Network File System Version 4 C. Lever
Internet-Draft Oracle
Intended status: Standards Track April 9, 2018
Expires: October 11, 2018

Linux-related Extensions to NFS version 4.2 Security Labels
draft-cel-nfsv4-linux-seclabel-xtensions-00

Abstract

NFS version 4.2 introduces an optional feature known as NFSv4 Security Labels. This document extends NFSv4 Security Labels to support Linux file capabilities and the Linux Integrity Measurement Architecture.

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 October 11, 2018.

Table of Contents

1. Introduction

In the current document, we define a way to manage both IMA metadata and Linux file capabilities on NFS files using existing NFSv4 Security Label protocol elements. Before specifying new protocol, let's contextualize and define the terminology used in the body of this document.

1.1. Linux Integrity Measurement Architecture

The Linux Integrity Measurement Architecture (hereafter, IMA) provides assurance that the content of a file is unaltered and authentic to what was originally written to that file. In addition, an Extended Verification Module (hereafter, EVM) can assure that a file's attribute information remains unaltered.

The goal is to detect when a remote attacker, a local attacker, or unintentional software behavior has modified the content or attributes of a file. This is done by cryptographically signing HMAC hashes of a file's content and attribute metadata, and then storing the signed hashes separately. These hashes are typically updated whenever the file is legitimately modified.

Integrity verification is not performed by applications or by file systems. Applications are not exposed to the operation of integrity verification. File systems are responsible only for persistent storage of file content and signed hashes. When a file is first read, content and hashes are passed to IMA modules for measurement

and appraisal. Application access is denied if the hashes cannot be verified.

Some files may be immutable, in which case their integrity metadata is signed by an RSA public key signature [RFC8017]. These files can only be accessed in read-only mode, or deleted by a privileged process.

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.

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

1.2. Linux File Capabilities

The Linux kernel divides privileges traditionally associated with the superuser into distinct subprivileges, referred to as "capabilities". These capabilies include but are not limited to

- The ability to override file read and write permission checks
- The ability to signal processes without permission checks
- The ability to bind to a privileged socket port or perform other privileged network administration tasks
- The ability to perform a range of system administrative tasks such as rebooting or setting the system clock

Being broken out from the monolithic superuser privilege, individual capabilities can be enabled and disabled independently of one another. Individual capabilities are associated with a process and are inherited during execve(2), similar to the traditional superuser privilege.

File capabilities enable capabilities to be associated with a file containing an executable object, like a setuid permission bit. File capability sets are combined with the capability sets of a parent process to determine the capabilities of a child process after an execve(2).

This enables an otherwise unaware application to be given a particular and specific privilege -- say, the privilege to bind to a

privileged port -- without granting it the privilege to reboot the system or kill processes that do not share its owner UID, thus improving overall system security.

The Linux capability implementation is based on the withdrawn POSIX.1e draft standard (see [POSIX1e]). An overview of the facility can be found in the Linux capabilities(7) man page (citation needed).

1.3. NFSv4 Security Labels

Section 9 of the NFS version 4.2 specification [RFC7862] defines an optional feature, known as NFSv4 Security Labels, that permits per-file extended security labels to be conveyed between NFSv4 clients and servers.

The intention of these labels is to enable the conveyance of MAC security labels, but in special cases, non-MAC security information may be handled. There is no prohibitory language in [RFC7204], [RFC7569], or [RFC7862] which excludes non-MAC security metadata from being conveyed via an NFSv4 Security Label, for instance.

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. Handling File Capabilities on NFS Mounts

Linux file capabilities are typically manipulated in a native binary format. However, a capability set can be converted to a human-readable text format. See the Linux cap_from_text(3) man page (citation needed).

An NFSv4 client stores the capability set associated with an NFS file by first converting the capability set to a text format and then conveying it to an NFSv4 server via a SETATTR operation, using the LFS number assigned for this purpose (see Section 6). This capability set completely replaces the previous capability set for that file.

Likewise, an NFSv4 client retrieves the capability set associated with a file by first retrieving the text form of the capability set, as stored in an NFSv4 Security Label, via a GETATTR operation, using the LFS number assigned for this purpose, and then converting the

retrieved text form into the client's native capability set representation.

There is no MAC policy associated with the file capabilities LFS number. The contents of the lfs_pi field MUST be ignored by clients and servers when this LFS number is used. See Section 12.2.4 of [RFC7862] for further details about how clients and servers form these GETATTR and SETATTR requests.

In order to enable file capabilities to be retrieved or updated in a single RPC, the text format representation of a capability set MUST NOT exceed 8192 bytes in length.

This document does not specify how an NFSv4 server handles file capability metadata. Server implementation choices include:

- If there are accessors of shared files that are local to an NFSv4 server, the server may choose to store file capabilities in a native format and convert them to a text form when an NFSv4 client retrieves them.

- Alternately, an NFSv4 server may itself not support file capabilities at all, in which case the server can store capability sets in text form without conversion to a native format.

4. Handling IMA Metadata on NFS Mounts

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

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

- A cryptographically signed HMAC hash of the file's attributes, stored in the file's security.evm extended attribute

An NFSv4 client stores the signed hash of an NFS file's byte stream or attributes by conveying the hash to an NFSv4 server via a SETATTR operation, using the LFS number appropriate for the hash (see Section 6). This signed hash completely replaces the previous hash. This document does not specify a policy for authorizing changes to IMA or EVM hashes.

Likewise, an NFSv4 client retrieves the signed hash of an NFS file's byte stream or attributes by retrieving the appropriate NFSv4 Security Label via a GETATTR operation using the LFS number assigned for this purpose. An NFSv4 server MUST NOT prevent an NFSv4 client from accessing a file based on IMA verification failures on the server.

There is no MAC policy associated with the IMA or EVM LFS numbers. The contents of the lfs_pi field MUST be ignored by clients and servers when these LFS numbers are used. See Section 12.2.4 of [RFC7862] for further details about how clients and servers form these GETATTR and SETATTR requests.

In order to enable IMA metadata to be retrieved or updated in a single RPC, a signed hash MUST NOT exceed 4096 bytes in length.

This document does not specify how an NFSv4 server handles IMA metadata. If there are local accessors of shared files on an NFSv4 server, the server may choose to store IMA metadata in a native format that can be handled by the server's local integrity modules.

A note about performance: IMA measurement and appraisal is always performed on the entirety of a file's byte stream. In other words, a file's entire byte stream must be read over to an NFSv4 client in order for its IMA module to verify its integrity. It is recognized that this requirement can have a significant performance impact for large files. An NFSv4 client may employ other 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, and that tree can be signed instead of signing the file content.

5. Security Considerations

An NFSv4 server is required to enforce a suitable level of privilege before allowing a local or remote agent to alter NFSv4 Security Labels. Consult Section 9.6 of [RFC7862] for further details.

This document does not specify a policy for authorizing changes to IMA or EVM metadata or file capabilities.

5.1. Protection of File Capability Metadata

File capabilities are conveyed as text strings. To prevent a man-in-the-middle from escalating the capability set of a file in transit, these strings must be protected in transit using a GSS service that provides integrity verification [RFC7861]. A server response that indicates that a file has no associated file capabilities must be similarly cryptographically protected. While at rest on durable storage or in a cache, capability metadata requires the same protections as other file attribute metadata.

5.2. Protection of IMA Metadata

IMA metadata is cryptographically signed. Receivers can detect unintentional and malicious alteration of this metadata simply by verifying the signature. Therefore additional protection using GSS [RFC7861] or other security mechanisms is not mandatory.

6. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding the addition of new entries in the "Security Label Format Selection Registry" in accordance with Section 5.2 of [RFC7569].

Label Format Specifier	Description	Status	Reference
1	LINUX-IMA	active	[RFC-TBD]
2	LINUX-EVM	active	[RFC-TBD]
3	LINUX-FCAP	active	[RFC-TBD]

New entries in the Security Label Format Selection Registry

Table 1

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>.

[RFC7569] Quigley, D., Lu, J., and T. Haynes, "Registry Specification for Mandatory Access Control (MAC) Security Label Formats", RFC 7569, DOI 10.17487/RFC7569, July 2015, <https://www.rfc-editor.org/info/rfc7569>.

[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>.

[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>.

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>.

[POSIX1e] POSIX, "POSIX.1e Specification", <http://wt.tuxomania.net/publications/posix.1e/>.

[RFC7204] Haynes, T., "Requirements for Labeled NFS", RFC 7204, DOI 10.17487/RFC7204, April 2014, <https://www.rfc-editor.org/info/rfc7204>.

[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 thanks Trond Myklebust for suggesting these extentions to NFSv4 Security Labels. 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