NFSv4 T. Haynes, Ed.
Internet-Draft NetApp
Intended status: Standards Track May 02, 2012
Expires: November 3, 2012

Requirements for Labeled NFS
draft-ietf-nfsv4-labreqs-01.txt

Abstract

This Internet-Draft outlines high-level requirements for the integration of flexible Mandatory Access Control (MAC) functionality into NFSv4.2. 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.

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 RFC 2119 [1].

Table of Contents

1. Introduction

Mandatory Access Control (MAC) systems have been mainstreamed in modern operating systems such as Linux (R), FreeBSD (R), and Solaris (TM). MAC systems bind security attributes to subjects (processes) and objects within a system. These attributes are used with other information in the system to make access control decisions.

Access control models such as Unix permissions or Access Control Lists are commonly referred to as Discretionary Access Control (DAC) models. These systems base their access decisions on user identity and resource ownership. In contrast MAC models base their access control decisions on the label on the subject (usually a process) and the object it wishes to access. These labels may contain user identity information but usually contain additional information. In DAC systems users are free to specify the access rules for resources that they own. MAC models base their security decisions on a system wide policy established by an administrator or organization which the users do not have the ability to override. DAC systems offer no real protection against malicious or flawed software due to each program running with the full permissions of the user executing it. Inversely MAC models can confine malicious or flawed software and usually act at a finer granularity than their DAC counterparts.

People desire to use NFSv4 with these systems. A mechanism is required to provide security attribute information to NFSv4 clients and servers. This mechanism has the following requirements:

(1) Clients must be able to convey to the server the security attribute of the subject making the access request. The server may provide a mechanism to enforce MAC policy based on the requesting subject's security attribute.

(2) Servers must be able to store and retrieve the security attribute of exported files as requested by the client.

(3) Servers must provide a mechanism for notifying clients of attribute changes of files on the server.

(4) Clients and Servers must be able to negotiate Label Formats and provide a mechanism to translate between them as needed.

2. Definitions

Label Format Specifier (LFS): is an identifier used by the client to establish the syntactic format of the security label and the semantic meaning of its components. These specifiers exist in a registry associated with documents describing the format and semantics of the label.

Label Format Registry: is the IANA registry (see [2]) containing all registered LFS along with references to the documents that describe the syntactic format and semantics of the security label.

Policy Identifier (PI): is an optional part of the definition of a Label Format Specifier which allows for clients and server to identify specific security policies.

Object: is a passive resource 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: is an active entity, usually a process, which is requesting access to an object.

MAC-Aware: is a server which can transmit and store object labels.

MAC-Functional: is a client or server which is LNFS enabled. Such a system can interpret labels and apply policies based on the security system.

Multi-Level Security (MLS): is a traditional model where objects are given a sensitivity level (Unclassified, Secret, Top Secret, etc) and a category set [5].

3. Requirements

The following initial requirements have been gathered from users, developers, and from previous development efforts in this area such as DTOS [6] and NSA's experimental NFSv3 enhancements [7].

3.1. Portability & Interoperability

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 NFSv4.2 [3] 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 NFSv4.2 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 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 NFSv4.2 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 Label Checking 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 LFS component provides a mechanism for identifying the structure and semantics of the opaque component. Meanwhile, the opaque component is the security label which will be interpreted by the MAC models.

MAC models base access decisions on security attributes bound to subjects and objects. With a given MAC model, all systems have semantically coherent labeling - a security label must always mean exactly the same thing on every system. 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 for the purposes of authorization, security service selection, and security label interpretation.

A negotiation scheme should be provided, allowing systems from different label formats to agree on how they will interpret or translate each others labels. Multiple concurrent 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.

If the LFS supported on 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 it does not recognize or cannot interpret, it must reject that label or state.

NFSv4.2 includes features which may cause a client to cross an LFS boundary when accessing what appears to be a single file system. If LFS 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 negotiation scheme with the primary aims of simplicity and completeness. This is to facilitate practical deployment of 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

The server and the client have mutually recognized 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 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 policies allow 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 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 its policy checks 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 is MAC-Functional enabled.

In the case of the server only being MAC-Functional, 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 being MAC-Functional, the client must operate as a standard NFSv4.2 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 NFSv4.2 over the wire, with the MAC-Functional system mapping the non-MAC-Functional system's processes or objects to security labels based on other characteristics in order to preserve its 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 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 but should not change the semantic meaning of the label.

3.7. Policy Enforcement

3.7.1. Full Mode

The server must enforce its 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-Functional and the client is not, the server must not accept security labels provided by the client, and only enforce its policy to exported objects. In the event that the client is MAC-Functional 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.

3.9. Upgrading Existing Server

Note that under the MAC model, all objects must have labels. Therefore, if an existing server is upgraded to include LNFS support, then it is the responsibility of the security system to define the behavior for existing objects. For example, if the security system is that the server just stores and returns labels, then existing files should return labels which are set to an empty value.

4. Use Cases

MAC labeling is meant to allow NFSv4.2 to be deployed in site configurable security schemes. The LFS and opaque data scheme allows for flexibility to meet these different implementations. In this section, we provide some examples of how NFSv4.2 could be deployed to meet existing needs. This is not an exhaustive listing.

4.1. Full MAC labeling support for remotely mounted filesystems

In this case, we assume a local networked environment where the servers and clients are under common administrative control. All systems in this network have the same MAC implementation and semantically identical MAC security labels for objects (i.e. labels mean the same thing on different systems, even if the policies on each system may differ to some extent). Clients will be able to apply fine-grained MAC policy to objects accessed via NFS mounts, and thus improve the overall consistency of MAC policy application within this environment.

An example of this case would be where user home directories are remotely mounted, and fine-grained MAC policy is implemented to protect, for example, private user data from being read by malicious web scripts running in the user's browser. With Labeled NFS, fine-grained MAC labeling of the user's files will allow the MAC policy to be implemented and provide the desired protection.

4.2. MAC labeling of virtual machine images stored on the network

Virtualization is now a commonly implemented feature of modern operating systems, and there is a need to ensure that MAC security policy is able to protect virtualized resources. A common implementation scheme involves storing virtualized guest filesystems on a networked server, which are then mounted remotely by guests upon instantiation. In this case, there is a need to ensure that the local guest kernel is able to access fine-grained MAC labels on the remotely mounted filesystem so that its MAC security policy can be applied.

4.3. International Traffic in Arms Regulations (ITAR)

The International Traffic in Arms Regulations (ITAR) is put forth by the United States Department of State, Directorate of Defense and Trade Controls. ITAR places strict requirements on the export and thus access of defense articles and defense services. Organizations that manage projects with articles and services deemed as within the scope of ITAR must ensure the regulations are met. The regulations require an assurance that ITAR information is accessed on a need-to-know basis, thus requiring strict, centrally managed access controls on items labeled as ITAR. Additionally, organizations must be able to prove that the controls were adequately maintained and that foreign nationals were not permitted access to these defense articles or service. ITAR control applicability may be dynamic; information may become subject to ITAR after creation (e.g., when the defense implications of technology are recognized).

4.4. Legal Hold/eDiscovery

Increased cases of legal holds on electronic sources of information (ESI) have resulted in organizations taking a pro-active approach to reduce the scope and thus costs associated with these activities. ESI Data Maps are increasing in use and require support in operating systems to strictly manage access controls in the case of a legal hold. The sizeable quantity of information involved in a legal discovery request may preclude making a copy of the information to a separate system that manages the legal hold on the copies; this results in a need to enforce the legal hold on the original information.

Organizations are taking steps to map out the sources of information that are most likely to be placed under a legal hold, these efforts result in ESI Data Maps. ESI Data Maps specify the Electronic Source of Information and the requirements for sensitivity and criticality. In the case of a legal hold, the ESI data map and labels can be used to ensure the legal hold is properly enforced on the predetermined set of information. An ESI data map narrows the scope of a legal hold to the predetermined ESI. The information must then be protected at a level of security of which the weight and admissibility of that evidence may be proved in a court of law. Current systems use application level controls and do not adequately meet the requirements. Labels may be used in advance when an ESI data map exercise is conducted with controls being applied at the time of a hold or labels may be applied to data sets during an eDiscovery exercise to ensure the data protections are adequate during the legal hold period.

Note that this use case requires multi-attribute labels, as both information sensitivity (e.g., to disclosure) and information criticality (e.g., to continued business operations) need to be captured.

4.5. Simple security label storage

In this case, a mixed and loosely administered network is assumed, where nodes may be running a variety of operating systems with different security mechanisms and security policies. It is desired that network file servers be simply capable of storing and retrieving MAC security labels for clients which use such labels. The LNFS protocol would be implemented here solely to enable transport of MAC security labels across the network. It should be noted that in such an environment, overall security cannot be as strongly enforced as when the server is also enforcing, and that this scheme is aimed at allowing MAC-capable clients to function with its MAC security policy enabled rather than perhaps disabling it entirely.

4.6. Diskless Linux

A number of popular operating system distributions depend on a mandatory access control (MAC) model to implement a kernel-enforced security policy. Typically, such models assign particular roles to individual processes, which limit or permit performing certain operations on a set of files, directories, sockets, or other objects. While the enforcing of the policy is typically a matter for the diskless NFS client itself, the filesystem objects in such models will typically carry MAC labels that are used to define policy on access. These policies may, for instance, describe privilege transitions that cannot be replicated using standard NFS ACL based

models.

For instance on a SYSV compatible system, if the 'init' process spawns a process that attempts to start the 'NetworkManager' executable, there may be a policy that sets up a role transition if the 'init' process and 'NetworkManager' file labels match a particular rule. Without this role transition, the process may find itself having insufficient privileges to perform its primary job of configuring network interfaces.

In setups of this type, a lot of the policy targets (such as sockets or privileged system calls) are entirely local to the client. The use of RPCSEC_GSSv3 ([4]) for enforcing compliance at the server level is therefore of limited value. The ability to permanently label files and have those labels read back by the client is, however, crucial to the ability to enforce that policy.

4.7. Multi-Level Security

In a MLS system objects are generally assigned a sensitivity level and a set of compartments. The sensitivity levels within the system are given an order ranging from lowest to highest classification level. Read access to an object is allowed when the sensitivity level of the subject "dominates" the object it wants to access. This means that the sensitivity level of the subject is higher than that of the object it wishes to access and that its set of compartments is a super-set of the compartments on the object.

The rest of the section will just use sensitivity levels. In general the example is a client that wishes to list the contents of a directory. The system defines the sensitivity levels as Unclassified (U), Secret (S), and Top Secret (TS). The directory to be searched is labeled Top Secret which means access to read the directory will only be granted if the subject making the request is also labeled Top Secret.

4.7.1. Full Mode - MAC-functional Client and Server

In the first part of this example a process on the client is running at the Secret level. The process issues a readdir() system call which enters the kernel. Before translating the readdir() system call into a request to the NFSv4.2 server the host operating system will consult the MAC module to see if the operation is allowed. Since the process is operating at Secret and the directory to be accessed is labeled Top Secret the MAC module will deny the request and an error code is returned to user space.

Consider a second case where instead of running at Secret the process

is running at Top Secret. In this case the sensitivity of the process is equal to or greater than that of the directory so the MAC module will allow the request. Now the readdir() is translated into the necessary NFSv4.2 call to the server. For the RPC request the client is using the proper credential to assert to the server that the process is running at Top Secret.

When the server receives the request it extracts the security label from the RPC session and retrieves the label on the directory. The server then checks with its MAC module if a Top Secret process is allowed to read the contents of the Top Secret directory. Since this is allowed by the policy then the server will return the appropriate information back to the client.

In this example the policy on the client and server were both the same. In the event that they were running different policies a translation of the labels might be needed. In this case it could be possible for a check to pass on the client and fail on the server. The server may consider additional information when making its policy decisions. For example the server could determine that a certain subnet is only cleared for data up to Secret classification. If that constraint was in place for the example above the client would still succeed, but the server would fail since the client is asserting a label that it is not able to use (Top Secret on a Secret network).

4.7.2. MAC-Functional Client

In these scenarios, the server is either non-MAC-Aware or MAC-Aware. The actions of the client will depend whether it is configured to treat the MAC-Aware server in the same manner as the non-MAC-Aware one. I.e., does it utilize the approach presented in Section 3.5.2 or does it allow the MAC-Aware server to return labels?

With a client that is MAC-Functional and using the example in the previous section, the result should be the same. The one difference is that all decisions are made on the client.

4.7.2.1. MAC-Aware Server

A process on the client labeled Secret wishes to access a directory labeled Top Secret on the server. This is denied since Secret does not dominate Top Secret. Note that there will be NFSv4.2 operations issued that return an object label for the client to process.

Note that in this scenario, all of the clients must be MAC-Functional. A single client which does not do its access control checks would violate the model.

4.7.2.2. Non-MAC-Aware Server

A process on the client labeled Secret wishes to access a directory which the client's policies label as Top Secret on the server. This is denied since Secret does not dominate Top Secret. Note that there will not be NFSv4.2 operations issued. If the process had instead a Top Secret process label, the client would issue NFSv4.2 operations to access the directory on the server.

4.7.3. MAC-Functional Server

With a MAC-Functional server and a client which is not, the client behaves as if it were in a normal NFSv4.2 environment. Since the process on the client does not provide a security attribute the server must define a mechanism for labeling all requests from a client. Assume that the server is using the same criteria used in the first example. The server sees the request as coming from a subnet that is a Secret network. The server determines that all clients on that subnet will have their requests labeled with Secret. Since the directory on the server is labeled Top Secret and Secret does not dominate Top Secret the server would fail the request with NFS4ERR_ACCESS.

5. Security Considerations

5.1. Guest Modes

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.

5.2. MAC-Functional Client Configuration

We defined a MAC model as a access control decision made on a system which normal users do not have the ability to override policies (see Section 1). If the process labels are created solely on the client, then if a malicious user has sufficient access on that client, the LNFS model is compromised. Note that this is no different from:

- current implementations in which the server uses policies to effectively determine the object label for requests from the client, or

- local decisions made on the client by the MAC security system.

The server must either explicitly trust the client (as in [7]) or the MAC model should enforce that users cannot override policies, perhaps via a externally managed source.

Once the labels leave the client, they can be protected by the transport mechanism as described in Section 3.3.

6. IANA Considerations

It is requested that IANA creates a registry of Label Formats to describe the syntactic format and semantics of the security label (see [2]).

7. References

7.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.

[2] Quigley, D. and J. Lu, "Registry Specification for MAC Security Label Formats", draft-quigley-label-format-registry (work in progress), 2011.

[3] Haynes, T., "NFS Version 4 Minor Version 2", draft-ietf-nfsv4-minorversion2-08 (Work In Progress), March 2011.

[4] Haynes, T. and N. Williams, "Remote Procedure Call (RPC) Security Version 3", draft-williams-rpcsecgssv3 (work in progress), 2011.

7.2. Informative References

[5] "Section 46.6. Multi-Level Security (MLS) of Deployment Guide: Deployment, configuration and administration of Red Hat Enterprise Linux 5, Edition 6", 2011.

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

[7] Carter, J., "Implementing SELinux Support for NFS", <http://www.nsa.gov/research/_files/selinux/papers/nfsv3.pdf>.

Appendix A. Acknowledgments

David Quigley was the early energy in motivating the entire LNFS effort.

James Morris, Jarrett Lu, and Stephen Smalley all were key contributors to both early versions of this document and to many conference calls.

Kathleen Moriarty provided the use cases for ITAR and Legal Hold/ eDiscovery.

Dan Walsh provided use cases for Secure Virtualization, Sandboxing, and NFS homedir labeling to handle process separation.

Trond Myklebust provided use cases for secure diskless NFS clients.

Both Nico Williams and Bryce Nordgren challenged assumptions during the review processes.

Appendix B. RFC Editor Notes

[RFC Editor: please remove this section prior to publishing this document as an RFC]

[RFC Editor: prior to publishing this document as an RFC, please replace all occurrences of RFCTBD10 with RFCxxxx where xxxx is the RFC number of this document]

Author's Address

Thomas Haynes (editor)
NetApp
9110 E 66th St
Tulsa, OK 74133
USA

Phone: +1 918 307 1415
Email: thomas@netapp.com