Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 15, 2011

D. Quigley

J. Morris
Red Hat, Inc.
March 14, 2011

MAC Security Label Requirements for NFSv4
draft-quigley-nfsv4-sec-label-requirements-03.txt

Abstract

This Internet-Draft outlines high-level requirements for the integration of flexible Mandatory Access Control (MAC) functionality into NFSv4.1 . It describes the level of protections that should be provided over protocol components and the basic structure of the proposed system. It also gives a brief explanation of what kinds of protections MAC systems offer and why existing NFSv4 protection mechanisms are not sufficient.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 15, 2011.

Table of Contents

1. Introduction

Mandatory Access Control (MAC) systems are becoming mainstreamed in modern operating systems such as Linux (R), FreeBSD (R), Solaris (TM), and Windows Vista (R). Most MAC systems share several key concepts. MAC systems bind a security attribute to subjects and objects within a system. A subject is an active entity in the environment which requests access to resources. In traditional systems subjects usually take the form of processes. The resources that are accessed by subjects are referred to as objects. In modern operating systems these take the form of files, directories, sockets, etc. The security attributes from subjects and objects are used in the access decision which is made by another important part of a MAC system. The reference monitor is a component in the system which enforces access to all subjects and objects, is tamper proof, non-bypassable. The policy decision engine may be co-located with the reference monitor or separated. The policy decision engine takes all relevant information into account while making an access decision.

The existing NFSv4 specification provides two access control methods. Both of these mechanisms fall into a category of access control models called Discretionary Access Control (DAC). In these models access decisions are made based on the user's identity and ownership of the resource being requested. Since access decisions are based on ownership, users and the programs they execute are free to modify access rules. This provides little security against malicious or flawed applications since the program can modify access to any file owned by the effective user. Another issue with DAC systems is that their privileges are very coarse-grained making them prone to privilege escalation.

Mandatory Access Control systems differ from DAC systems in several important ways. Instead of basing access decisions only on user identity and ownership, MAC systems mediate control between subjects and objects based on a variety of security characteristics relevant to their respective security models. It also varies in that the access policy on the system is defined by the administrator or organization, and the user does not have the "discretion" of violating the policy or changing its rules. Since MAC system policy is defined in terms of a variety of security characteristics it can protect against flawed or malicious programs by only allowing subjects to behave in ways that are specified in the policy.

2. Problem Statement

In MAC systems every object is assigned a security label containing the security-relevant characteristics of the object. The existing

NFSv4 protocol contains a method for binding additional meta-data to a file-object namely Named Attributes. While this allows the system to bind a label to a file-object it does not provide the needed semantics. Named attributes are treated as a directory hierarchy with the attribute being stored in a file.

The attribute-as-files model is useful for enforcing application or user-defined semantics. However, MAC labeling has semantics that are defined and enforced by the system. Because of this there are several features that a labeling mechanisms should have. The first required feature is the ability to naturally support atomic reads and writes of attribute values, which prevents intermediate states from being visible. The second required feature is a way of atomically specifying the value of the attribute upon creation. This allows for a file to be created in its proper state and not be left in an unlabeled and potentially dangerous state before the correct label is applied. The last required feature is that label integrity must be preserved. The attribute-as-files model creates a recursive protection problem in this respect. What is the label of a label and how do you prevent labels from being directly manipulated by userspace as files?

In addition to the issues listed above Named Attributes (NAs) as they are defined in the NFSv4 specification conflict with some requirements of MAC labeling. NAs as defined by the specification are user managed and opaque to the system where MAC labels are managed by the system and have well defined semantics. Also, the semantics extend beyond simple label operations. Other security state relating to the client and the server needs to be maintained, negotiated and communicated via the protocol.

To accommodate this, a system needs a method that allows for the attribute to be set upon object creation and for the label to be bound to the underlying file system object. Some have suggested adding an extended attribute protocol to the NFSv4 specification, but extended attributes are neither standardized in their interface nor do they provide the necessary semantics. The only method currently in the NFSv4 specification that provides the required semantics is recommended attributes.

Without a method for providing per file-object labeling as described above existing MAC systems have fallen back to methods of handling file systems that lack labeling attributes. While this allows policy to provide coverage of NFSv4 file systems, there are several limitations, including:

- Labeling granularity is generally too coarse when applied to the entire filesystem.

- Any underlying security labeling of the exported file system is not conveyed over the network.

- Security labels cannot be set on the remote file system by the client.

When making access decisions MAC systems may also take into account all relevant security information about the process making the access. Currently there is no way for this information to be conveyed with the NFS request. While RPCSECGSS provides authentication in the form of user identity. This is not compatible with the MAC concept since there is more than user identity to consider in making access decisions and processes may have more complex sets of credentials than user identity, including attributes such as role, level.

3. Requirements

The following initial requirements have been gathered from users, developers, and from previous development efforts in this area such as DTOS [DTOS] and NSA's experimental NFSv3 enhancements [SENFSV3].

3.1. Portability & Interoperability

Labeled-NFS (LNFS) must be designed with portability in mind, to facilitate implementations on any operating system that supports mandatory access controls.

LNFS must be designed and developed to facilitate interoperability between different LNFS implementations.

LNFS modifications to standard NFSv4 implementations must not adversely impact any existing interoperability of those implementations.

3.2. Performance & Scalability

Security mechanisms often impact on system performance. LNFS should be designed and implemented in a way which avoids significant performance impact where possible.

As NFSv4 is designed for large-scale distributed networking, LNFS should also be capable of scaling in a similar manner to underlying implementations where possible.

LNFS should be respond in a robust manner to system and network outages associated with typical enterprise and Internet environments.

At the very least, LNFS should always operate in a fail-safe manner, so that service disruptions do not cause or facilitate security vulnerabilities.

3.3. Security Services

LNFS should ensure that the following security services are provided for all NFSv4 messaging. These services may be provided by lower layers even if NFS has to be aware of and leverage them:

- Authentication
- Integrity
- Privacy

Mechanisms and algorithms used in the provision of security services must be configurable, so that appropriate levels of protection may be flexibly specified per mandatory security policy.

Strong mutual authentication will be required between the server and the client for Full Mode operation Section 3.5.1.

MAC security labels and any related security state must always be protected by these security services when transferred over the network; as must the binding of labels and state to associated objects and subjects.

LNFS should support authentication on a context granularity so that different contexts running on a client can use different cryptographic keys and facilities.

3.4. Label Encoding ,Label Format Specifiers, and Labeling Authorities

Encoding of MAC labels and attributes passed over the network must be specified in a complete and unambiguous manner while maintaining the flexibility of MAC implementations. To accomplish this the labels should consist of an opaque component bound with a Label Format Specifier (LFS). The opaque component consists of the label which will be interpreted by the MAC model on the other end while the LFS provides a mechanism for identifying the structure and semantics of the label's components.

In MAC models, a Domain of Interpretation (DOI) represents a collection of systems, where all systems within the DOI have semantically coherent labeling. That is, a security label must always mean exactly the same thing anywhere within the DOI. While this may not be necessary for simple MAC models it is recommended

that most label formats assigned an LFS incorporate this concept into their label format.

LNFS must provide a means for servers and clients to identify their LFS and DOIs for the purposes of authorization, security service selection, and security label interpretation.

A negotiation scheme should be provided, allowing systems from different label formats and DOIs to agree on how they will interpret or translate each others labels. Multiple concurrent DOI agreements may be current between a server and a client.

All security labels and related security state transferred across the network must be tagged with a valid LFS and where applicable DOI.

If the LFS or DOI of a system changes, it should renegotiate agreements to reflect these changes.

If a system receives any security label or security state tagged with an LFS or DOI it does not recognize or cannot interpret, it must reject that label or state.

NFSv4 includes features which may cause a client to cross an LFS or DOI boundary when accessing what appears to be a single file system. If LFS and DOI negotiation is supported by the client and the server, the server should negotiate a new, concurrent agreement with the client, acting on behalf of the externally located source of the files.

LNFS should define an initial DOI negotiation scheme and DOI format with the primary aims of simplicity and completeness. This is to facilitate practical deployment of multi-DOI systems without being weighed down by complex and over-generalized global schemes. Future extensibility should also be taken into consideration.

3.5. Modes of Operation

LNFS must cater for two potentially concurrent operating modes, depending on the state of MAC functionality:

3.5.1. Full Mode

Both the server and the client have MAC functionality enabled, and full LNFS functionality is extended over the network between both client and server.

An example of an operation in full mode is as follows. On the initial lookup, the client requests access to an object on the

server. It sends its process security context over to the server. The server checks all relevant local policies to determine if that process context from that client is allowed to access the resource. Once this has succeeded the object with its associated security information is released to the client. Once the client receives the object it determines if its local policy allows the process running on the client access to the object.

On subsequent operations where the client already has a handle for the file, the order of enforcement is reversed. Since the client already has the security context it may make an access decision against its local policy first. This enables the client to avoid sending requests to the server that it knows will fail regardless of the server's policy. If the client passes the local policy check then it sends the request to the server where the client's process context is used to determine if the server will release that resource to the client. If both checks pass, the client is given the resource and everything succeeds.

In the event that the client does not trust the server, it may opt to use an alternate labeling mechanism regardless of the server's ability to return security information.

3.5.2. Guest Mode

Only one of the server or client has MAC functionality enabled.

In the case of the server only having MAC functionality enabled, the server locally enforces its policy, and may selectively provide standard NFS services to clients based on their authentication credentials and/or associated network attributes (e.g. IP address, network interface) according to security policy. The level of trust and access extended to a client in this mode is configuration-specific.

In the case of the client only having MAC functionality enabled, the client must operate as a standard NFSv4 client, and should selectively provide processes access to servers based upon the security attributes of the local process, and network attributes of the server, according to policy. The client may also override default labeling of the remote file system based upon these security attributes, or other labeling methods such as mount point labeling.

In other words, Guest Mode is standard NFSv4 over the wire, with the MAC-aware system mapping the MAC-unaware system's processes or objects to security labels based on other characteristics in order to preserve its local MAC guarantees.

3.6. Labeling

Implementations must validate security labels supplied over the network to ensure that they are within a set of labels permitted from a specific peer, and if not, reject them. Note that a system may permit a different set of labels to be accepted from each peer.

3.6.1. Client Labeling

At the client, labeling semantics for NFS mounted file systems must remain consistent with those for locally mounted file systems. In particular, user-level labeling operations local to the client must be enacted locally via existing APIs, to ensure compatibility and consistency for applications and libraries.

Note that this does not imply any specific mechanism for conveying labels over the network.

When an object is newly created by the client, it will calculate the label for the object based on its local policy. Once that is done it will send the request to the server which has the ability to deny the creation of the object with that label based on the server's policy. In creating the file the server must ensure that the label is bound to the object before the object becomes visible to the rest of the system. This ensures that any access control or further labeling decisions are correct for the object.

3.6.2. Server Labeling

The server must provide the capability for clients to retrieve security labels on all exported file system objects where possible. This includes cases where only in-core and/or read-only security labels are available at the server for any of its exported file systems.

The server must honor the ability for a client to specify the label of an object on creation. If the server is MAC enabled it may choose to reject the label specified by the client due to restrictions in the server policy. The server should not attempt to find a suitable label for an object in event of different labeling rules on its end. The server is allowed to translate the label into its DOI but should not change the semantic meaning of the label.

3.7. Policy Enforcement

3.7.1. Full Mode

The server must enforce its local security policy over all exported objects, for operations which originate both locally and remotely.

Requests from authenticated clients must be processed using security labels and credentials supplied by the client as if they originated locally.

As with labeling, the system must also take into account any other volatile client security state, such as a change in process security context via dynamic transition. Access decisions should also be made based upon the current client security label accessing the object, rather than the security label which opened it, if different.

The client must apply its own policy to remotely located objects, using security labels for the objects obtained from the server. It must be possible to configure the maximum length of time a client may cache state regarding remote labels before re-validating that state with the server.

The server must recall delegation of an object if the object's security label changes.

A mechanism must exist to allow the client to obtain access, and labeling decisions from the server for locally cached and delegated objects, so that it may apply the server's policy to these objects. If the server's policy changes, the client must flush all object state back to the server. The server must ensure that any flushed state received is consistent with current policy before committing it to stable storage.

Any local security state associated with cached or delegated objects must also be flushed back to the server when any other state of the objects is required to be flushed back.

3.7.2. Guest Mode

If the server is MAC aware and the client is not, the server must not accept security labels provided by the client, and only enforce its local policy to exported objects. In the event that the client is MAC aware while the server is not then the client may deny access or fall back on other methods for providing security labeling.

3.8. Namespace Access

The server should provide a means to authorize selective access to the exported file system namespace based upon client credentials and

according to security policy.

This is a common requirement of MLS-enabled systems, which often need to present selective views of namespaces based upon the clearances of the subjects.

4. Security Considerations

When either the client or server is operating in guest mode it is important to realize that one side is not enforcing MAC protections. Alternate methods are being used to handle the lack of MAC support and care should be taken to identify and mitigate threats from possible tampering outside of these methods.

5. IANA Considerations

It is requested that IANA creates a registry of Domain of Interpretation numbers to be consumed by a Domain of Interpretation authority which will provide the mappings. A range of these numbers should be reserved for private DOIs that are consumed solely on internal networks analogous to the 127 and 198 class A ip ranges for IPv4.

6. Terms and Definitions

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].

Domain of Interpretation: A Domain of Interpretation (DOI) represents a security boundary, where all systems within the DOI have semantically coherent labeling. That is, a security label must always mean exactly the same thing anywhere within the DOI.

Object: An object is a passive entity within the system that we wish to be protected. Objects can be entities such as files, directories, pipes, sockets, and many other system resources relevant to the protection of the system state.

Subject: A subject is an active entity usually a process which is requesting access to an object.

7. References

7.1. Normative References

[I-D.ietf-nfsv4-minorversion1]
Shepler, S., Eisler, M., and D. Noveck, "NFS Version 4 Minor Version 1", draft-ietf-nfsv4-minorversion1-29 (work in progress), December 2008.

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

7.2. Informational References

[BSDMAC] Rhodes, T., "FreeBSD Handbook: Chapter 16 Mandatory Access Control", <http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/mac.html>.

[DTOS] Smalley, S., "The Distributed Trusted Operating System (DTOS) Home Page", <http://www.cs.utah.edu/flux/fluke/html/dtos/HTML/dtos.html>.

[FLASK] Spencer, R., Smalley, S., Loscocco, P., Hibler, M., Andersen, D., and J. Lepreau, "The Flask Security Architecture: System Support for Diverse Security Policies", August 1999, <http://www.cs.utah.edu/flux/papers/flask-usenixsec99-abs.html>.

[FMAC] Sun Microsystems, "OpenSolaris Project: Flexible Mandatory Access Control", <http://www.opensolaris.org/os/project/fmac/>.

[SEBSD] SPARTA, "SEBSD: Port of SELinux FLASK and Type Enforcement to TrustedBSD", <http://www.trustedbsd.org/sebsd.html>.

[SEDARWIN]
SPARTA, "SEDarwin: Security Enhanced Darwin", <http://sedarwin.org/>.

[SELINUX] National Security Agency, "Security Enhanced Linux (SELinux)", <http://www.nsa.gov/selinux/>.

[SENFSV3] Carter, J., "Implementing SELinux Support for NFS", <http://www.nsa.gov/selinux/papers/nfsv3-abs.cfm>.

Authors' Addresses

David P. Quigley

Email: dpquigl@davequigley.com

James Morris
Red Hat, Inc.

Email: jmorris@namei.org