Network Working Group
Request for Comments: 5343
Updates: 3411
Category: Standards Track

J. Schoenwaelder
Jacobs University Bremen

September 2008

Simple Network Management Protocol (SNMP) Context EngineID Discovery

Status of This Memo

Abstract

The Simple Network Management Protocol (SNMP) version three (SNMPv3) requires that an application know the identifier (snmpEngineID) of the remote SNMP protocol engine in order to retrieve or manipulate objects maintained on the remote SNMP entity.

This document introduces a well-known localEngineID and a discovery mechanism that can be used to learn the snmpEngineID of a remote SNMP protocol engine. The proposed mechanism is independent of the features provided by SNMP security models and may also be used by other protocol interfaces providing access to managed objects.

This document updates RFC 3411.

Table of Contents

1. Introduction

To retrieve or manipulate management information using the third version of the Simple Network Management Protocol (SNMPv3) [RFC3410], it is necessary to know the identifier of the remote SNMP protocol engine, the so-called snmpEngineID [RFC3411]. While an appropriate snmpEngineID can in principle be configured on each management application for each SNMP agent, it is often desirable to discover the snmpEngineID automatically.

This document introduces a discovery mechanism that can be used to learn the snmpEngineID of a remote SNMP protocol engine. The proposed mechanism is independent of the features provided by SNMP security models. The mechanism has been designed to coexist with discovery mechanisms that may exist in SNMP security models, such as the authoritative engine identifier discovery of the User-based Security Model (USM) of SNMP [RFC3414].

This document updates RFC 3411 [RFC3411] by clarifying the IANA rules for the maintenance of the SnmpEngineID format registry.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Background

Within an administrative domain, an SNMP engine is uniquely identified by an snmpEngineID value [RFC3411]. An SNMP entity, which consists of an SNMP engine and several SNMP applications, may provide access to multiple contexts.

An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts [RFC3411]. A context is identified by the snmpEngineID value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a contextName).

To identify an individual item of management information within an administrative domain, a four tuple is used consisting of

1. a contextEngineID,

2. a contextName,

3. an object type, and

4. its instance identification.

The last two elements are encoded in an object identifier (OID) value. The contextName is a character string (following the SnmpAdminString textual convention of the SNMP-FRAMEWORK-MIB [RFC3411]) while the contextEngineID is an octet string constructed according to the rules defined as part of the SnmpEngineID textual convention of the SNMP-FRAMEWORK-MIB [RFC3411].

The SNMP protocol operations and the protocol data units (PDUs) operate on OIDs and thus deal with object types and instances [RFC3416]. The SNMP architecture [RFC3411] introduces the concept of a scopedPDU as a data structure containing a contextEngineID, a contextName, and a PDU. The SNMP version 3 (SNMPv3) message format uses ScopedPDUs to exchange management information [RFC3412].

Within the SNMP framework, contextEngineIDs serve as end-to-end identifiers. This becomes important in situations where SNMP proxies are deployed to translate between protocol versions or to cross middleboxes such as network address translators. In addition, snmpEngineIDs separate the identification of an SNMP engine from the transport addresses used to communicate with an SNMP engine. This property can be used to correlate management information easily, even in situations where multiple different transports were used to retrieve the information or where transport addresses can change dynamically.

To retrieve data from an SNMPv3 agent, it is necessary to know the appropriate contextEngineID. The User-based Security Model (USM) of SNMPv3 provides a mechanism to discover the snmpEngineID of the remote SNMP engine, since this is needed for security processing reasons. The discovered snmpEngineID can subsequently be used as a contextEngineID in a ScopedPDU to access management information local to the remote SNMP engine. Other security models, such as the Transport Security Model (TSM) [TSM], lack such a procedure and may use the discovery mechanism defined in this memo.

3. Procedure

The proposed discovery mechanism consists of two parts, namely (i) the definition of a special well-known snmpEngineID value, called the localEngineID, which always refers to a local default context, and (ii) the definition of a procedure to acquire the snmpEngineID scalar of the SNMP-FRAMEWORK-MIB [RFC3411] using the special well-known local localEngineID value.

3.1. Local EngineID

An SNMP command responder implementing this specification MUST register their pduTypes using the localEngineID snmpEngineID value (defined below) by invoking the registerContextEngineID() Abstract Service Interface (ASI) defined in RFC 3412 [RFC3412]. This registration is done in addition to the normal registration under the SNMP engine's snmpEngineID. This is consistent with the SNMPv3 specifications since they explicitly allow registration of multiple engineIDs and multiple pduTypes [RFC3412].

The SnmpEngineID textual convention [RFC3411] defines that an snmpEngineID value MUST be between 5 and 32 octets long. This specification proposes to use the variable length format 3) of the SnmpEngineID textual convention and to allocate the reserved, unused format value 6, using the enterprise ID 0 for the localEngineID. An ASN.1 definition for localEngineID would look like this:

```
localEngineID OCTET STRING ::= '8000000006'H
```

The localEngineID value always provides access to the default context of an SNMP engine. Note that the localEngineID value is intended to be used as a special value for the contextEngineID field in the ScopedPDU. It MUST NOT be used as a value to identify an SNMP engine; that is, this value MUST NOT be used in the snmpEngineID.0 scalar [RFC3418] or in the msgAuthoritativeEngineID field in the securityParameters of the User-based Security Model (USM) [RFC3414].

3.2. EngineID Discovery

Discovery of the snmpEngineID is done by sending a Read Class protocol operation (see Section 2.8 of [RFC3411]) to retrieve the snmpEngineID scalar using the localEngineID defined above as a contextEngineID value. Implementations SHOULD only perform this discovery step when it is needed. In particular, if security models are used that already discover the remote snmpEngineID (such as USM), then no further discovery is necessary. The same is true in situations where the application already knows a suitable snmpEngineID value.

The procedure to discover the snmpEngineID of a remote SNMP engine can be described as follows:

1. Check whether a suitable contextEngineID value is already known. If yes, use the provided contextEngineID value and stop the discovery procedure.

2. Check whether the selected security model supports discovery of the remote snmpEngineID (e.g., USM with its discovery mechanism). If yes, let the security model perform the discovery. If the remote snmpEngineID value has been successfully determined, assign it to the contextEngineID and stop the discovery procedure.

3. Send a Read Class operation to the remote SNMP engine using the localEngineID value as the contextEngineID in order to retrieve the scalar snmpEngineID.0 of the SNMP-FRAMEWORK-MIB [RFC3411]. If successful, set the contextEngineID to the retrieved value and stop the discovery procedure.

4. Return an error indication that a suitable contextEngineID could not be discovered.

The procedure outlined above is an example and can be modified to retrieve more variables in step 3, such as the sysObjectID.0 scalar or the snmpSetSerialNo.0 scalar of the SNMPv2-MIB [RFC3418].

4. IANA Considerations

RFC 3411 requested that IANA create a registry for SnmpEngineID formats. However, RFC 3411 did not ask IANA to record the initial assignments made by RFC 3411 nor did RFC 3411 spell out the precise allocation rules. To address this issue, the following rules are hereby established.

IANA maintains a registry for SnmpEngineID formats. The first four octets of an SnmpEngineID carry an enterprise number, while the fifth octet in a variable length SnmpEngineID value, called the format octet, indicates how the following octets are formed. The following format values were allocated in [RFC3411]:

Format	Description	References
0	reserved, unused	[RFC3411]
1	IPv4 address	[RFC3411]
2	IPv6 address	[RFC3411]
3	MAC address	[RFC3411]
4	administratively assigned text	[RFC3411]
5	administratively assigned octets	[RFC3411]
6-127	reserved, unused	[RFC3411]
128-255	enterprise specific	[RFC3411]

IANA can assign new format values out of the originally assigned and reserved number space 1-127. For new assignments in this number

space, a specification is required as per [RFC5226]. The number space 128-255 is enterprise specific and is not controlled by IANA.

Per this document, IANA has made the following assignment:

Format	Description	References
6	local engine	[RFC5343]

5. Security Considerations

SNMP version 3 (SNMPv3) provides cryptographic security to protect devices from unauthorized access. This specification recommends use of the security services provided by SNMPv3. In particular, it is RECOMMENDED to protect the discovery exchange.

An snmpEngineID can contain information such as a device's MAC address, IPv4 address, IPv6 address, or administratively assigned text. An attacker located behind a router / firewall / network address translator may not be able to obtain this information directly, and he therefore might discover snmpEngineID values in order to obtain this kind of device information.

In many environments, making snmpEngineID values accessible via a security level of noAuthNoPriv will benefit legitimate tools that try to algorithmically determine some basic information about a device. For this reason, the default View-based Access Control Model (VACM) configuration in Appendix A of RFC 3415 [RFC3415] gives noAuthNoPriv read access to the snmpEngineID. Furthermore, the USM discovery mechanism defined in RFC 3414 [RFC3414] uses unprotected messages and reveals snmpEngineID values.

In highly secure environments, snmpEngineID values can be protected by using the discovery mechanism described in this document together with a security model that does not exchange cleartext SNMP messages, such as the Transport Security Model (TSM) [TSM].

The isAccessAllowed() abstract service primitive of the SNMP access control subsystem does not take the contextEngineID into account when checking access rights [RFC3411]. As a consequence, it is not possible to define a special view for context engineID discovery. A request with a localEngineID is thus treated like a request with the correct snmpEngineID by the access control subsystem. This is inline with the SNMPv3 design where the authenticated identity is the securityName (together with the securityModel and securityLevel information), and transport addresses or knowledge of contextEngineID values do not impact the access-control decision.

6. Acknowledgments

Dave Perkins suggested the introduction of a "local" contextEngineID during the interim meeting of the ISMS (Integrated Security Model for SNMP) working group in Boston, 2006. Joe Fernandez, David Harrington, Dan Romascanu, and Bert Wijnen provided helpful review and feedback, which helped to improve this document.

7. References

7.1. Normative References

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

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

[RFC3412] Case, J., Harrington, D., Presuhn, R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3412, December 2002.

[RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.

[RFC3416] Presuhn, R., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, December 2002.

[RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

7.2. Informative References

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

[RFC3415] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3415, December 2002.

[TSM] Harrington, D., "Transport Security Model for SNMP", Work in Progress, July 2008.

Author's Address

Juergen Schoenwaelder
Jacobs University Bremen
Campus Ring 1
28725 Bremen
Germany

Phone: +49 421 200-3587
EMail: j.schoenwaelder@jacobs-university.de

Full Copyright Statement

Intellectual Property