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 Support for NFSv4
draft-quigley-nfsv4-sec-label-03.txt

Abstract

This Internet-Draft describes additions to NFSv4 minor version one to support Mandatory Access Control systems. The current draft describes the mechanism for transporting a MAC security label using the NFSv4.1 protocol and the semantics required for that label. In addition to this it describes an example system of using this label in a fully MAC aware environment.

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 have been mainstreamed in modern operating systems such as Linux (R), FreeBSD (R), Solaris (TM), and Windows Vista (R). 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 refered 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 of 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:

- 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.

- Server must be able to store and retrieve the security attribute of exported files as requested by the client.

- Server must provide a mechanism for notifying clients of attribute changes of files on the server.

- Clients and Servers must be able to negotiate Label Formats and Domains of Interpretation (DOI) and provide a mechanism to translate between them as needed.

These four requirements are key to the system with only requirements two and three requiring changes to NFSv4.1. The ability to convey the security attribute of the subject as described in requirement one falls upon the RPC layer to implement. Requirement four allows

communication between different MAC implementations. The management of label formats, DOIs and the translation between them does not require any support from NFS on a protocol level and is out of the scope of this document.

2. 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].

Label Format Specifier: 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: The IANA registry containing all registered Label Format Specifiers along with references to the documents that describe the syntactic format and semantics of the security label.

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

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

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

3. MAC Security Attribute

MAC models base access decisions on security attributes bound to subjects and objects. This information can range from a user identity for an identity based MAC model, sensitivity levels for Multi-level security, or a type for Type Enforcement. These models base their decisions on different criteria but the semantics of the security attribute remain the same. The semantics required by the security attributes are listed below:

- Must provide flexibility with respect to MAC model.
- Must provide the ability to atomically set security information upon object creation
- Must provide the ability to enforce access control decisions both on the client and the server
- Must not expose an object to either the client or server name space before its security information has been bound to it.

NFSv4 provides several options for implementing the security attribute. The first option is to implement the security attribute as a named attribute. Named attributes provide flexibility since they are treated as an opaque field but lack a way to atomically set the attribute on creation. In addition, named attributes themselves are file system objects which need to be assigned a security attribute. This raises the question of how to assign security attributes to the file and directories used to hold the security attribute for the file in question. The inability to atomically assign the security attribute on file creation and the necessity to assign security attributes to its subcomponents makes named attributes unacceptable as a method for storing security attributes.

The second option is to implement the security attribute as a recommended attribute as defined in section 5.2 of RFC 3530. Recommended attributes have a fixed format and semantics, which conflicts with the flexible nature of the security attribute. To resolve this the security attribute consists of two components. The first component is a Label Format Specifier(LFS) as defined in [XXX: Insert Doc here] to allow for interoperability between MAC mechanisms. The second component is an opaque field which the actual security attribute data. To allow for various MAC models NFSv4 should be used solely as a transport mechanism for the security attribute. It is the responsibility of the endpoints to consume the security attribute and make access decisions based on their respective models. In addition, creation of objects through OPEN and CREATE allows for the security attribute to be specified upon creation. By providing an atomic create and set operation for the security attribute it is possible to enforce the second and fourth requirements. To this end the attribute number XXX:ATTRNUM is requested for the security_attribute recommended attribute.

3.1. Interpreting security_attribute

The security_attribute field contains two components the first component being an LFS. The LFS serves to provide the receiving end with the information necessary to translate the security attribute

into a form that is usable by the endpoint. Label Formats assigned an LFS may optionally choose to include a Policy Identifier (PI) field to allow for complex policy deployments. The Label Format Specifier and Label Format Registry are described in detail in [XXX: Insert RFC Here]. The translation used to interpret the security attribute is not specified as part of the protocol as it may depend on various factors. The second component is an opaque section which contains the data of the attribute. This component is dependent on the MAC model to interpret and enforce. The character '@' is reserved as a separator between the LFS and opaque section of the security attribute.

3.2. Delegations

In the event that a security attribute is changed on the server while a client holds a delegation on the file, the client should follow the existing protocol with respect to attribute changes. It should flush all changes back to the server and relinquish the delegation.

3.3. Permission Checking

It is not feasible to enumerate all possible MAC models and even levels of protection within a subset of these models. This means that the NFSv4 client and servers can not be expected to directly make access control decisions based on the security attribute. Instead NFSv4 should defer permission checking on this attribute to the host system. These checks are performed in addition to existing DAC and ACL checks outlined in the NFSv4 protocol. Section 4 gives a specific example of how the security attribute is handled under a particular MAC model.

3.4. Object Creation

When creating files in NFSv4 the OPEN and CREATE operations are used. One of the parameters to these operations is an fattr4 structure containing the attributes the file is to be created with. This allows NFSv4 to atomically set the security attribute of files upon creation. When a client is MAC aware it must always provide the initial security attribute upon file creation. In the event that the server is the only MAC aware entity in the system it should ignore the security attribute specified by the client and instead make the determination itself. A more in depth explanation can be found in Section 4.

4. MAC Security NFS Modes of Operation

A system using Labeled NFS may operate in three modes. The first

mode provides the most protection and is called "full" mode. In this mode both the client and server implement a MAC model allowing each end to make an access control decision. The remaining two modes are variations on each other and are called "smart client" and "smart server" modes. In these modes one end of the connection is not implementing a MAC model and because of this these operating modes offer less protection than full mode.

4.1. Full Mode

Full mode environments consist of MAC aware NFSv4 servers and clients and may be composed of mixed MAC models and policies. The system requires that both the client and server have an opportunity to perform an access control check based on all relevant information within the network. The file object security attribute is provided using the mechanism described in Section 3. The security attribute of the subject making the request is transported at the RPC layer using the mechanism described in XXX: Insert RFC for RPC layer label draft.

4.1.1. Initial Labeling and Translation

The ability to create a file is an action that a MAC model may wish to mediate. The client is given the responsibility to determine the initial security attribute to be placed on a file. This allows the client to make a decision as to the acceptable security attributes to create a file with before sending the request to the server. Once the server receives the creation request from the client it may choose to evaluate if the security attribute is acceptable.

Security attributes on the client and server may vary based on MAC model and policy. To handle this the security attribute field has an LFS component. This component is a mechanism for the host to identify the format and meaning of the opaque portion of the security attribute. A full mode environment may contain hosts operating in several different LFSs and DOIs. In this case a mechanism for translating the opaque portion of the security attribute is needed. The actual translation function will vary based on MAC model and policy and is out of the scope of this document. If a translation is unavailable for a given LFS and DOI then the request SHOULD be denied. Another recource is to allow the host to provide a fallback mapping for unknown security attributes.

4.1.2. Policy Enforcement

In full mode access control decisions are made by both the clients and servers. When a client makes a request it takes the security attribute from the requesting process and makes an access control

decision based on that attribute and the security attribute of the object it is trying to access. If the client denies that access an RPC call to the server is never made. If however the access is allowed the client will make a call to the NFS server.

When the server receives the request from the client it extracts the security attribute conveyed in the RPC request. The server then uses this security attribute and the attribute of the object the client is trying to access to make an access control decision. If the server's policy allows this access it will fulfill the client's request, otherwise it will return NFS4ERR_ACCESS

Implementations MAY validate security attributes supplied over the network to ensure that they are within a set of attributes permitted from a specific peer, and if not, reject them. Note that a system may permit a different set of attributes to be accepted from each peer. An example of this can be seen in Section 5.1.1.

4.2. Smart Client Mode

Smart client environments consist of NFSv4 servers that are not MAC aware but NFSv4 clients that are. Clients in this environment are may consist of groups implementing different MAC models policies. The system requires that all clients in the environment be responsible for access control checks. Due to the amount of trust placed in the clients this mode is only to be used in a trusted environment.

4.2.1. Initial Labeling and Translation

Just like in full mode the client is responsible for determining the initial label upon object creation. The server in smart client mode does not implement a MAC model, however, it may provide the ability to restrict the creation and labeling of object with certain labels based on different criteria as described in Section 4.1.2.

In a smart client environment a group of clients operate in a single DOI. This removes the need for the clients to maintain a set of DOI translations. Servers should provide a method to allow different groups of clients to access the server at the same time. However it should not let two groups of clients operating in different DOIs to access the same files.

4.2.2. Policy Enforcement

In smart client mode access control decisions are made by the clients. When a client accesses an object it obtains the security attribute of the object from the server and combines it with the

security attribute of the process making the request to make an access control decision. This check is in addition to the DAC checks provided by NFSv4 so this may fail based on the DAC criteria even if the MAC policy grants access. As the policy check is located on the client an access control denial should take the form that is native to the platform.

4.3. Smart Server Mode

Smart server environments consist of NFSv4 servers that are MAC aware and one or more MAC unaware clients. The server is the only entity enforcing 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). The level of trust and access extended to a client in this mode is configuration-specific.

4.3.1. Initial Labeling and Translation

In smart server mode all labeling and access control decisions are performed by the NFSv4 server. In this environment the NFSv4 clients are not MAC aware so they can not provide input into the access control decision. This requires the server to determine the initial labeling of objects. Normally the subject to use in this calculation would originate from the client. Instead the NFSv4 server may choose to assign the subject security attribute based on their authentication credentials and/or associated network attributes (e.g. IP address, network interface).

In smart server mode security attributes are contained solely within the NFSv4 server. This means that all security attributes used in the system remain within a single LFS and DOI. Since security attributes will not cross DOIs or change format there is no need to provide any translation functionality above that which is needed internally by the MAC model.

4.3.2. Policy Enforcement

All access control decisions in smart server mode are made by the server. The server will assign the subject a security attribute based on some criteria (e.g. IP address, network interface). Using the newly calculated security attribute and the security attribute of the object being requested the MAC model makes the access control check and returns NFS4ERR_ACCESS on a denial and NFS4_OK on success. This check is done transparently to the client so if the MAC permission check fails the client may be unaware of the reason for the permission failure. When operating in this mode administrators attempting to debug permission failures should be aware to check the

MAC policy running on the server in addition to the DAC settings.

5. Examples

The section below outlines an example of the Labeled NFS operation modes using a traditional Multi Level Security(MLS) model where objects are given a sensitivity level (Unclassified, Secret, Top Secret etc..) and a category set. This is just one example of a possible MAC model and is not required by labeled NFS implementations.

5.1. Multi-Level Security

In a Multi-level security (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 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.

5.1.1. Full Mode

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 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 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).

5.1.2. Smart Client Mode

In smart client mode the example is identical to the first part of a full mode operation. A process on the client labeled Secret wishes to access a Top Secret directory. As in the full mode example this is denied since Secret does not dominate Top Secret. If the process were operating at Top Secret it would pass the local access control check and the NFSv4 operation would proceed as in a normal NFSv4 environment.

5.1.3. Smart Server Mode

In a smart server mode the client behaves as if it were in a normal NFSv4 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 full mode 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.

6. Security Considerations

Depending on the level of protection the MAC system offers there may

be a requirement to tightly bind the security attribute to the data. It must be taken into consideration that when used in a pNFS environment is it possible that the security attribute and file data will be stored on separate servers.

7. IANA Considerations

XXX: Add section about LFS and LFS Registry referecing the other document.

8. Normative References

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

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

Authors' Addresses

David P. Quigley

Email: dpquigl@davequigley.com

James Morris
Red Hat, Inc.

Email: jmorris@namei.org