Network Configuration Protocol Access Control Model
draft-bierman-netconf-access-control-02

Abstract

The standardization of network configuration interfaces for use with
the NETCONF protocol requires a structured and secure operating
environment, which promotes human usability and multi-vendor
interoperability. There is a need for standard mechanisms to
restrict NETCONF protocol access for particular users to a pre-
configured subset of all available NETCONF operations and content.
This document discusses requirements for a suitable access control
model, and provides one solution which meets these requirements.

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 January 5, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
Provisions Relating to IETF Documents
(http://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect
Table of Contents

1.  Introduction .................................................. 5
  1.1.  Terminology .............................................. 5
        1.1.1.  Requirements Notation .......................... 5
        1.1.2.  NETCONF Terms .................................. 5
        1.1.3.  NACM Terms .................................... 6
  2.  Authentication Requirements .................................. 7
  3.  Access Control Requirements ................................... 8
      3.1.  Protocol Control Points ............................ 8
      3.2.  Simplicity .......................................... 9
      3.3.  Procedural Interface ................................ 9
      3.4.  Database Access ................................... 10
        3.4.1.  Access Rights .................................. 10
        3.4.2.  <get> and <get-config> Operations ............. 10
        3.4.3.  <edit-config> Operation ....................... 10
        3.4.4.  <copy-config> Operation ....................... 11
      3.5.  Users and Groups .................................. 12
      3.6.  Maintenance ........................................ 12
      3.7.  Configuration Capabilities ......................... 12
      3.8.  Identifying Security Holes ......................... 13
      3.9.  Data Shadowing ................................... 13
      3.10.  NETCONF Specific Requirements ................... 14
  4.  NETCONF Authentication and Authorization Model ................ 15
      4.1.  SSH Public Key Authentication ..................... 15
      4.2.  Local User Password Authentication ................ 16
        4.3.  RADIUS Password Authentication and Service
              Authorization ................................... 16
              4.3.1.  Operation ................................... 16
  5.  NETCONF Access Control Model (NACM) .......................... 18
      5.1.  Introduction ....................................... 18
        5.1.1.  Features ...................................... 18
        5.1.2.  External Dependencies .......................... 19
        5.1.3.  Message Processing Model ...................... 19
      5.2.  Model Components .................................. 21
        5.2.1.  Users ......................................... 21
        5.2.2.  Groups ........................................ 22
        5.2.3.  Sessions ...................................... 22
        5.2.4.  Access Permissions ............................. 22
        5.2.5.  Global Enforcement Controls .................. 23
        5.2.6.  Access Control Rules ........................... 23
      5.3.  Access Control Enforcement Procedures ................ 23
        5.3.1.  Initial Operation ................................ 24
        5.3.2.  Session Establishment ........................... 24
        5.3.3.  ‘access-denied’ Error Handling ................ 24
        5.3.4.  Incoming RPC Message Validation ................ 24
        5.3.5.  Data Node Access Validation .................... 27
        5.3.6.  Outgoing <rpc-reply> Authorization ............. 29
1. Introduction

The NETCONF protocol does not provide any standard mechanisms to restrict the operations and content that each user is authorized to use.

There is a need for inter-operable management of the controlled access to operator selected portions of the available NETCONF content within a particular server.

This document addresses NETCONF protocol authentication and access control mechanisms for the Operation and Content layers, as defined in [RFC4741], and [RFC5277]. It contains five main sections:

1. Authentication Requirements
2. Access Control Requirements
3. NETCONF Authentication and Authorization Model
4. NETCONF Access Control Model (NACM)
5. YANG Data Model (nacm.yang)

1.1. Terminology

1.1.1. Requirements Notation

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].

1.1.2. NETCONF Terms

The following terms are defined in RFC 4741 and are not redefined here:

- client
- operation
- RPC operation
- server
- session
1.1.3. NACM Terms

The following terms are used throughout this documentation:

access control: A security feature provided by the NETCONF server, which allows an operator to restrict access to a subset of all NETCONF protocol operations and data, based on various criteria.

access control model (ACM): A conceptual model used to configure and monitor the access control procedures desired by the operator to enforce a particular access control policy.

access control rule: The conceptual criteria used to determine if a particular NETCONF protocol operation should be permitted or denied.

authentication: The process of verifying a user’s identity.

superuser: The special administrative user account which is given unlimited NETCONF access, and is exempt from all access control enforcement.
2. Authentication Requirements

The authentication mechanism must support password authentication over RADIUS, to support deployment scenarios with centralized authentication servers. Additionally, local users must be supported, for scenarios when no centralized authentication server exists, or for situations where the centralized authentication server cannot be reached from the device.

Since the mandatory transport protocol for NETCONF is SSH NETCONF Over SSH [RFC4742], the authentication model must support SSH’s "publickey" and "password" authentication methods [RFC4252].

The model for authentication configuration should be flexible enough to support authentication methods defined by other standard documents or by vendors.
3. Access Control Requirements

3.1. Protocol Control Points

The NETCONF protocol allows new operations to be added at any time, and the YANG data modeling language supports this feature. It is not possible to design an ACM for NETCONF which only focuses on a static set of operations, like some other protocols. Since few assumptions can be made about an arbitrary protocol operation, the NETCONF architectural server components must be protected at several conceptual control points.

![Diagram of protocol control points](image)

Figure 1

The following access control points are defined:

RPC operation: Configurable permission to invoke specific RPC operations is required. Wildcard or multiple target mechanisms to reduce configuration and effort are also required.

NETCONF database: Configurable permission to read and/or alter specific data nodes within any conceptual database is required. Wildcard or multiple target mechanisms to reduce configuration and effort are also required.

RPC Reply Content: Configurable permission to read specific data nodes within any conceptual RPC output section is required. Unauthorized data is silently omitted from the reply, instead of dropping the reply or sending an 'access-denied' error.
Notification Content: Configurable permission to receive specific notification event types is required.

3.2. Simplicity

Experience has shown that a complicated ACM will not be widely deployed, because it is too hard to use. The key factor that is ignored in such solutions is the concept of ‘localized cost’. It should be easy to do simple things, and hard to do complex things, instead of hard to do everything.

Configuration of the access control system must be simple to use. Simple and common tasks should be easy to configure, and require little expertise or domain-specific knowledge. Complex tasks should be possible using additional mechanisms which may require additional expertise.

A single set of access control rules should be able to control all types of NETCONF RPC operation invocation, all conceptual database access, and all NETCONF session output.

Default access control policy needs to be as secure as possible.

Protocol access should be defined with a small and familiar set of permissions, while still allowing full control of NETCONF database access.

Access control does not need to be applied to NETCONF <hello> messages.

3.3. Procedural Interface

The NETCONF protocol uses a procedural interface model, and an extensible set of protocol operations. Access control for any possible protocol operation is required.

It must be possible to configure the ACM to permit or deny access to specific NETCONF operations.

YANG modules should be designed so that different access levels for input parameters to RPC operations is not required.

3.4. Database Access

It must be possible control access to specific nodes and sub-trees within the conceptual NETCONF database.

In order for a user to obtain access to a particular database node,
the user must be authorized to have the same requested access to the specified node, and all of its ancestors.

The same access control rules apply to all conceptual databases. For example, the candidate configuration or the running configuration.

Only the standard NETCONF databases (candidate, running, and startup) are controlled by the ACM. Local or remote files or databases accessed via the <url> parameter are optional to support.

The non-volatile startup configuration needs to be loaded into the running configuration without applying any access control rules.

Only a privileged user should be able to alter the factory-default access control rules.

3.4.1. Access Rights

A small set of hard-wired database access rights is needed to control access to all possible NETCONF database operations, including vendor extensions to the standard operation set.

The familiar ‘CRUDX’ model can support all NETCONF operations:

- Create: Allows the client to add a new data node instance to a database.
- Read: Allows the client to read a data node instance from a database, or receive the notification event type.
- Update: Allows the client to update an existing data node instance in a database.
- Delete: Allows the client to delete a data node instance from a database.
- eXec: Allows the client to execute the protocol operation.

3.4.2. <get> and <get-config> Operations

Read operations for restricted configuration data, either directly or via wildcard access, are silently omitted from the <rpc-reply> message.

3.4.3. <edit-config> Operation

The NACM access rights are not directly coupled to the NETCONF operation attribute, although they are similar. Instead, a NACM
access right applies to all operations which would result in a particular access operation to the target database. This section describes how these access rights apply to the specific database operations supported by the <edit-config> operation.

If the effective operation is 'none' (i.e., default-operation='none') for a particular data node, then no access control is applied to that data node.

A 'create', 'merge', or 'replace' operation on a database node which would result in the creation of a new data node instance, for which the user does not have 'create' access permission, is rejected with an 'access-denied' error.

A 'merge' or 'replace' operation on a database node which would result in the modification of an existing data node instance, for which the user does not have 'update' access permission, is rejected with an 'access-denied' error.

A 'replace' or 'delete' operation on a database node which would result in the deletion of an existing data node instance, for which the user does not have 'delete' access permission, is rejected with an 'access-denied' error.

A 'merge' operation may include data nodes which do not alter portions of the existing database. For example, a container or list nodes may be present for naming purposes, which do not actually alter the corresponding database node. These unaltered data nodes within the scope of a 'merge' operation are ignored by the server, and do not require any access rights by the client.

A 'merge' operation may include data nodes, but not include particular child data nodes that are present in the database. These missing data nodes within the scope of a 'merge' operation are ignored by the server, and do not require any access rights by the client.

The contents of specific restricted database nodes must not be exposed in any <rpc-error> elements within the reply.

3.4.4. <copy-config> Operation

Access control for the <copy-config> operation requires special consideration because the operator is replacing the entire target database. Write access to the entire database is needed for this operation to succeed.

A client must have access to every database node, even ones that are
not present in the source configuration data.

For example, consider a common use-case such as a simple backup and restore procedure. The operator must have full read access to the database in order to receive a complete copy of its contents. If not, the server will simply omit these sub-trees from the reply. If that copy is later used to restore the server database, the server will interpret the missing nodes as a request to delete those nodes, and return an error.

3.5. Users and Groups

The server must obtain a user name from the underlying NETCONF transport, such as an SSH user name.

It must be possible to specify access control rules for a single user or a configurable group of users.

A configurable superuser account is needed which bypasses all access control rules. This is needed in case the access control rules are mis-configured, and all access is denied.

The ACM must support the concept of administrative groups, to support the well-established distinction between a root account and other types of less-privileged conceptual user accounts. These groups must be configurable by the operator.

3.6. Maintenance

It should be possible to disable part or all of the access control model without deleting any configuration. By default, only the ‘superuser’ should be able to perform this task.

It should be possible to configure a ‘superuser’ account so that all access control is disabled for just this user. This allows the access control rules to always be modified without completely disabling access control for all users.

3.7. Configuration Capabilities

Suitable control and monitoring mechanisms are needed to allow an operator to easily manage all aspects of the ACM behavior. A standard data model, suitable for use with the <edit-config> operation must be available for this purpose.

Access control rules to restrict operations on specific sub-trees within the configuration database must be supported. Existing mechanisms should be used to identify the sub-tree(s) for this
3.8. Identifying Security Holes

One of the most important aspects of the data model documentation, and biggest concerns during deployment, is the identification of security-sensitive content. This applies to operations in NETCONF, not just data and notifications.

It is customary for security-sensitive objects to be documented in the Security Considerations section of an RFC. This is nice, but it is not good enough, for the following reasons:

- This documentation-only approach forces operators to study the RFC and determine if there are any potential security holes introduced by a new YANG module.
- If any security holes are identified, then the operator must study some more RFC text, and determine how to close the security hole(s).
- The ACM on each server must be configured to close the security holes, e.g., require privileged access to read or write the specific data identified in the Security Considerations section.
- If the ACM is not pre-configured, then there will be a time window of vulnerability, after the new module is loaded, and before the new access control rules for that module are configured, enabled, and debugged.

Often, the operator just wants to disable default access to the secure content, so no inadvertent or malicious changes can be made to the server. This allows the default rules to be more lenient, without significantly increasing the security risk.

A data model designer should be able to use machine-readable statements to identify NETCONF content which should be protected by default. This will allow client and server tools to automatically close data-model specific security holes, by denying access to sensitive data unless the user is explicitly authorized to perform the requested operation.

3.9. Data Shadowing

One of the more complicated security administration problems is identifying data nodes which shadow or mirror the content of another data node. An access control rule to prevent read operations for a particular node may be insufficient to prevent access to the data...
node with the copied value.

If the YANG leafref data type is used, then this data shadowing can be detected by applications (and the server stack), and prevented.

If the description statement, other documentation, or no documentation exists to identify a data shadow problem, then it may not be detected.

Since NETCONF allows any vendor operation to be added to the protocol, there is no way to reliably identify all of the operations that may expose copies of sensitive data nodes in <rpc-reply> messages.

A NETCONF server must insure than unauthorized access to its conceptual databases and non-configuration data nodes is prevented.

It is beyond the scope of this document to define access control enforcement procedures for underlying device instrumentation that may exist to support the NETCONF server operation. An operator must identify each operation that the server provides, and decide if it needs any access control applied to it.

Proprietary protocol operations should be properly documented by the vendor, so it is clear to operators what data nodes (if any) are affected by the operation, and what information (if any) is returned in the <rpc-reply> message.

3.10. NETCONF Specific Requirements

The server must be able to identify the specific protocol access request at the 4 access control points defined above.

The server must be able to identify any database access request, even for proprietary operations.

A session must always be authorized to invoke the <close-session> operation, defined in [RFC4741].

A session must always be authorized to receive the <replayComplete> and <notificationComplete> notification events, defined in [RFC5277]

The set of module name strings used within one particular server must be unique.

Within a single server, the module namespace URI associated with a specific module name string must persist across a reboot, and never change, once assigned.
4. NETCONF Authentication and Authorization Model

This document defines three authentication methods for use with NETCONF:

- publickey for local users over SSH
- password for local users over any transport
- password for RADIUS users over any transport

Additional methods may be defined by other standard documents or by vendors.

Conceptually, the NETCONF transport subsystem authenticates the user, and passes the name of the authenticated user to the NETCONF server. The NETCONF server authorizes the user by mapping it to one or more groups. Access to specific operations and content is then controlled by access control rules as described in Section 5.

Some protocols, such as RADIUS, performs both authentication and authorization, and has a mechanism to report authorization attributes to the client. These attributes are made available to the NETCONF server in an implementation specific manner.

This document defines two optional YANG features, 'local-users' and 'radius', which the server advertises to indicate support for configuring local users on the device, and for configuring RADIUS access, respectively.

4.1. SSH Public Key Authentication

If the NETCONF server advertises the 'local-users' feature, configuration of local users and their SSH public keys is supported in the /nacm/authentication/user list.

Public key authentication is requested by the SSH client. The SSH server looks up the user name provided by the client in the /nacm/authentication/user list, and verifies the key as described in [RFC4253].

If the 'local-users' feature is supported, then when a NETCONF client starts an SSH session towards the server, using the "publickey" authentication 'method name' [RFC4252], the SSH server looks up the user name given in the SSH authentication request in the /nacm/authentication/user list,
4.2. Local User Password Authentication

If the NETCONF server advertises the 'local-users' feature, configuration of local users and their passwords is supported in the /nacm/authentication/user list.

For NETCONF transport protocols that support password authentication, the leaf-list 'user-authentication-order' is used to control if local user password authentication should be used.

In SSH, password authentication is requested by the client. Other NETCONF transport protocols may also support password authentication.

When local user password authentication is requested, the NETCONF transport looks up the user name provided by the client in the /nacm/authentication/user list, and verifies the password.

4.3. RADIUS Password Authentication and Service Authorization

If the NETCONF server advertises the 'radius' feature, it supports user authentication and service authorization with RADIUS, as described in this section.

For NETCONF transport protocols that support password authentication, the leaf-list 'user-authentication-order' is used to control if RADIUS password authentication should be used.

In SSH, password authentication is requested by the client. Other NETCONF transport protocols may also support password authentication.

4.3.1. Operation

[Editor’s Note: I prefer to keep this section short, and just refer to the relevant rfcs which have detailed information on radius usage, instead of duplicating this info here...]

When RADIUS user authentication is requested, the NETCONF transport subsystem acts as a RADIUS client. In the Access-Request request [RFC2865], the following RADIUS attributes SHOULD be sent by the client [RFC5607]:

- Service-Type with the value Framed-Management
- Framed-Management-Protocol with the value NETCONF
- Management-Transport-Protocols with the value Integrity-Confidentiality-Protection
As described in RFC 5607, if an Access-Accept message is received which does not authorize the requested service, access MUST be denied.

If any Management-Policy-Id attributes are present in the Access-Accept message, they are treated as group names in the access control procedure, as described in Section 5.

The following RADIUS attributes MAY be sent by the RADIUS server:

- Session-Timeout
- Idle-Timeout

See [RFC2865] for a description of these attributes. These timeout values MUST be enforced by the NETCONF server.
5. NETCONF Access Control Model (NACM)

5.1. Introduction

This section provides a high-level overview of the access control model structure. It describes the NETCONF protocol message processing model, and the conceptual access control requirements within that model.

5.1.1. Features

The NACM data model provides the following features:

- Independent control of RPC, data, and notification access.
- Very simple access control rules configuration data model which is easy to use.
- The concept of a ‘superuser’ type of account is supported, but configuration such an account is beyond the scope of this document. The server must be able to determine if a superuser account is available, and if so, the actual user name for this account. A session associated with the superuser account will bypass all access control enforcement.
- A simple and familiar set of database permissions is used.
- Support for YANG security tagging (e.g., nacm:secure extension) allows default security modes to automatically exclude sensitive data.
- Separate default access modes for read, write, and execute permissions.
- Access control rules are applied to configurable groups of users.
- The entire ACM can be disabled during operation, in order to debug operational problems.
- Access control rules are simple to configure.
- The number of denied RPC operation requests and denied database write requests can be monitored by the client.
- Simple unconstrained YANG instance identifiers are used to configure access control rules for specific data nodes, or child nodes within specific RPC input, RPC output, and notification event type content.
5.1.2. External Dependencies

The NETCONF [RFC4741] protocol is used for all management purposes within this document. The server must support the features identified by the ‘NETCONF-base’ capability. It is expected that the mandatory transport mapping NETCONF Over SSH [RFC4742] is also supported by the server, and that the server has access to the user name associated with each session.

The YANG Data Modeling Language [I-D.ietf-netmod-yang] is used to define the NETCONF data models specified in this document. The YANG instance-identifier data type can be used to configure data-node-specific access control rules.

5.1.3. Message Processing Model

The following diagram shows the NETCONF message flow model, including the points at which access control is applied, during NETCONF message processing.
The following high-level sequence of conceptual processing steps is executed for each received <rpc> message, if access control enforcement is enabled:

- Access control is applied to all <rpc> messages (except <close-session>) received by the server, individually, for each active session, unless the user identity for the session is the 'superuser'.
If the session is authorized to execute the specified RPC
operation, then processing continues, otherwise the request is
rejected with an 'access-denied' error.

If the configuration database or conceptual state data is accessed
by the RPC operation, then the configuration access must be
authorized first. If the session is authorized to perform the
requested operation on the requested data, then processing
continues.

The following sequence of conceptual processing steps is executed for
each generated notification event, if access control enforcement is
enabled:

- Server instrumentation generates a conceptual notification, for a
  particular subscription.

- The notification access control enforcer checks the notification
event type, and if it is one which the session is not authorized
to read, then the notification is dropped for that subscription.

5.2. Model Components

This section defines the conceptual components related to access
control model.

5.2.1. Users

A ‘user’ is the conceptual identity, which is associated with the
access permissions granted to a particular session. A user is
identified by a string which must be unique within the server.

The user name string is usually derived from the transport layer
during session establishment. A server is required to have an
authenticated user name for a session before <rpc> requests will be
accepted. Otherwise all write requests must be rejected with an
‘access-denied’ error-tag value. If a read operation is not
authorized, then the requested data is silently dropped from the
reply.

The server MAY support a ‘superuser’ administrative user account,
which will bypass all access control enforcement. This is useful for
restricting initial access and repairing a broken access control
configuration. This account may be configurable to use a specific
user, or disabled completely. Some systems have factory-selected
superuser account names. There is no need to standardize the exact
user name for the superuser account. If no such account exists, then
all NETCONF access will be controlled by NACM.
5.2.2. Groups

Access to a specific NETCONF operation is granted to a session, associated with a group, not a user.

A group is identified by its name. All group names must be unique within the server.

A group member is identified by a user name string.

The same user may be configured in multiple groups.

The server should support the 3 default group identities defined in this document (admin, monitor, guest), however these roles are just unique identities, provided for operator convenience. There is no standard behavior defined for each group identity. That is up to the operator who configures the groups.

5.2.3. Sessions

A session is simply a NETCONF session, which is the entity which is granted access to specific NETCONF operations.

A session is associated with a single user name for the lifetime of the session.

5.2.4. Access Permissions

The access permissions are the NETCONF protocol specific set of permissions that have been assigned to a particular session role or group.

The same access permissions MUST stay in effect for the processing of a particular message.

The server MUST use the access control rules in effect at the time the message is processed.

The access control model treats RPC operation execution separately from configuration database access and outgoing messages:

create: Permission to create conceptual server data.

read: Read access to conceptual server data, <rpc-reply> and <notification> content.
update: Permission to modify existing conceptual server data.

delete: Permission to delete existing conceptual server data.

del: Permission to invoke an RPC operation.

5.2.5. Global Enforcement Controls

A global on/off switch is provided to enable or disable all access control enforcement.

An on/off switch is provided to enable or disable default access to invoke RPC operations.

An on/off switch is provided to enable or disable default permission to receive data in replies and notifications.

An on/off switch is provided to enable or disable default access to alter configuration data.

5.2.6. Access Control Rules

There are 4 types of rules available in NACM:

module rule: Controls access for definitions in a specific module, identified by its name.

RPC operation rule: Controls access for a specific RPC operation, identified by its module and name.

data node rule: Controls access for a specific data node, identified by its path location within the conceptual XML document for the data node.

notification rule: Controls access for a specific notification event type, identified by its module and name.

5.3. Access Control Enforcement Procedures

There are seven separate phases that must be addressed, four of which are related to the NETCONF message processing model. In addition, the initial start-up mode for a NETCONF server, session establishment, and 'access-denied' error handling procedures must also be considered.
5.3.1. Initial Operation

Upon the very first start-up of the NETCONF server, the access control configuration will probably not be present. If not, a server should not allow any write access to any session role except 'superuser' type of account in this state.

There is no requirement to enforce access control rules before or while the non-volatile configuration data is processed and loaded into the running configuration.

5.3.2. Session Establishment

The access control model applies specifically to the well-formed XML content transferred between a client and a server, after session establishment has been completed, and after the <hello> exchange has been successfully completed.

A server should not include any sensitive information in any <capability> elements within the <hello> exchange.

Once session establishment is completed, and a user identity has been authenticated, a NETCONF server will enforce the access control rules, based on the supplied user identity and the configuration data stored on the server.

5.3.3. ‘access-denied’ Error Handling

The ‘access-denied’ error-tag is generated when the access control system denies access to either a request to invoke an RPC operation or a request to perform a particular operation on the configuration database.

A server must not include any sensitive information in any <error-info> elements within the <rpc-error> response.

5.3.4. Incoming RPC Message Validation

The diagram below shows the basic conceptual structure of the access control processing model for incoming NETCONF <rpc> messages, within a server.
Access control begins with the message dispatcher. Only well-formed XML messages should be processed by the server.

After the server validates the <rpc> element, and determines the namespace URI and the element name of the RPC operation being requested, the RPC access control enforcer verifies that the session is authorized to invoke the RPC operation.

The RPC operation is authorized by following these steps:

1. If the <enable-nacm> parameter is set to 'false', then the RPC operation is permitted.
2. If the session is associated with the ‘superuser’ account, then the RPC operation is permitted.

3. If the requested operation is the NETCONF <close-session> operation, then the RPC operation is permitted.

4. Check all the <group> entries for ones that contain a <username> entry that matches the user name for the session making the request.

5. If no groups are found:
   * If the requested RPC operation is associated with a YANG module advertised in the server capabilities, and the rpc statement contains a nacm:secure or nacm:very-secure extension, then the RPC operation is denied.
   * If the <exec-default> parameter is set to ‘permit’, then permit the RPC operation, otherwise deny the request.

6. Check if there are any matching <rpc-rule> entries for the requested RPC operation. Any matching rules are processed in user-defined order, in case there are multiple <rpc-rule> entries for the requested RPC operation.

7. If an <rpc-rule> entry is found, then check the <allowed-rights> bits field for the entry, otherwise continue. The ‘exec’ bit MUST be present in the <allowed-rights> bits field for an <rpc-rule>, so it is not used in this procedure.

8. If the <rpc-rule> entry is considered a match, the the ‘nacm-action’ leaf is checked. If is equal to ‘permit’, then the RPC operation is permitted, otherwise it is denied.

9. Check if there are any matching <module-rule> entries for the same module as the requested RPC operation. Any matching rules are processed in user-defined order, in case there are multiple <module-rule> entries for the module containing the requested RPC operation.

10. If a <module-rule> entry is found, then check the <allowed-rights> bits field for the entry, otherwise continue. If the ‘exec’ bit is present in the <allowed-rights> bits field then the RPC rule is considered a match. otherwise it is not considered to match the request.

11. If the <module-rule> entry is considered a match, the the ‘nacm-action’ leaf is checked. If is equal to ‘permit’, then the RPC
operation is permitted, otherwise it is denied.

12. If the requested operation is identified as a nacm:secure or
nacm:very-secure RPC operation, then the RPC operation is
denied.

13. If the <exec-default> parameter is set to 'permit', then permit
the RPC operation, otherwise the RPC operation is denied.

If the session is not authorized to invoke the RPC operation then an
<rpc-error> is generated with the following information:

error-tag: access-denied

error-path: /rpc/method-QName, where 'method-QName' is a qualified
name identifying the actual RPC operation name. For example,
'/rpc/edit-config' represents the <edit-config> operation in the
NETCONF base namespace.

If the configuration database is accessed, either directly or as a
side effect of the RPC operation, then the server must intercept the
operation and make sure the session is authorized to perform the
requested operation on the specified data.

5.3.5. Data Node Access Validation

If a data node within a configuration database is accessed, or a
conceptual non-configuration node is accessed, then the server must
ensure that the client session is authorized to perform the requested
operation create, read, update, or delete operation on the specified
data node.

The data node access request is authorized by following these steps:

1. If the <enable-nacm> parameter is set to 'false', then the data
node access request is permitted.

2. If the session is associated with the 'superuser' account, then
the data node access request is permitted.

3. Check all the <group> entries for ones that contain a <user-
name> entry that matches the user name for the session making
the request.

4. If no groups are found:

   * If the requested data node is associated with a YANG module
     advertised in the server capabilities, and the data
definition statements (or any of its ancestors) contains a
nacm:secure or nacm:very-secure extension, then the data node
access request is denied.

* For a read request, if the <read-default> parameter is set to
  'permit', then permit the data node access request, otherwise
deny the request. For a read operation, this means that the
requested node is not included in the rpc-reply.

* For a write request, if the <write-default> parameter is set
to 'permit', then permit the data node access request,
otherwise deny the request.

5. Check if there are any matching <data-rule> entries for the
requested data node access request. Any matching rules are
processed in user-defined order, in case there are multiple
<data-rule> entries for the requested data node.

6. If an <data-rule> entry is found, then check the <allowed-
rights> bits field for the entry, otherwise continue.
   1. For a creation operation, if the 'create' bit is present in
      the <allowed-rights> bits field then the entry is considered
to be a match.
   2. For a read operation, if the 'read' bit is present in the
      <allowed-rights> bits field, then the entry is considered to
      be a match.
   3. For an update (e.g., 'merge' or 'replace') operation, if the
      'update' bit is present in the <allowed-rights> bits field
      then the entry is considered to be a match.
   4. For a deletion (e.g., 'delete') operation, if the 'delete'
      bit is present in the <allowed-rights> bits field then the
      entry is considered to be a match.

7. If the <data-rule> entry is considered a match, the the 'nacm-
action' leaf is checked. If it is equal to 'permit', then the
data operation is permitted, otherwise it is denied. For 'read'
operations, 'denied' means the requested data is not returned in
the reply.

8. Check if there are any matching <module-rule> entries for the
same module as the requested data node. Any matching rules are
processed in user-defined order, in case there are multiple
<module-rule> entries for the module containing the requested
data node.
9. If a <module-rule> entry is found, then check the <allowed-rights> bits field for the entry, otherwise continue.

1. For a creation operation, if the ‘create’ bit is present in the <allowed-rights> bits field then the entry is considered to be a match.

2. For a read operation, if the ‘read’ bit is present in the <allowed-rights> bits field, then the entry is considered to be a match.

3. For an update (e.g., ‘merge’ or ‘replace’) operation, if the ‘update’ bit is present in the <allowed-rights> bits field then the entry is considered to be a match.

4. For a deletion (e.g., ‘delete’) operation, if the ‘delete’ bit is present in the <allowed-rights> bits field then the entry is considered to be a match.

10. If the <module-rule> entry is considered a match, the the ‘nacm-action’ leaf is checked. If it is equal to ‘permit’, then the data operation is permitted, otherwise it is denied. For ‘read’ operations, ‘denied’ means the requested data is not returned in the reply.

11. For a read request, if the requested data node is identified as a nacm:very-secure definition, then the requested data node is not included in the reply.

12. For a write request, if the requested data node is identified as a nacm:secure or nacm:very-secure definition, then the data node access request is denied.

13. For a read request, if the <read-default> parameter is set to ‘permit’, then include the requested data in the reply, otherwise do not include the requested data in the reply.

14. For a write request, if the <write-default> parameter is set to ‘permit’, then permit the data node access request, otherwise deny the request.

5.3.6. Outgoing <rpc-reply> Authorization

The <rpc-reply> message should be checked by the server to make sure no unauthorized data is contained within it. If so, the restricted data must be removed from the message before it is sent to the client.
For RPC operations which do not access any data nodes, then any client authorized to invoke the RPC operation is also authorized to receive the <rpc-reply> for that RPC operation.

5.3.7. Outgoing <notification> Authorization

The <notification> message should be checked by the server to make sure no unauthorized data is contained within it. If so, the restricted data must be removed from the message before it is sent to the client.

Configuration of access control rules specifically for descendent nodes of the notification event type element are outside the scope of this document. If the session is authorized to receive the notification event type, then it is also authorized to receive any data it contains.

The following figure shows the conceptual message processing model for outgoing <notification> messages.
The generation of a notification event for a specific subscription is authorized by following these steps:

1. If the `<enable-nacm>` parameter is set to ‘false’, then the notification event is permitted.

2. If the session is associated with the ‘superuser’ account, then the notification event is permitted.

3. If the requested operation is the NETCONF `<replayComplete>` or `<notificationComplete>` event type, then the notification event is permitted.
4. Check all the <group> entries for ones that contain a <user-name> entry that matches the user name for the session that started the notification subscription.

5. If no groups are found:

   * If the requested notification is associated with a YANG module advertised in the server capabilities, and the notification statement contains a nacm:secure or nacm:very-secure extension, then the notification event is dropped for the associated subscription.

   * If the <read-default> parameter is set to ‘permit’, then permit the notification event, otherwise drop this event type for the associated subscription.

6. Check if there are any matching <notification-rule> entries for the specific notification event type being delivered to the subscription. Any matching rules are processed in user-defined order, in case there are multiple <notification-rule> entries for the requested notification event type.

7. If a <notification-rule> entry is found, then check the <allowed-rights> bits field for the entry, otherwise continue. If the ‘read’ bit is present in the <allowed-rights> bits field then the notification event type is permitted, otherwise it is dropped for the associated subscription.

8. Check if there are any matching <module-rule> entries for the same module as the notification event type. Any matching rules are processed in user-defined order, in case there are multiple <module-rule> entries for the module containing the notification event type.

9. If a <module-rule> entry is found, then check the <allowed-rights> bits field for the entry, otherwise continue. If the ‘read’ bit is present in the <allowed-rights> bits field then the notification event type is permitted, otherwise it is dropped for the associated subscription.

10. If the requested event type is identified as a nacm:very-secure notification definition, then the notification event type is denied.

11. If the <read-default> parameter is set to ‘permit’, then permit the notification event type, otherwise it is dropped for the associated subscription.
5.4. Data Model Definitions

This section defines the semantics of the conceptual data structures found in the data model in Section 5.4.

5.4.1. High Level Procedures

There are some high level management procedures that an administrator needs to consider before using this access control model:

1. Configure the global settings.
2. Configure one or more user groups.
3. Configure zero or more access control rules for specific modules.
4. Configure zero or more access control rules for specific RPC operations.
5. Configure zero or more access control rules for data node access.
6. Configure zero or more access control rules for notification event type access.

5.4.2. Data Organization

The top-level element is called <nacm>, and it is defined in the ‘nacm’ module namespace.

There are several data structures defined as child nodes of the <nacm> element:

leaf <enable-nacm>: On/off boolean switch to enable or disable access control enforcement.

container <authentication>: Configuration of the NETCONF server user authentication mechanisms.

leaf <read-default>: Enumeration to permit or deny default read access requests.

leaf <write-default>: Enumeration to permit or deny default write access requests.

leaf <exec-default>: Enumeration to permit or deny default RPC operation execution requests.
leaf <denied-rpcs>: Read-only counter of the number of times the server has denied an RPC operation request, since the last reboot of the server.

leaf <denied-data-writes>: Read-only counter of the number of times the server has denied a data node write request, since the last reboot of the server.

container <groups>: Configures the groups used within the access control system.

    list <group>: A list of user names belonging to the same administrative group.

container <rules>: Configures the access control rules used within the server.

    list <module-rule>: Configures the access control rules for a specific module.

    list <rpc-rule>: Configures the access control rules for RPC operation invocation.

    list <data-rule>: Configures the access control rules for configuration database access.

    list <notification-rule>: Configures the access control rules for controlling delivery of <notification> events.

5.4.3. YANG Module

The following YANG module is provided to specify the normative NETCONF content that must by supported by the server.

<CODE BEGINS> file="nacm@2010-06-29.yang"

module nacm {

    namespace "file://draft-bierman-netconf-access-control-02.txt";
    prefix "nacm";

    import ietf-yang-types {
        prefix yang;
    }

    import ietf-inet-types {
        prefix inet;
    }

    ...
extension secure {
  description
    "Used to indicate that the data model node
    represents a sensitive security system parameter.

    If present, the NETCONF server will only allow
    the designated ‘superuser’ to have write or execute
    default nacm-rights-type for the node. An explicit access
    control rule is required for all other users.

    The ‘secure’ extension may appear within a data, rpc,
    or notification node definition. It is ignored
    otherwise."
}

extension very-secure {
  description
    "Used to indicate that the data model node
    controls a very sensitive security system parameter.

    If present, the NETCONF server will only allow
    the designated ‘superuser’ to have read, write, or execute
    default nacm-rights-type for the node. An explicit access
    control rule is required for all other users.

    The ‘very-secure’ extension may appear within a data, rpc,
    or notification node definition. It is ignored
otherwise; 
}

/*
 * Features
 */

feature authentication {
    description
        "Indicates that the NETCONF server can be configured
to do authentication of users."
}

feature radius {
    if-feature authentication;
    description
        "Indicates that the NETCONF server can be configured
to act as a NAS and authenticate users
with RADIUS.";
    reference
        "RFC 2865: Remote Authentication Dial In User Service (RADIUS)
RFC 5607: Remote Authentication Dial-In User Service (RADIUS)
Authorization for Network Access Server (NAS)
Management";
}

feature local-users {
    if-feature authentication;
    description
        "Indicates that the NETCONF server supports
local user authentication.";
}

/*
 * Identities
 */

identity authentication-method {
    description
        "Base identity for NETCONF authentication methods."
}

identity radius {
    base authentication-method;
    description
        "Indicates NETCONF authentication using RADIUS.";
    reference
        "RFC 2865: Remote Authentication Dial In User Service (RADIUS)
RFC 5607: Remote Authentication Dial-In User Service (RADIUS)
Authorization for Network Access Server (NAS)
Management;

identity local-users {
    base authentication-method;
    description
        "Indicates password-based NETCONF authentication using locally
        configured users.";
}

/*
 * Derived types
 */
typedef nacm-user-name-type {
    type string {
        length "1..max";
    }
    description
        "General Purpose User Name string.";
}

typedef nacm-matchall-string-type {
    type string {
        pattern "\*";
    }
    description
        "The string containing a single asterisk '*' is used
        to conceptually represent all possible values
        for the particular leaf using this data type.";
}

typedef nacm-rights-type {
    type union {
        type nacm-matchall-string-type;

        type bits {
            bit create {
                description
                    "Create access allowed to all specified data.
                    Any protocol operation that creates a
                    new instance of the specified data is a create
                    operation.";
            }
            bit read {
                description
            }
            bit write {
                description
            }
            bit delete {
                description
            }
        }
    }
}
"Read access allowed to all specified data.
Any protocol operation or notification that
returns data to an application is a read
operation."
}

bit update {
  description
"Update access allowed to all specified data.
Any protocol operation that alters an existing
data node is an update operation."
}

bit delete {
  description
"Delete access allowed to all specified data.
Any protocol operation that removes a database
node instance is a delete operation."
}

bit exec {
  description
"Execution access to the specified RPC operation.
Any RPC operation invocation is an exec operation."
}

description
"NETCONF Access Rights.
The string ‘*’ indicates that all possible access
rights apply to the access rule. Otherwise, only
the specific access rights represented by the bit names
that are present apply to the access rule."

typedef nacm-group-name-type {
  type string {
    length "1..max";
    pattern "-\*[.]*";
  }
  description
"Name of administrative group that can be
assigned to the user, and specified in
an access control rule."
}

typedef nacm-action-type {
  type enumeration {
    enum permit {
      description
"Requested action is permitted."
    }
    enum deny {
      description
"Requested action is denied."
    }
    enum omit {
      description
"Requested action is not performed.
The action is not a delete operation."
    }
  }
}


enum deny {
    description
    "Requested action is denied.";
}

description
"Action taken by the server when a particular rule matches."

typedef schema-instance-identifier {
    type yang:xpath1.0;
    description
    "Path expression used to represent a special schema-instance identifier string.

    A schema-instance-identifier value is an unrestricted YANG instance-identifier expression. All the same rules as an instance-identifier apply except predicates for keys are optional. If a key predicate is missing, then the schema-instance-identifier represents all possible server instances for that key.

    This XPath expression is evaluated in the following context:
    
    o The set of namespace declarations are those in scope on the leaf element where this type is used.
    
    o The set of variable bindings contains one variable, 'USER', which contains the name of user of the current session.
    
    o The function library is the core function library, but note that due to the syntax restrictions of an instance-identifier, no functions are allowed.
    
    o The context node is the root node in the data tree."
}

typedef md5-crypt {
    type string {
        pattern "$0.* | $1[a-zA-Z0-9./]{2,8}$.*";
    }
    description
    "The md5-crypt type is used to store a password hash based on the MD5 message digest algorithm. When a clear text value is set to a leaf of this type, the server calculates a MD5 password hash,
and stores the result in the datastore. Thus, the password is never stored in clear text.

When a leaf of this type is read, the stored password hash is returned.

A value of this type matches one of the forms:

$0$<clear text password>
$1$<salt>$<password hash>

The '$0$' prefix signals that the value is clear text. When such a value is received by the server, an MD5 digest is calculated, and the string '$1$<salt>$' is prepended to the result, where <salt> is a random 2-8 characters long salt used to generate the digest. This value is stored in the configuration data store.

If a value starting with '$1$<salt>$' is received, the server knows that the value already represents an MD5 digest, and stores it as is in the data store.

When a server needs to verify a password given by a user, it finds the stored password hash string for that user, extracts the salt, and calculates the hash with the salt and given password as input. If the calculated hash value is the same as the stored value, the password given by the client is correct.

The digest algorithm is the md5 crypt function used for encrypting passwords for various UNIX systems.

reference
"RFC 1321: The MD5 Message-Digest Algorithm http://en.wikipedia.org/wiki/Crypt_(Unix)"
// FIXME: ref to wikipedia ok??

container nacm {
    nacm:very-secure;
    presence
        "An empty nacm container indicates that the NACM service is running, and using all default parameters."
    description
        "Parameters for NETCONF Access Control Model."
}
container authentication {
    nacm:very-secure;
    if-feature authentication;

description
    "The authentication configuration for the NETCONF server."

leaf-list user-authentication-order {
    type identityref {
        base authentication-method;
    }
    must '(. = "nacm:radius" and ../radius/server) or'
        + '(. != "nacm:radius")' {
        error-message
            "When 'radius' is used, a radius server must be configured.";
    }
    ordered-by user;

description
    "When the NETCONF server authenticates a user with a password, it tries the authentication methods in this leaf-list in order. If authentication with one method fails, the next method is used. If no method succeeds, the user is denied access.

    If the 'radius' feature is advertised by the NETCONF server, the 'radius' identity can be added to this list.

    If the 'local-users' feature is advertised by the NETCONF server, the 'local-users' identity can be added to this list."
}

container radius {
    if-feature radius;

description
    "The radius configuration for the NETCONF server."

list server {
    key address;

description
    "The radius server configuration used by the NETCONF server.";
leaf address {
    type inet:host;
    description
        "The address of the radius server."
}

leaf port {
    type inet:port-number;
    default "1812";
    description
        "The port number of the radius server."
}

leaf shared-secret {
    type string; // FIXME
    description
        "The shared secret which is known to both the RADIUS client and server."
    reference
        "RFC 2865: Remote Authentication Dial In User Service"
}

/*
We're using a special type aes-cfb-128-encrypted-string which works like the md5-crypt string, but encrypts the clear text value using a pre-provisioned password (not part of the config db!).

We use $0$ for cleartext and $4$ for the encrypted value. (we also have a des-version which uses $3$).

But I was thinking that maybe we could define a type for encrypted values without specifying the encryption algorithm, just specifying the format. $0$<clear text> | $x$<encrypted value>, and how it is encrypted is implementation specific.

One alternative is to store this shared secret in clear text. It is transmitted over a secure transport, and marked as very-secure. (The same argument could be made for user passwords, but these are personal and not even root should be able to read my passwd in clear text, so it makes more sense to keep them hidden.)
*/

description
    "The shared secret which is known to both the RADIUS client and server."
reference
    "RFC 2865: Remote Authentication Dial In User Service"
}

/*
How about configuration of number of retransmits and timeout?
*/

list user {
    if-feature local-users;
    key name;
description
"The list of local users configured on this device.";

leaf name {
  type nacm-user-name-type;
  description
    "The user name string identifying this entry.";
}
leaf password {
  type md5-crypt;
  description
    "The password for this entry.";
}
leaf ssh-dsa {
  type binary;
  description
    "The public DSA key for this entry.";
}
leaf ssh-rsa {
  type binary;
  description
    "The public RSA key for this entry.";
}

leaf enable-nacm {
  type boolean;
  default true;
  description
    "Enable or disable all NETCONF access control enforcement. If 'true', then enforcement is enabled. If 'false', then enforcement is disabled.";
}

leaf read-default {
  type nacm-action-type;
  default "permit";
  description
    "Controls whether read access is granted if no appropriate rule is found for a particular read request.";
}

leaf write-default {
  type nacm-action-type;
  default "deny";
description
 "Controls whether create, update, or delete access is granted if no appropriate rule is found for a particular write request.";
}

leaf exec-default {
  type nacm-action-type;
  default "permit";
  description
  "Controls whether exec access is granted if no appropriate rule is found for a particular RPC operation request.";
}

leaf denied-rpcs {
  type yang:zero-based-counter32;
  config false;
  mandatory true;
  description
  "Number of times an RPC operation request was denied since the server last restarted.";
}

leaf denied-data-writes {
  type yang:zero-based-counter32;
  config false;
  mandatory true;
  description
  "Number of times a request to alter a data node was denied, since the server last restarted.";
}

container groups {
  list group {
    key name;
    description
    "One NACM Group Entry.";

    leaf name {
      type nacm-group-name-type;
      description
      "Group name associated with this entry.";
    }

    leaf-list user-name {
      type nacm-user-name-type;
      description
    }
  }
}
"Each entry identifies the user name of a member of the group associated with this entry."
}
}
}

container rules {
  description
  "NETCONF Access Control Rules."
}

grouping common-rule-parms {
  leaf rule-name {
    type string {
      length "1..256";
    }
    description
    "Arbitrary name assigned to the access control rule."
  }
  leaf allowed-rights {
    type nacm-rights-type;
    description
    "List of access rights granted to specified administrative groups for the content specified by the associated path."
  }
  leaf-list allowed-group {
    type union {
      type nacm-matchall-string-type;
      type nacm-group-name-type;
    }
    min-elements 1;
    description
    "List of administrative groups which will be assigned the associated access rights for the content specified by the associated path. The string '*' indicates that all configured administrative groups apply to the entry."
  }
  leaf nacm-action {
    type nacm-action-type;
    mandatory true;
    description

"The access control action associated with the rule. If a rule is determined to match a particular request, then this object is used to determine whether to permit or deny the request."
}

leaf comment {
  type string {
    length "1..4095";
  }
  description
    "A textual description of the access rule.";
}
}

list module-rule {
  key "module-name rule-name";
  ordered-by user;
  description
    "One Module Access Rule.

    Rules are processed in user-defined order. A module rule is considered a match if the XML namespace for the specified module name matches the XML namespace used within a NETCONF PDU, and the administrative group associated with the requesting session is specified in the 'allowed-group' leaf-list, and the requested operation is included in the 'allowed-rights' leaf."

  leaf module-name {
    type string;
    description
      "Name of the module associated with this rule.";
  }

does common-rule-parms {
  refine allowed-rights {
    mandatory true;
  }
}
}

list rpc-rule {
  key "module-name rpc-name rule-name";
  ordered-by user;
description
"One RPC Operation Access Rule.

Rules are processed in user-defined order. An RPC rule is considered a match if the module name of the requested RPC operation matches ‘module-name’, the requested RPC operation matches ‘rpc-name’, and an administrative group associated with the session user is listed in the ‘allowed-group’ leaf-list. The ‘allowed-rights’ leaf is ignored by the server if it is present. Only the ‘exec’ bit can possibly cause a match for an RPC rule.

leaf module-name {
  type string;
  description
    "Name of the module defining this RPC operation.";
}

leaf rpc-name {
  type string;
  description
    "Name of the RPC operation.";
}

uses common-rule-parms;

list data-rule {
  key "rule-name";
  ordered-by user;
  description
    "One Data Access Control Rule.

Rules are processed in user-defined order. A data rule is considered to match when the path expression identifies the same node that is being accessed in the NETCONF database, and the administrative group associated with the session is identified in the ‘allowed-group’ leaf-list, and the requested operation is included in the ‘allowed-rights’ leaf."

leaf path {
  type schema-instance-identifier;
  mandatory true;
  description
    "Schema Instance Identifier associated with the data node
controlled by this rule.

Configuration data or state data instance identifiers start with a top-level data node. A complete instance identifier is required for this type of path value.

The special value ‘/’ refers to all possible database contents.

uses common-rule-parms {
  refine allowed-rights {
    mandatory true;
  }
}

list notification-rule {
  key "module-name
    notification-name
    rule-name";
  ordered-by user;

  description
  "One Notification Access Rule.

  A notification is considered a match if the module name of the requested event type matches ‘module-name’, the requested event type matches the ‘notification-name’, and the administrative group associated with the requesting session is listed in the ‘allowed-group’ leaf-list. If the ‘allowed-rights’ leaf is present, it is ignored by the server. Only the ‘read’ bit can possibly cause a match for a notification rule."

  leaf module-name {
    type string;
    description
    "Name of the module defining this notification event type.";
  }

  leaf notification-name {
    type string;
    description
    "Name of the notification event.";
  }
}
5.5. IANA Considerations

There are two actions that are requested of IANA:

1. register data model schema namespace URI (TBD)
2. register data model name (‘nacm’)

5.6. Security Considerations

This entire document discusses access control requirements and mechanisms for restricting NETCONF protocol behavior within a given session.

Configuration of the access control system is highly sensitive to system security. A server may choose not to allow any user configuration to some portions of it, such as the global security level, or the groups which allowed access to system resources.

This document incorporates the optional use of a ‘superuser’ account, which can be used to bypass access control enforcement. It is suggested that the ‘root’ account not be used for NETCONF over SSH servers, because ‘root’ SSH logins should be disabled in the SSH server.

If the server chooses to allow user configuration of the access control system, then only sessions using the ‘superuser’ administrative user should be allowed to have write access to the data model.

If the server chooses to allow user retrieval of the access control system configuration, then only sessions using the ‘superuser’ administrative user should be allowed to have read access to the data model.

There is a risk that invocation of non-standard RPC operations will have undocumented side effects. An administrator should construct
access control rules such that the configuration database is protected from such side effects. Also, such RPC operations should never be invoked by a session using the 'superuser' administrative user.

There is a risk that non-standard RPC operations, or even the standard <get> operation, may return data which 'aliases' or 'copies' sensitive data from a different data object. In this case, the namespace and/or the element name will not match the values for the sensitive data, which is then fully or partially copied into a different namespace and/or element. An administrator should avoid using data models which use this practice.

An administrator should restrict write access to all configurable objects within this data model. It is suggested that only sessions using the 'superuser' administrative role be permitted to configure the data model defined in this document.

If write access is allowed for configuration of access control rules, then care must be taken not to disrupt the access control enforcement.

An administrator should restrict read access to the following objects within this data model, which reveal access control configuration which could be considered sensitive.

- enable-nacm
- read-default
- write-default
- exec-default
- groups
- rules
6. Normative References


Appendix A. Usage Examples

The following XML snippets are provided as examples only, to demonstrate how NACM can be configured to perform some access control tasks.

A.1. <groups> Example

There must be at least one <group> entry in order for any of the access control rules to be useful.

The following XML shows arbitrary groups, and is not intended to represent any particular use-case.

```xml
<nacm xmlns="file://draft-bierman-netconf-access-control-02.txt">
  <groups>
    <group>
      <name>admin</name>
      <user-name>admin</user-name>
      <user-name>andy</user-name>
    </group>
    <group>
      <name>monitor</name>
      <user-name>wilma</user-name>
      <user-name>bam-bam</user-name>
    </group>
    <group>
      <name>guest</name>
      <user-name>guest</user-name>
      <user-name>guest@example.com</user-name>
    </group>
  </groups>
</nacm>
```

This example shows 3 groups:

1. The nacm:admin group contains 2 users named ‘admin’ and ‘andy’.
2. The nacm:monitor group contains 2 users named ‘wilma’ and ‘bam-bam’.
3. The nacm:guest group contains 2 users named ‘guest’ and ‘guest@example.com’.

A.2. <module-rule> Example

Module rules are used to control access to all the content defined in a specific module. These rules are checked after none of the specific rules (i.e., rpc-rule, data-rule, or notification-rule) matched the current access request.

```xml
<nacm xmlns="file://draft-bierman-netconf-access-control-02.txt">
  <rules>
    <module-rule>
      <module-name>ietf-netconf-monitoring</module-name>
      <rule-name>mod-1</rule-name>
      <allowed-rights>*/allowed-rights
      <allowed-group>guest</allowed-group>
      <nacm-action>deny</nacm-action>
      <comment>
        Do not allow guests any access to the netconf monitoring information.
      </comment>
    </module-rule>

    <module-rule>
      <module-name>ietf-netconf-monitoring</module-name>
      <rule-name>mod-2</rule-name>
      <allowed-rights>read</allowed-rights>
      <allowed-group>monitor</allowed-group>
      <nacm-action>permit</nacm-action>
      <comment>
        Allow the monitor group read access to the netconf monitoring information.
      </comment>
    </module-rule>

    <module-rule>
      <module-name>*</module-name>
      <rule-name>mod-3</rule-name>
      <allowed-rights>exec</allowed-rights>
      <allowed-group>monitor</allowed-group>
      <nacm-action>permit</nacm-action>
      <comment>
        Allow the monitor group to invoke any of the supported server operations.
      </comment>
    </module-rule>
  </rules>
</nacm>
```
<module-rule>
  <module-name>*</module-name>
  <rule-name>mod-4</rule-name>
  <allowed-rights>*</allowed-rights>
  <allowed-group>admin</allowed-group>
  <nacm-action>permit</nacm-action>
  <comment>
    Allow the admin group complete access to all
    operations and data.
  </comment>
</module-rule>

This example shows 4 module rules:

mod-1: This rule prevents the guest group from reading any
monitoring information in the ietf-netconf-monitoring YANG module.

mod-2: This rule allows the monitor group to read the ietf-netconf-
monitoring YANG module.

mod-3: This rule allows the monitor group to invoke any RPC
operation supported by the server.

mod-4: This rule allows the admin group complete access to all
content in the server. No subsequent rule will match for the
admin group, because of this module rule.

A.3. <rpc-rule> Example

RPC rules are used to control access to a specific RPC operation.
<nacm xmlns="file://draft-bierman-netconf-access-control-02.txt">
    <rules>
        <rpc-rule>
            <module-name>ietf-netconf</module-name>
            <rpc-name>kill-session</rpc-name>
            <rule-name>rpc-1</rule-name>
            <allowed-group>monitor</allowed-group>
            <allowed-group>guest</allowed-group>
            <nacm-action>deny</nacm-action>
            <comment>
                Do not allow the monitor or guest group
to kill another session.
            </comment>
        </rpc-rule>

        <rpc-rule>
            <module-name>ietf-netconf</module-name>
            <rpc-name>delete-config</rpc-name>
            <rule-name>rpc-2</rule-name>
            <allowed-group>monitor</allowed-group>
            <allowed-group>guest</allowed-group>
            <nacm-action>deny</nacm-action>
            <comment>
                Do not allow monitor or guest group
to delete any configurations.
            </comment>
        </rpc-rule>

        <rpc-rule>
            <module-name>ietf-netconf</module-name>
            <rpc-name>edit-config</rpc-name>
            <rule-name>rpc-3</rule-name>
            <allowed-group>monitor</allowed-group>
            <nacm-action>permit</nacm-action>
            <comment>
                Allow the monitor group to edit the configuration.
            </comment>
        </rpc-rule>
    </rules>
</nacm>

This example shows 3 RPC operation rules:
rpc-1: This rule prevents the monitor or guest groups from invoking the NETCONF <kill-session> RPC operation.

rpc-2: This rule prevents the monitor or guest groups from invoking the NETCONF <delete-config> RPC operation.

rpc-3: This rule allows the monitor group to invoke the NETCONF <edit-config> RPC operation. This rule will have no real affect unless the 'exec-default' leaf is set to 'deny'.

A.4. <data-rule> Example

Data rules are used to control access to specific (config and non-config) data nodes within the NETCONF content provided by the server.

```xml
<nacm xmlns="file://draft-bierman-netconf-access-control-02.txt">
  <rules>
    <data-rule>
      <rule-name>data-1</rule-name>
      <path>/nacm</path>
      <allowed-rights>*</allowed-rights>
      <allowed-group>guest</allowed-group>
      <nacm-action>deny</nacm-action>
      <comment>
        Deny the guest group any access to the /nacm data.
      </comment>
    </data-rule>
    <data-rule>
      <rule-name>data-acme-config</rule-name>
      <allowed-rights>read create update delete</allowed-rights>
      <allowed-group>monitor</allowed-group>
      <nacm-action>permit</nacm-action>
      <comment>
        Allow the monitor group complete access to the acme netconf configuration parameters. Showing long form of 'allowed-rights' instead of shorthand.
      </comment>
    </data-rule>
    <data-rule>
      <rule-name>dummy-itf</rule-name>
    </data-rule>
  </rules>
</nacm>
```
This example shows 4 data rules:

**data-1**: This rule denies the guest group any access to the `<nacm>` sub-tree. Note that the default namespace is only applicable because this sub-tree is defined in the same namespace as the `<data-rule>` element.

**data-acme-config**: This rule gives the monitor group read-write access to the `acme <config-parameters>`.

**dummy-itf**: This rule gives the monitor and guest groups read-update access to the `acme <interface>` entry named ‘dummy’. This entry cannot be created or deleted by these groups, just altered.
admin-itf: This rule gives the admin group read-write access to all acme <interface>. entries. This is an example of an unreachable rule because the ‘mod-3’ rule already gives the admin group full access to this data.

A.5. <notification-rule> Example

Notification rules are used to control access to a specific notification event type.

<nacm xmlns="file://draft-bierman-netconf-access-control-02.txt">
  <rules>
    <notification-rule>
      <module-name>acme-system</module-name>
      <notification-name>sys-config-change</notification-name>
      <rule-name>notif-1</rule-name>
      <allowed-group>monitor</allowed-group>
      <allowed-group>guest</allowed-group>
      <nacm-action>deny</nacm-action>
      <comment>
        Do not allow the guest or monitor groups to receive config change events.
      </comment>
    </notification-rule>
  </rules>
</nacm>

This example shows 1 notification rule:

notif-1: This rule prevents the monitor or guest groups from receiving the acme <sys-config-change> event type.
Appendix B. Open Issues

1. Do modules need to be identified by their XML namespace URI, or is the module name good enough?

2. Are any more wildcard mechanisms needed to specify the scope of an access control rule?

3. Should regular expressions (module='foo-*') be allowed in schema-instance-identifier strings?

4. Should XPath be allowed for specifying access control rules for data nodes?

5. Are any 'access-denied' notifications needed?

6. Should data rules support nodes that would not be eligible for retrieval with the <get> operation? If so, should schema nodes such as rpc 'input' or 'output' be in the path expression? How would notification content be identified?

7. Do any external access control models need to be supported somehow? For example, should the <groups> configuration be optionally read-only, so it can just mirror the internal (external or proprietary) group configuration?

8. Should the nacm:secure and nacm:very-secure extensions be optional to support, via a YANG feature?

9. Should the default access levels (e.g., read-default) be more restrictive by default? Should these defaults be a vendor decision? An operator decision? It is important that the server be able to install a factory default <nacm> container if needed.
Authors’ Addresses

Andy Bierman
InterWorking Labs
Scotts Valley, CA
USA

Email: andyb@iwl.com

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com