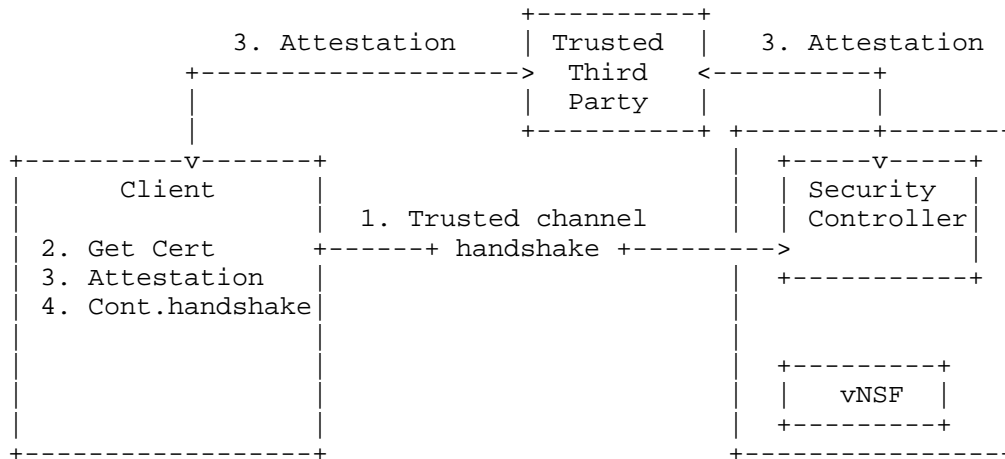   already attested during the previous phase.

```
                                      +----------+
                 3. Attestation       | Trusted  |   3. Attestation
              +-------------------->  Third   <----------+
              |                       | Party    |           |
              |                       +----------+ +--------+-------+
 +---------v-------+                        |    +-----v-----+ |
 |    Client       |                        |    | Security  | |
 |                 |   1. Trusted channel   |    | Controller| |
 | 2. Get Cert     +------+ handshake +--------->          | |
 | 3. Attestation  |                        |    +-----------+ |
 | 4. Cont.handshake|                       |                 |
 |                 |                        |                 |
 |                 |                        |    +---------+  |
 |                 |                        |    |  vNSF   |  |
 |                 |                        |    +---------+  |
 +-----------------+                        +---------------+
```

                   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