Network File System Version 4 C. Lever
Internet-Draft Oracle
Intended status: Standards Track May 29, 2018
Expires: November 30, 2018

Integrity Measurement for Network File System version 4
draft-ietf-nfsv4-integrity-measurement-00

Abstract

This document specifies an OPTIONAL extension to NFS version 4.2 that enables Linux Integrity Measurement Architecture metadata (IMA) to be conveyed between NFSv4.2 servers and clients.

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 November 30, 2018.

Table of Contents

1. Introduction

The Linux Integrity Measurement Architecture (IMA) provides assurance that the content of files is unaltered and authentic to what was originally written to those files. In addition, an Extended Verification Module (EVM) assures that file attribute information remains similarly unaltered.

The primary goal is to detect when a remote attacker, a local attacker, or unintentional software behavior has modified the content or attributes of a file either in transit or at rest. This is done by separately storing HMAC hashes [RFC2104] of a file's byte stream content and attribute metadata. These hashes are updated whenever the file is legitimately modified. The hashes themselves can be protected by cryptographic signature.

Subsequent integrity verification is not performed by applications or by file systems. File systems are responsible only for persistent storage of file content and hashes. When a file is read, content, attributes, and hashes are passed to IMA and EVM modules for measurement and appraisal. Application access is denied if the hashes or their signatures cannot be verified.

Some files may be immutable, in which case their integrity metadata is signed by an RSA public key signature [RFC8017]. Such files can be accessed in read-only mode or deleted by an appropriately privileged agent, but cannot otherwise be modified.

Key material used to sign and verify file content and attribute metadata must be protected. A Trusted Platform Module [TPM-SUM] can

be used to seal the key material. This use case is typical for providing a read-only operating system image that is cryptographically verified; for example, in a cloud environment or on mobile devices.

On Linux, there are two parts to a file's IMA metadata:

- An HMAC hash, possibly cryptographically signed, of the file's byte stream, stored in the file's security.ima extended attribute.

- An HMAC hash, possibly cryptographically signed, of a subset of the file's attributes, stored in the file's security.evm extended attribute.

The goals and use cases of the Linux Integrity Measurement Architecture (IMA) are presented in further detail in [IMA-WP].

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 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 NFSv4 minor version 2 [RFC7862]. NFSv4.2 servers and clients implemented without knowledge of this extension will continue to interoperate with NFSv4.2 clients and servers that are aware of the extension, whether or not they support it.

Because [RFC7862] does not define NFSv4.2 as non-extensible, [RFC8178] treats it as an extensible minor version. Therefore this Standards Track RFC extends NFSv4.2 but does not update [RFC7862] or [RFC7863].

3.1. XDR Extraction

Section 4.1 contains a description of an extension to the NFSv4.2 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.

<CODE BEGINS>

sed -E 's:^ *// ?::;t;d'

<CODE ENDS>

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

<CODE BEGINS>

sed -E 's:^ *// ?::;t;d' < integrity-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 structure used to transport HMAC data, and two new OPTIONAL GETATTR attributes used to access and update HMAC data associated with a file.

HMAC data is a varying length opaque array of octets. To enable a single HMAC payload to be retrieved or updated in a single RPC, an HMAC (including signatures) MUST NOT exceed 4096 bytes in length.

<CODE BEGINS>

```
/// /*
///  * Copyright (c) 2018 IETF Trust and the person identified
///  * as author of the code.  All rights reserved.
///  *
///  * The author of the code is: C. Lever
///  */
///
/// const IMA_HMAC_MAXSIZE   = 4096;
///
/// typedef opaque             ima_hmac4<IMA_HMAC_MAXSIZE>;
/// typedef opaque             uuid4[16];
/// typedef opaque             verified_attribute4<>;
///
/// struct evm_verflist4 {
///       uuid4                *evm_uuid;
///       verified_attribute4  evm_attrlist<>;
/// };
///
/// %/*
/// % * New For Integrity Measurement
/// % */
/// const FATTR4_IMA_HMAC_CONTENT = 85;
/// const FATTR4_IMA_HMAC_ATTR    = 86;
/// const FATTR4_EVM_VERF_LIST    = 87;
///
/// typedef ima_hmac4          fattr4_ima_hmac_content;
/// typedef ima_hmac4          fattr4_ima_hmac_attr;
/// typedef evm_verflist4      fattr4_evm_verf_list;
```

<CODE ENDS>

Name	Id	Data Type	Acc	Defined in
IMA_HMAC_CONTENT	85	ima_hmac4	W	Section 4.2
IMA_HMAC_ATTR	86	ima_hmac4	W	Section 4.3
EVM_VERF_LIST	87	evm_verflist4	W	Section 4.3

Table 1

When an NFSv4.2 server does not recognize, or does recognize but does not support, these new attributes, it must respond according to the requirements defined in Section 4.3 of [RFC8178].

4.2. Protecting File Content

4.2.1. Computing an IMA HMAC on an NFS File

Integrity measurement is performed on the entirety of a file's primary byte stream. When a file is first accessed, after file content changes, or if any portion of a file is evicted from an NFSv4.2 client's cache, the file's entire byte stream must be read in order for the IMA module to verify that the stored HMAC matches the just-computed HMAC. This requirement can incur a significant performance impact for large files or files that change frequently.

An NFSv4.2 client may employ mechanisms, not specified here, to reduce this performance impact. For example, instead of signing a hash of the file's byte stream, a Merkle tree can be constructed that allows clients to verify the integrity of smaller portions of a large file [MERKLE]. The top hash of that tree can be signed instead of signing the HMAC of the file content. This Merkle tree is then used to verify subsections of the file's byte stream that are needed by applications running on an NFSv4.2 client.

4.2.2. Storing an IMA HMAC

An NFSv4.2 client stores the HMAC of an NFS file's byte stream by sending a SETATTR operation that specifies the FATTR4_IMA_HMAC_CONTENT attribute. This HMAC completely replaces the previous one. To remove the FATTR4_IMA_HMAC_CONTENT attribute from a file, the client specifies an ima_hmac_data field whose length is zero.

When a SETATTR is presented to an NFSv4.2 server with a credential that is unauthorized to replace the FATTR4_IMA_HMAC_CONTENT attribute, the server MUST respond with NFS4ERR_ACCESS. This document does not specify a policy for authorizing changes to the FATTR4_IMA_HMAC_CONTENT attribute.

When a SETATTR is presented to an NFSv4.2 server with an ima_hmac_data field that is larger than IMA_HMAC_MAXSIZE, the server MUST respond with NFS4ERR_NAMETOOLONG.

When a SETATTR is presented to an NFSv4.2 server that supports FATTR4_IMA_HMAC_CONTENT, but the SETATTR targets an object which does not support this attribute, the server MUST respond with NFS4ERR_TYPE.

Likewise, an NFSv4.2 client retrieves the HMAC of an NFS file's byte stream by retrieving the FATTR4_IMA_HMAC_CONTENT attribute via a GETATTR operation. This HMAC may have been computed (and signed)

previously on this client or by some other agent. An NFSv4.2 server MUST NOT prevent an NFSv4.2 client from accessing a file based on HMAC verification failures on the server.

4.3. Protecting File Attributes

4.3.1. Computing an EVM HMAC on an NFS File

The EVM HMAC protects a subset of file attributes, referred to as "verified attributes". There are several categories of verified attributes:

- Normal (N). Such an attribute is always present on a file, and is always verified by the file's EVM HMAC hash.

- Extended (E). Such an attribute is a Linux extended attribute that may be present on a file. If this attribute is present on a file, it is always verified by the file's EVM HMAC hash.

- File system (F). Such an attribute is a file system attribute (ie., its value is the same for all files that share the same FSID). If the local system configuration calls for it, this attribute is verified by a file's EVM HMAC hash.

- Optional (O). Such an attribute is a Linux extended attribute that may be present on a file. If the local system configuration calls for it, this attribute is verified by a file's EVM HMAC hash.

The following table enumerates all attributes that may be a verified attribute. File attributes not listed in this table are never included when computing an EVM HMAC hash.

Name	Cat	NFSv4 equivalent	Notes
inode number	N	fattr4_fileid	1
inode generation	N	fattr4_change	1
UID	N	fattr4_owner	2
GID	N	fattr4_owner_group	2
file mode	N	fattr4_mode	3
security.selinux	E	NFSv4 SecLabel	4
security.ima	E	fattr4_ima_hmac_content	
security.SMACK64	E	None	5
security.capabilities	E	None	5
File system UUID	F	fattr4_evm_verf_list	
security.SMACK64EXEC	O	None	5
security.SMACK64TRANSMUTE	O	None	5
security.SMACK64MMAP	O	None	5

Table 2

Notes:

1. NFSv4.2 server and client implementations are responsible for ensuring that the native representation of these values correctly correspond anywhere that an EVM HMAC is to be computed.

2. The NFSv4 ID mapping configuration must ensure that numeric user ID values map identically whereever an EVM HMAC is to be computed.

3. Some implementations may alter the file mode bits depending on the presence of ACLs or a umask.

4. The security.selinux extended attribute typically maps to NFSv4 Security Label LFS value 0.

5. This attribute is not exposed in current versions of NFSv4, but may be exposed by future extensions to the NFSv4 protocol, or it may be accessible by other means.

The EVM HMAC for a file must be updated whenever one or more verified attributes changes. Whenever a verified attribute is present on a stored file, it MUST be accessible to the NFSv4.2 client computing that HMAC. Otherwise, if that attribute is present and is not accessible when the HMAC is verified, the client's EVM module denies access to that file.

To verify an EVM HMAC, the verifier must know which optional attributes were included when the hash was originally computed. An NFSv4.2 server advertises this via the per file system FATTR4_EVM_VERF_LIST attribute (Section 5.4 of [RFC5661] defines the meaning of the term "per file system attribute"). If EVM HMACs are supported on a file system, the server MUST also support and expose FATTR4_EVM_VERF_LIST.

If an FS UUID is included in the EVM HMACs on this file system, the server MUST fill in the evm_uuid field with its value. The server MUST leave this field empty if the FS UUID is not included in EVM HMACs. The server MUST append the names of all optional extended attributes included in the EVM HMACs on this file system.

EVM HMAC hashes may be computed either on the NFSv4.2 server that contains the files, or by NFSv4.2 clients. If an NFSv4.2 server or a shared file system can store these hashes, but the hashes are always computed elsewhere, the server still has to advertise the attribute input set that is used to compute EVM HMACs on this file system.

An NFSv4.2 server MAY allow modifications of FATTR4_EVM_VERF_LIST by NFSv4.2 clients. Changes to the value of this attribute render existing EVM HMACs on this file system invalid. The format of the FATTR4_EVM_VERF_LIST attribute a client submits via SETATTR is the same as the the format the server uses when returning this attribute via GETATTR.

If an NFSv4.2 client attempts to update this attribute and provides one or more unrecognized optional extended attributes, the server MUST respond with NFS4ERR_INVAL. If an NFSv4.2 client attempts to update this attribute and the server does not support changes to it by clients, the server MUST respond with NFS4ERR_INVAL, as required by Section 5.5 of [RFC5661]. This document does not specify a policy for authorizing changes to the FATTR4_EVM_VERF_LIST attribute.

4.3.2. Storing an EVM HMAC

An NFSv4.2 client stores the HMAC of an NFS file's attributes by sending a SETATTR operation that specifies the FATTR4_IMA_HMAC_ATTR attribute. This HMAC completely replaces the previous one. To remove the FATTR4_IMA_HMAC_ATTR attribute from a file, a client specifies an ima_hmac_data field whose length is zero.

When a SETATTR is presented to an NFSv4.2 server with a credential that is unauthorized to replace the FATTR4_IMA_HMAC_ATTR attribute, the server MUST respond with NFS4ERR_ACCESS. This document does not specify a policy for authorizing changes to the FATTR4_IMA_HMAC_ATTR attribute.

When a SETATTR is presented to an NFSv4.2 server with an ima_hmac_data field that is larger than IMA_HMAC_MAXSIZE, the server MUST respond with NFS4ERR_NAMETOOLONG.

When a SETATTR is presented to an NFSv4.2 server that supports FATTR4_IMA_HMAC_ATTR, but the SETATTR targets an object which does not support this attribute, the server MUST respond with NFS4ERR_TYPE.

Likewise, an NFSv4.2 client retrieves the HMAC of an NFS file's attributes by retrieving the FATTR4_IMA_HMAC_ATTR attribute via a GETATTR operation. This HMAC may have been computed (and signed) previously on this client or by some other agent. An NFSv4.2 server MUST NOT prevent an NFSv4.2 client from accessing a file based on HMAC verification failures on the server.

5. Security Considerations

An NFSv4.2 server is required to enforce a suitable level of privilege before permiting a local or remote agent to alter IMA or EVM HMAC hashes. This document does not specify a policy for authorizing replacement of IMA or EVM HMAC hashes.

When protected by both IMA and EVM HMAC hashes, the content of a file and its attributes are protected from end-to-end. Receivers can reliably detect unintentional or malicious alteration of file content or attributes by verifying the HMAC hashes that cover them. Additional protection of such content or attributes while in transit on an untrusted network is not required.

When an HMAC is cryptographically signed, receivers can reliably detect unintentional or malicious alteration simply by verifying its signature. Additional protection of a signed HMAC while in transit on an untrusted network is not required.

In cases when IMA and EVM HMAC hashes are not otherwise cryptographically protected, these hashes MUST be protected while in transit on an untrusted network using a cryptographically strong transport layer security service that can detect tampering, such as RPCSEC with an integrity-protecting service [RFC7861].

Like other mechanisms that protect integrity during transit, it is possible for a malicious agent or a network malfunction to create a denial-of-service condition by repeatedly triggering integrity verification failures on clients.

6. IANA Considerations

This document does not require any actions by IANA.

7. References

7.1. 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>.

[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>.

7.2. Informative References

[IMA-WP] Safford, D., "An Overview of The Linux Integrity Subsystem", <http://downloads.sf.net/project/linux-ima/linux-ima/Integrity_overview.pdf>.

[MERKLE] Merkle, R., ""A Digital Signature Based on a Conventional Encryption Function" Advances in Cryptology - CRYPTO '87", DOI 10.1007/3-540-48184-2_32, 1988.

[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>.

[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>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[TPM-SUM] Trusted Computing Group, "Trusted Platform Module (TPM) Summary", April 2008, <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, and Wim Coekaerts for his encouragement of this work. Great thanks to Calum Mackay for his improved XDR extraction script.

Special thanks go to Transport Area Director Spencer Dawkins, NFSV4 Working Group Chair Spencer Shepler, and NFSV4 Working Group Secretary Thomas Haynes for their support.

Author's Address

Charles Lever
Oracle Corporation
1015 Granger Avenue
Ann Arbor, MI 48104
United States of America

Phone: +1 248 816 6463
Email: chuck.lever@oracle.com