

**Integrity Measurement for Network File System version 4
draft-ietf-nfsv4-integrity-measurement-05**

Abstract

This document specifies an OPTIONAL extension to NFS version 4 minor version 2 that enables Linux Integrity Measurement Architecture metadata (IMA) to be conveyed between NFS version 4.2 servers and clients. Integrity measurement authenticates the creator of a file's content and helps guarantee the content's integrity end-to-end from creation to use.

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 <https://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 December 8, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4](#).e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
1.1.	The Linux Integrity Measurement Architecture	3
1.1.1.	IMA Metadata	4
1.1.2.	Creating and Verifying IMA Metadata	4
1.1.3.	Distributing and Protecting Keying Material	5
1.1.4.	Using IMA to Protect NFS Files	5
2.	Requirements Language	5
3.	Protocol Extension Considerations	5
3.1.	XDR Extraction	5
4.	Managing IMA Metadata on NFS Files	6
4.1.	XDR Definition	6
4.1.1.	NFS4ERR_INTEGRITY (Error Code YYYY)	7
4.2.	Detecting support for IMA Metadata	7
4.2.1.	Reporting Server-Side IMA Appraisal Failures	7
4.3.	Storing IMA Metadata	8
4.3.1.	Sending IMA Metadata When Creating a New Object	9
4.3.2.	Authorizing Updates to IMA Metadata	9
4.4.	Retrieving IMA Metadata	10
4.5.	Using NFS Attribute Fencing (VERIFY/NVERIFY)	10
5.	Deployment Examples	11
5.1.	Terminology	11
5.2.	Instantiating IMA Metadata	12
5.3.	Interaction With Non-Participating Implementations	12
6.	Implementation Status	13
6.1.	Linux NFS server and client	14
7.	Security Considerations	14
8.	IANA Considerations	15
9.	References	15
9.1.	Normative References	15
9.2.	Informative References	16
9.3.	URIs	16
	Acknowledgments	16
	Author's Address	17

[1.](#) Introduction

The security of software distribution systems is complex and challenging, especially as software distribution has become increasingly decentralized. An end administrator needs to trust that she is running executables just as they are supplied by a software vendor; in other words, that they have not been modified by malicious actors, contracted system administration services, or broken hardware or software. Software vendors want a guarantee that customer-

Lever

Expires December 8, 2019

[Page 2]

installed executables that fall under support contracts have similarly not been modified.

There already exist mechanisms that protect file data during certain portions of a file's life cycle:

- o Whole file system checksumming can verify so-called Golden Master installation media before it is used to install the software it contains.
- o File or block integrity mechanisms can protect data at rest on storage servers.
- o For a distributed file system such as NFS, transport layer security or a GSS integrity service (as described in [\[RFC7861\]](#)) can protect data while it traverses a network between a storage server and a client.

A more extensive mechanism is needed to guarantee that no modification of a particular file has occurred since it was created, perhaps even after several generations of copies have been made of the file's content.

[1.1.1.](#) The Linux Integrity Measurement Architecture

The Linux Integrity Measurement Architecture (IMA) [\[SAILER\]](#) provides assurance that the content of a file is unaltered and authentic to what was originally written to that file. The goal is to detect when an attacker, unintentional platform behavior, or local tinkering has modified the content of a file, either in transit or at rest.

This is done by separately storing metadata about a file's content and then using that metadata to verify the content before it is used. Verification of the content is entirely independent of the file system. File systems, both local and remote, act simply as storage for both the content and the metadata, both of which are opaque to the storage subsystem.

An informative description of this mechanism is presented in the following subsections to provide context for understanding the NFS protocol extension described later in this document. As the file system does not interpret IMA metadata, this description is not necessary to implement the extension.

Lever

Expires December 8, 2019

[Page 3]

1.1.1. IMA Metadata

A cryptographically signed hash stored separately from a file's content serves as a strong check of file content integrity and authenticates the identity of the provider of the file's content. The signer is verified at time of content use via a web of trust commonly provided by PKI or x.509 certificates [[RFC4158](#)].

The hash is typically computed using either the SHA-1 or SHA-256 algorithm and is stored as an HMAC [[RFC2104](#)]. For the purposes of this document, the current document refers to this blob as "IMA metadata".

The precise format of this metadata is determined by policies set by the local security administrator; the metadata and its format are opaque to the mechanisms that store or transport it (i.e., file systems). The particulars of the PKI and the hash algorithm are agreed upon out-of-band and recognized by all participating IMA subsystems.

1.1.2. Creating and Verifying IMA Metadata

In a typical deployment, an authority (such as a software vendor) computes the hash of a file after its content has been finalized. The hash is then signed and attached to the file. A web of trust typically links the signer to the users of the file's content (such as customers of the software vendor).

Directly before file content is to be used, a security module locally re-computes the hash of the file content and stores it in a cache. This step is known as "measurement".

The next step is referred to as "appraisal". The security module then reads the associated IMA metadata and validates its signature. If the signature is invalid or the locally computed hash does not match the stored hash, the security module applies an appraisal policy. The file may be flagged in an audit log or access to the file may be denied.

Underlying file and storage systems play no part in measurement or appraisal. They act only as a conduit by which file content and IMA metadata move between at-rest storage and the security module on the host where that content is to be used. Both IMA metadata and file content are opaque to storage subsystems.

Lever

Expires December 8, 2019

[Page 4]

1.1.3. Distributing and Protecting Keying Material

A Trusted Platform Module [[1](#)] can seal key material used to sign and appraise file content. Unprotected keys are not stored in or distributed via file systems. Distributing and protecting such key material is outside the scope of the extension specified in this document.

1.1.4. Using IMA to Protect NFS Files

The protocol extension in this document enables the storage and use of IMA metadata so that measurement and appraisal can occur at point-of-use on NFS client and server hosts. This mechanism is similar to NFSv4 Security Labels (specified in [[RFC7862](#)] et al). The purpose of the mechanism defined in the current document is to store security-related file metadata that is not interpreted by the file system itself.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14](#) [[RFC2119](#)] [[RFC8174](#)] when, and only when, they appear in all capitals, as shown here.

3. Protocol Extension Considerations

This document specifies an OPTIONAL extension to NFS version 4 minor version 2 [[RFC7862](#)], hereafter referred to as NFS version 4.2. NFS version 4.2 servers and clients implemented without knowledge of this extension will continue to interoperate with NFS version 4.2 clients and servers that are aware of the extension, whether or not they support it.

Because [[RFC7862](#)] does not define NFS version 4.2 as non-extensible, [[RFC8178](#)] treats it as an extensible minor version. Therefore this Standards Track RFC extends NFS version 4.2 but does not update [[RFC7862](#)] or [[RFC7863](#)].

3.1. XDR Extraction

[Section 4.1](#) contains a description of an extension to the NFS version 4.2 protocol, expressed in the External Data Representation (XDR) language [[RFC4506](#)]. This description is provided in a way that makes it simple to extract into ready-to-compile form. The reader can apply the following sed script to this document to produce a machine-readable XDR description of the extension.

Lever

Expires December 8, 2019

[Page 5]

<CODE BEGINS>

```
sed -n -e 's:^ */// ::p' -e 's:^ *///$::p'
```

<CODE ENDS>

That is, if this document is in a file called "ima-extension.txt" then the reader can do the following to extract an XDR description file:

<CODE BEGINS>

```
sed -n -e 's:^ */// ::p' -e 's:^ *///$::p'
    < ima-extension.txt > ima.x
```

<CODE ENDS>

Once that extraction is done, these added lines need to be inserted into an appropriate base XDR of the generated XDR from [\[RFC7863\]](#) together with XDR from any additional extensions to be recognized by the implementation. This will result in a ready-to-compile XDR file.

[4.](#) Managing IMA Metadata on NFS Files

[4.1.](#) XDR Definition

This section defines a new data type to encapsulate and a new OPTIONAL attribute to access and update IMA metadata associated with a particular file.

To enable a single IMA metadata payload to be retrieved or updated via a single RPC, and to constrain the transport resources required for the operations defined in this section, the length of IMA metadata MUST NOT exceed 4096 bytes in length.

When an NFS version 4.2 server does not recognize, or does recognize but does not support, this new attribute, the server responds in accordance with the requirements specified in [Section 4.3 of \[RFC8178\]](#).

Lever

Expires December 8, 2019

[Page 6]

<CODE BEGINS>

```
/// /*
///  * Copyright (c) 2019 IETF Trust and the person identified
///  * as author of the code. All rights reserved.
///  *
///  * The author of the code is: C. Lever
///  */
///
/// %/*
/// % * New For Integrity Measurement support
/// % */
/// opaque                ima_data4<4096>;
///
/// const FATTR4_IMA = XXX;      /* to be assigned */
///
/// %/*
/// % *New value added to enum nfsstat4
/// % */
/// const NFS4ERR_INTEGRITY = YYYYY; /* to be assigned */
```

<CODE ENDS>

4.1.1. NFS4ERR_INTEGRITY (Error Code YYYYY)

The server rejected this request because a data or metadata integrity check failed during its execution.

4.2. Detecting support for IMA Metadata

An NFS version 4.2 client discovers support for IMA metadata on an NFS version 4.2 server by sending an NFS GETATTR operation that specifies the FATTR4_SUPPORTED_ATTRS attribute and the FATTR4_IMA attribute. When a server supports IMA metadata, it sets the FATTR4_IMA attribute bit in the NFS GETATTR bitmask returned in the reply. Otherwise that bit is clear.

An NFS version 4.2 server MUST NOT return NFS4ERR_INTEGRITY to a client unless that client has queried the server for IMA metadata support using the above mechanism. The server identifies clients using their client_id4 for this purpose.

4.2.1. Reporting Server-Side IMA Appraisal Failures

An NFS server that has rigorous integrity checking must somehow report integrity-related failures to clients. Until now, a server implementer chose amongst status codes that were available in the base NFS version 4.2 protocol, typically NFS4ERR_IO or

Lever

Expires December 8, 2019

[Page 7]

NFS4ERR_ACCESS, even though these code points have generic meanings that do not necessarily imply an integrity-related failure.

Once the above FATTR4_SUPPORTED_ATTRS handshake is done, the server has determined that a client can properly recognize the NFS4ERR_INTEGRITY status code. In instances where an NFS request fails due to an integrity-related issue, and the server has determined that the client recognizes the NFS4ERR_INTEGRITY status code, the server MAY return NFS4ERR_INTEGRITY for the following operations: ACCESS, COMMIT, CREATE, GETATTR, GETDEVICELIST, LINK, LOOKUP, LOOKUPP, NVERIFY, OPEN, OPENATTR, READ, READDIR, READLINK, REMOVE, RENAME, SETATTR, VERIFY, WRITE. The server MUST NOT return NFS4ERR_INTEGRITY for any other operation.

The NFS4ERR_INTEGRITY status code is useful to inform the client (or the end user, depending on the client implementation) that access to the file's content was not blocked because of a permissions setting but rather because an integrity check failed. This distinction can guide the user or client towards a recovery action that is appropriate.

[cel: this is still an open issue]

4.3. Storing IMA Metadata

An NFS version 4.2 client stores IMA metadata by sending an NFS SETATTR operation that specifies the FATTR4_IMA attribute and targets the file system object associated with the metadata to be stored. This attribute completely replaces any previous FATTR4_IMA attribute associated with that object. Modifying an object in any other way MUST NOT alter or remove FATTR4_IMA attributes.

To remove IMA metadata from an object, the client sends a FATTR4_IMA attribute whose length is zero.

When an NFS SETATTR is presented to an NFS version 4.2 server with a credential that is not authorized to replace a FATTR4_IMA attribute, the server MUST respond with NFS4ERR_ACCESS.

When an NFS SETATTR is presented to an NFS version 4.2 server with an ima_data4 field whose length is larger than 4096 bytes, the server MUST respond with NFS4ERR_INVALID.

When an NFS SETATTR is presented to an NFS version 4.2 server and the target object resides in a file system which supports FATTR4_IMA but the object itself does not support the FATTR4_IMA attribute, the server MUST respond with NFS4ERR_WRONGTYPE. For example, if the server's file system supports associating IMA metadata with regular

Lever

Expires December 8, 2019

[Page 8]

files but not with sockets or FIFOs, then the result of an attempt to associate IMA metadata with a FIFO will be NFS4ERR_WRONGTYPE.

When an NFS SETATTR is presented to an NFS version 4.2 server but the target object resides in a file system which does not support the FATTR4_IMA attribute, the server MUST respond with NFS4ERR_ATTRNOTSUPP.

When a client presents an NFS SETATTR that modifies FATTR4_IMA along with other attributes and the server responds with an error, the client can retry setting each attribute separately to sort out which attribute is causing the server to reject the NFS SETATTR operation.

A detailed description of the NFS SETATTR operation can be found in [Section 18.30 of \[RFC5661\]](#).

4.3.1. Sending IMA Metadata When Creating a New Object

An alternate way to set an attribute is to provide the attribute during an NFS OPEN(CREATE) operation. Upon creation, an object has no content to protect. If a client presents an FATTR4_IMA attribute to an NFS version 4.2 server during NFS OPEN(CREATE), the server MUST respond with NFS4ERR_INVALID.

4.3.2. Authorizing Updates to IMA Metadata

An NFS version 4.2 server needs to ensure that modifications to IMA metadata are done only by appropriately authorized agents. Although access to file content is typically controlled by ACLs and permission bits, these mechanisms do not apply to IMA metadata. The question of "who is authorized to modify IMA metadata" is left to local security policy.

Possible implementations include limiting IMA metadata update authority to:

Particular users

The server may allow IMA metadata updates only by UID 0 or by a client's machine principal.

Particular clients

The server may allow IMA metadata updates only from specific client IP addresses.

File owners

The server may allow IMA metadata updates only by the file's owner or group owner.

Lever

Expires December 8, 2019

[Page 9]

[cel: this is still an open issue.]

4.4. Retrieving IMA Metadata

An NFS version 4.2 client retrieves IMA metadata by retrieving the FATTR4_IMA attribute via an NFS GETATTR operation, specifying the file handle of the object associated with the metadata to be retrieved.

The IMA subsystem typically manages its own cache of this metadata to maintain reasonable performance. The NFS client implementation **MUST** always pass retrieval requests for this metadata to the server. This metadata **MUST NOT** be cached by the NFS client.

When an NFS GETATTR is presented to an NFS version 4.2 server and the target object resides in a file system which supports the FATTR4_IMA attribute but the object does not support the FATTR4_IMA attribute, the server **MUST** respond with NFS4ERR_WRONGTYPE. For example, if the server's file system supports associating IMA metadata with regular files but not named attributes, then the result of an attempt to retrieve IMA metadata on a named attribute will be NFS4ERR_WRONGTYPE.

When an NFS GETATTR is presented to an NFS version 4.2 server but the target object resides in a file system which does not support FATTR4_IMA, this does not result in an error and the FATTR4_IMA attribute bit is cleared in the server's response.

Otherwise, if the target object supports FATTR4_IMA and there is no IMA metadata is available for the target object, the server returns a FATTR4_IMA attribute whose length is zero.

When a client presents an NFS GETATTR that retrieves FATTR4_IMA along with other attributes and the server responds with an error, the client can retry by retrieving each attribute separately to sort out which attribute is causing the server to reject the NFS GETATTR operation.

A detailed description of the NFS GETATTR operation can be found in [Section 18.7 of \[RFC5661\]](#).

4.5. Using NFS Attribute Fencing (VERIFY/NVERIFY)

The NFS VERIFY and NVERIFY operations, described in Sections [18.31](#) and 18.15 of [\[RFC5661\]](#) respectively, permit a client to add a fence in an NFS COMPOUND where, if a provided FATTR4 attribute does or does not match, the server can force processing of that COMPOUND to stop. The FATTR4_IMA attribute is a valid choice for these operations.

Lever

Expires December 8, 2019

[Page 10]

The server MUST use a simple byte comparison to evaluate whether the client-provided FATTR4_IMA matches the FATTR4_IMA attribute associated with the target object. If the server has a local IMA implementation, it MAY prevent the use of the local FATTR4_IMA attribute value for the purpose of this comparison (via EVM protection). If the client has indicated support for IMA metadata, the server MUST respond with NFS4ERR_INTEGRITY. Otherwise it MUST respond with NFS4ERR_ACCESS.

5. Deployment Examples

5.1. Terminology

Because the protocol extension described in this document is OPTIONAL, clients and servers that support it will necessarily interact with clients and servers that do not support it. To aid the discussion in this section, we define the following terms:

Integrity Assessor: A security module separate from the file system that appraises file content based on a policy and IMA measurement results.

Participating Client: An NFS version 4.2 client that employs a integrity assessor, supports the OPTIONAL extension described in this document, and indicates this support to servers using the handshake described in [Section 4.2](#).

Non-participating Client: An NFS client that does not support the OPTIONAL extension described in this document.

Participating Server: An NFS version 4.2 server that supports the OPTIONAL extension described in this document, indicates this support to clients using the handshake described in [Section 4.2](#), and its shared file systems can store IMA metadata. A participating server is not required to implement an integrity assessor.

Non-participating Server: An NFS server that does not support the OPTIONAL extension described in this document.

In addition, there are intermediate modes of operation on participating peers:

Full-function Client: A participating client that can modify IMA metadata via NFS.

Fetch-only Client: A participating client that does not support modifying IMA metadata on a participating server.

Lever

Expires December 8, 2019

[Page 11]

Full-function Server: A participating server that has a local user execution environment and supports updating IMA metadata that resides on local shared file systems.

Store-only Server: A participating server where there is only remote access to file content and IMA metadata.

Lastly, we provide the following possible simple appraisal policies that might be applied by an integrity assessor:

Strict: Access is prevented to a file's content if the file has no IMA metadata or if the extant IMA metadata fails to verify the file content. Otherwise access to the file's content is not prevented.

Audit: Access to a file's content is never prevented. Warnings are reported when a file has no IMA metadata or when extant IMA metadata fails to verify the file's content.

Disabled: Access to file content is never prevented and IMA metadata is ignored.

5.2. Instantiating IMA Metadata

Once a file is written and closed, a specialized tool generates and signs the IMA metadata and then writes it to the file system. The tool can be used on a full-function client to sign files on a participating server. Or, the tool can be used on a full-function server to sign local files. The IMA metadata is then visible to participating clients and local users on the server (if there are any).

Typically, once IMA metadata is associated with a file, the file's content is essentially immutable, even if the file's permissions settings permit writing to it. This is because changing the content without updating the associated IMA metadata will make the file's content inaccessible, depending on the appraisal policy in effect. Updating the file content requires generating fresh IMA metadata to prevent subsequent IMA appraisal failures.

5.3. Interaction With Non-Participating Implementations

Given the example policies and definitions we provided earlier, the following statements are true:

- o A participating client that uses the Disabled policy is equivalent to a non-participating client, except that a participating server

Lever

Expires December 8, 2019

[Page 12]

is allowed to respond with NFS4ERR_INTEGRITY to a participating client.

- o A non-participating client never prevents access to file content on a participating server, but a participating server that has a local integrity assessor may prevent access to a non-participating client.
- o A participating client using the Strict policy never allows access to files stored on a non-participating server.

An integrity assessor on a participating NFS version 4.2 peer needs to be prepared to deal gracefully with IMA metadata it does not recognize or cannot parse. Its policy may treat this case as a appraisal failure.

It is not required for an NFS version 4.2 server to implement an integrity assessor. However, some servers, such as the Linux NFS server, do just that, applying local IMA policy to both local and remote file accesses.

If an appraisal failure occurs during a remote access, the server responds to a non-participating client with NFS4ERR_ACCESS or NFS4ERR_IO. The server's local policy can decide that a participating client sees NFS4ERR_INTEGRITY, or rather that the server allows access to the file content and IMA metadata so that the client's own IMA policies can be fully applied.

[cel: Permitting remote access to files that fail a local integrity check is still an open issue.]

6. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [\[RFC7942\]](#). The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs.

Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

Lever

Expires December 8, 2019

[Page 13]

6.1. Linux NFS server and client

Organization: The Linux Foundation

URL: <https://www.kernel.org>

Maturity: Prototype software based on early versions of this document.

Coverage: The bulk of this specification is implemented.

Licensing: GPLv2

Implementation experience: No comments from implementors.

7. Security Considerations

The design of the NFS extension described in this document assumes that all IMA metadata is cryptographically signed to prevent unwanted alteration at rest or in transit.

When IMA metadata for a file exists and the end host has an IMA assessor, the content of a file is protected from creation to use. Receivers can reliably detect unintentional or malicious alteration of file content by verifying its content using the file's IMA metadata. Additional protection of file content while at rest or in transit on an untrusted network is unnecessary.

Likewise, receivers can also reliably detect unintentional or malicious alteration of IMA metadata that is cryptographically signed, simply by verifying its signature. Additional protection of signed metadata while at rest or in transit on an untrusted network is unnecessary.

Like other mechanisms that protect data integrity during transit, a malicious agent or a network malfunction can create a denial-of-service condition by repeatedly triggering integrity verification failures on NFS version 4.2 clients.

To prevent a malicious denial-of-service attempt by altering IMA metadata at rest, an NFS version 4.2 server can enforce a suitable level of privilege before authorizing a local or remote agent to alter this information. See [Section 4.3.2](#) for more detail.

Lever

Expires December 8, 2019

[Page 14]

8. IANA Considerations

This document has no IANA actions.

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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4506] Eisler, M., Ed., "XDR: External Data Representation Standard", STD 67, [RFC 4506](#), DOI 10.17487/RFC4506, May 2006, <<https://www.rfc-editor.org/info/rfc4506>>.
- [RFC5661] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 Protocol", [RFC 5661](#), DOI 10.17487/RFC5661, January 2010, <<https://www.rfc-editor.org/info/rfc5661>>.
- [RFC7862] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 Protocol", [RFC 7862](#), DOI 10.17487/RFC7862, November 2016, <<https://www.rfc-editor.org/info/rfc7862>>.
- [RFC7863] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 External Data Representation Standard (XDR) Description", [RFC 7863](#), DOI 10.17487/RFC7863, November 2016, <<https://www.rfc-editor.org/info/rfc7863>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", [BCP 205](#), [RFC 7942](#), DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8178] Noveck, D., "Rules for NFSv4 Extensions and Minor Versions", [RFC 8178](#), DOI 10.17487/RFC8178, July 2017, <<https://www.rfc-editor.org/info/rfc8178>>.

Lever

Expires December 8, 2019

[Page 15]

9.2. Informative References

- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", [RFC 2104](#), DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.
- [RFC4158] Cooper, M., Dzambasow, Y., Hesse, P., Joseph, S., and R. Nicholas, "Internet X.509 Public Key Infrastructure: Certification Path Building", [RFC 4158](#), DOI 10.17487/RFC4158, September 2005, <<https://www.rfc-editor.org/info/rfc4158>>.
- [RFC5662] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 External Data Representation Standard (XDR) Description", [RFC 5662](#), DOI 10.17487/RFC5662, January 2010, <<https://www.rfc-editor.org/info/rfc5662>>.
- [RFC7861] Adamson, A. and N. Williams, "Remote Procedure Call (RPC) Security Version 3", [RFC 7861](#), DOI 10.17487/RFC7861, November 2016, <<https://www.rfc-editor.org/info/rfc7861>>.
- [SAILER] Sailer, R., Zhang, X., Jaeger, T., and L. van Doorn, "Design and Implementation of a TCG-based Integrity Measurement Architecture", Proceedings of the 13th USENIX Security Symposium, August 2004.

9.3. URIs

- [1] https://trustedcomputinggroup.org/wp-content/uploads/Trusted-Platform-Module-Summary_04292008.pdf

Acknowledgments

The author wishes to thank Mimi Zohar and James Morris for their early review of the concepts in this document, Wim Coekaerts for his encouragement of this work, and Dave Noveck for his work on NFS version 4 extensibility.

The author wishes to acknowledge review comments from Dave Noveck, Craig Everhart, and Bruce Fields which helped to make this a better document.

The XDR extraction conventions were first described by the authors of the NFS version 4.1 XDR specification [[RFC5662](#)]. Herbert van den Bergh suggested the replacement sed script used in this document.

Lever

Expires December 8, 2019

[Page 16]

Special thanks go to Transport Area Director Magnus Westerlund, NFSV4 Working Group Chairs Spencer Shepler and Brian Pawlowski, and NFSV4 Working Group Secretary Thomas Haynes for their support.

Author's Address

Charles Lever
Oracle Corporation
United States of America

Email: chuck.lever@oracle.com