Network Working Group
Internet-Draft
Expires: November 27, 2005

D. Harrington
Independent
J. Schoenwaelder
International University Bremen
May 26, 2005

Transport Mapping Security Model (TMSM) for the Simple Network Management Protocol version 3 (SNMPv3) draft-schoenw-snmp-tlsm-02.txt

Abstract

This document describes a Transport Mapping Security Model (TMSM) for the Simple Network Management Protocol (SNMP) architecture defined in RFC3411. At this stage, this document describes a framework, not a protocol. It does not provide a complete solution - it rather identifies and discusses some key aspects that need discussion and future work.

Table of Contents

1. Introduction

This document describes a Transport Mapping Security Model (TMSM) for the Simple Network Management Protocol (SNMP) architecture defined in RFC3411. At this stage, this document describes a framework, not a protocol. It does not provide a complete solution - it rather identifies and discusses some key aspects that need discussion and future work.

There are multiple ways to secure one's home or business, but they largely boil down to a continuum of alternatives. Let's consider three general approaches. In the first approach, an individual could buy a gun, learn to use it, and sit on your front porch waiting for intruders. In the second approach, one could hire an employee with a gun, schedule the employee, position the employee to guard what you want protected, hire a second guard to cover if the first gets sick, and so on. In the third approach, you could hire a security company, tell them what you want protected, and they could hire employees, train them, buy the guns, position the guards, schedule the guards, send a replacement when a guard cannot make it, etc., thus providing the security you want, with no significant effort on your part other than identifying requirements and verifying the quality of the service being provided.

The User-based Security Model (USM) as defined in [RFC3414] largely uses the first approach - it provides its own security. It utilizes existing mechanisms (MD5=the gun), but provides all the coordination. USM provides for the authentication of a principal, message encryption, data integrity checking, timeliness checking, etc.

USM was designed to be independent of other existing security infrastructures. USM therefore requires a separate user and key management infrastructure. Operators have reported that deploying another user and key management infrastructure in order to use SNMPv3 is a reason for not deploying SNMPv3 at this point in time. It is possible but difficult to define external mechanisms that handle the distribution of keys for use by the USM approach.

A solution based on the second approach might use a USM-compliant architecture, but replace the authentication mechanism with an external mechanism, such as RADIUS, to provide the authentication service. It might be possible to utilize an external protocol to encrypt a message, to check timeliness, to check data integrity, etc. It is difficult to cobble together a number of subcontracted services and coordinate them however, because it is difficult to build solid security bindings between the various services, and potential for gaps in the security is significant.

A solution based on the third approach might utilize one or more lower-layer security mechanisms to provide the message-oriented security services required. These would include authentication of the sender, encryption, timeliness checking, and data integrity checking. There are a number of IETF standards available or in development to address these problems at lower layers, frequently at the transport layer. A solution based on this approach might also utilize a "transport application" that is actually another application operating at the application layer, such as SSH [SSHauth]

This document proposes a Transport Mapping Security Model (TMSM), as an extension of the SNMPv3 architecture, that would allow security to be provided by an external protocol connected to the SNMP engine through an SNMP transport-mapping. Such a TMSM would then enable the use of existing security mechanisms such as (TLS) [RFC2246], Kerberos [RFC1510] or SASL [RFC2222] within the SNMPv3 architecture.

As pointed out in the EUSM proposal [EUSM], it is desirable to use mechanisms that could "unify the approach for administrative security for SNMPv3 and CLI" and other management interfaces. The use of security services provided by lower layers or other applications is the approach commonly used for the CLI, and is the approach being proposed for NETCONF

This document describes the motivation for leveraging transport layer security mechanisms for secure SNMP communication, identifies some key issues and provides some proposals for design choices that may be made to provide a workable solution that meets operational requirements and fits into the SNMP architecture defined in [RFC3411]

2. Motivation

There are a number of Internet security protocols and mechanisms that are in wide spread use. Many of them try to provide a generic infrastructure to be used by many different application layer protocols. The motivation behind TMSM is to leverage these protocols where it seems useful.

There are a number of challenges to be addressed to map the security provided by a secure transport into the SNMP architecture so that SNMP continues to work without any surprises. These are discussed in detail below.

Some points requiring further WG research and discussion are identified by [todo] markers in the text.

3. Requirements of a Transport Mapping Security Model

3.1 Security Requirements

Transport mapping security protocols SHOULD ideally provide the protection against the following message-oriented threats [RFC3411]:

1. modification of information
2. masquerade
3. message stream modification
4. disclosure

According to [RFC3411], it is not required to protect against denial of service or traffic analysis.

3.1.1 Security Protocol Requirements

There are a number of standard protocols that could be proposed as possible solutions within the TMSM framework. Some factors should be considered when selecting a protocol for use within this framework.

Using a protocol in a manner for which it was not designed has numerous problems. The advertised security characteristics of a protocol may depend on its being used as designed; when used in other ways, it may not deliver the expected security characteristics. It is recommended that any proposed model include a discussion of the applicability statement of the protocols to be used.

A protocol used for the TMSM framework should ideally require no modifications to the protocol. Modifying the protocol may change its security characteristics in ways that would impact other existing usages. If a change is necessary, the change should be an extension that has no impact on the existing usages. It is recommended that any proposed model include a discussion of potential impact on other usages of the protocol.

It has been a long-standing requirement that SNMP be able to work when the network is unstable, to enable network troubleshooting and repair. The UDP approach has been considered to meet that need well, with an assumption that getting small messages through, even if out of order, is better than gettting no messages through. There has been a long debate about whether UDP actually offers better support than TCP when the underlying IP or lower layers are unstable. There has been recent discussion of whether operators actually use SNMP to troubleshoot and repair unstable networks.

There has been discussion of ways SNMP could be extended to better support management/monitoring needs when a network is running just

fine. Use of a TCP transport, for example, could enable larger message sizes and more efficient table retrievals.

TMSM models MUST be able to coexist with other protocol models, and may be designed to utilize either TCP or UDP, depending on the transport.

3.1.2 Session Requirements

Sessions are not part of RFC3411 architecture, but are considered desirable because the cost of authentication can be amortized over potentially many transactions.

For transports that utilize sessions, a session should have a single user and security level associated with it. If an exchange between communicating engines would require a different security level or would be on behalf of a different user, then another session would be needed. An immediate consequence of this is that implementations should be able to maintain some reasonable number of concurrent sessions.

3.2 Architectural Modularity Requirements

[RFC3411] section 3 describes a modular architecture to allow the evolution of the SNMP protocol standards over time, and to minimize side effects between subsystems when changes are made. This architecture includes a Security Subsystem which is responsible for realizing security services.

In SNMPv2, there were many problems of side effects between subsystems caused by the manipulation of MIB objects, especially those related to authentication and authorization, because many of the parameters were stored in shared MIB objects, and different models and protocols could assign different values to the objects. Contributors assumed slightly different shades of meaning depending on the models and protocols being used. As the shared MIB module design was modified to accommodate a specific model, other models which used the same MIB objects were broken.

ASIs were developed to pass model-independent parameters. The models were required to translate from their model-dependent formats into a model-independent format, defined using model-independent semantics, which would not impact other models.

Parameters have been provided in the ASIs to pass model-independent information about the authentication that has been provided. These parameters include a model-independent identifier of the security "principal", the security model used to perform the authentication,

and which SNMP-specific security features were applied to the message (authentication and/or privacy).

The design of a transport mapping security model must abide the goals of the RFC3411 architecture. To that end, this transport mapping security model proposal focuses on a modular subsystem that can be advanced through the standards process independently of other proposals, and independent of other subsystems as much as possible.

There has been some discussion of maintaining multiple tunnels or sessions for different security levels or for different applications.The ability to have an application select different sessions or connections on a per-message basis would likely require a modification to the SNMP architecture to provide new ASIs, which is out of scope for this document.

IETF standards typically require one mandatory-to-implement solution, with the capability of adding new security mechanisms in the future. Any transport mapping security model should define one minimum-compliance mechanism, preferably one which is already widely deployed within the transport layer security protocol used.

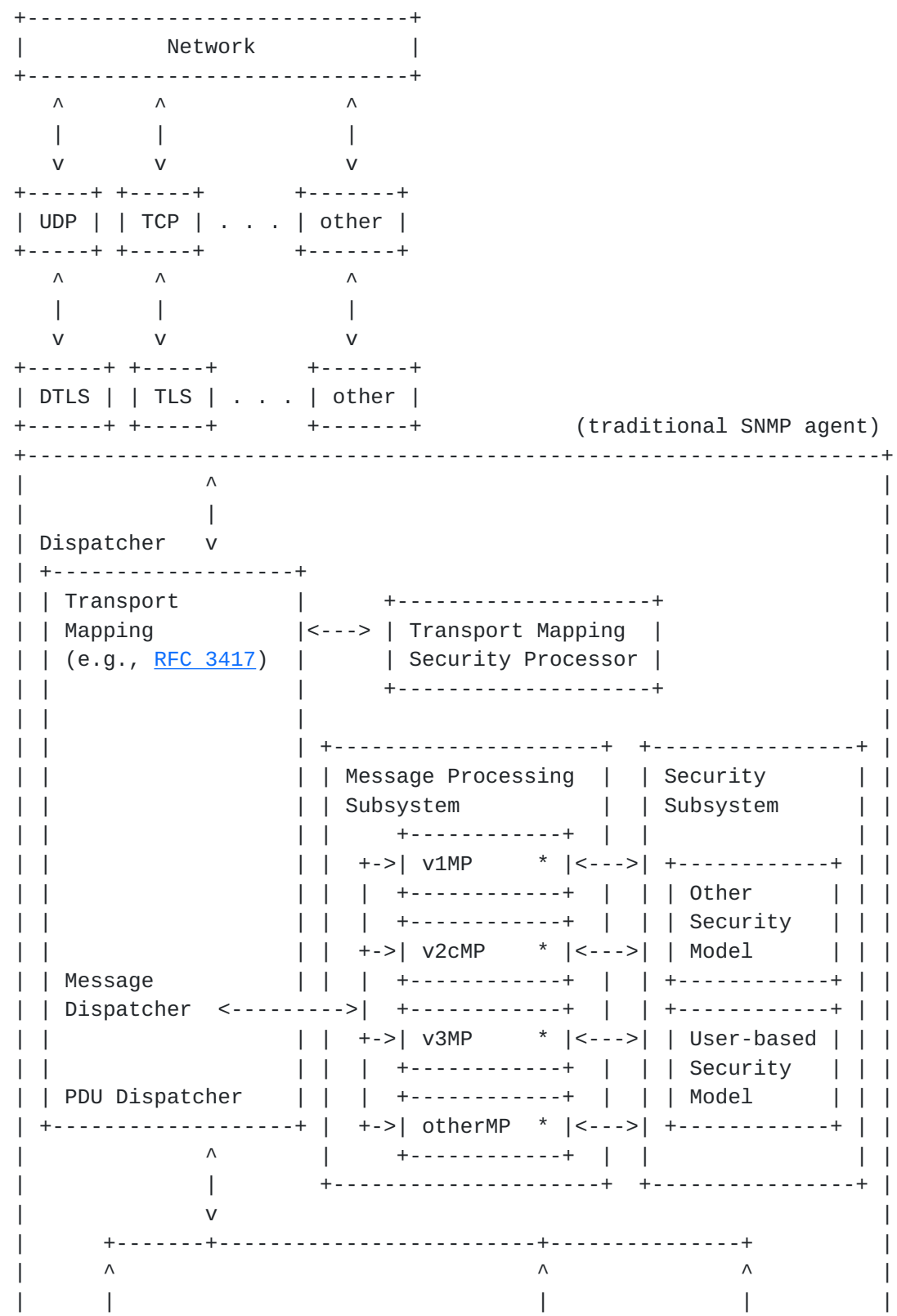
The TMSM subsystem is designed as an architectural extension that permits additional transport security protocols to be "plugged into" the RFC3411 architecture, supported by corresponding transport-security-aware transport mapping models.

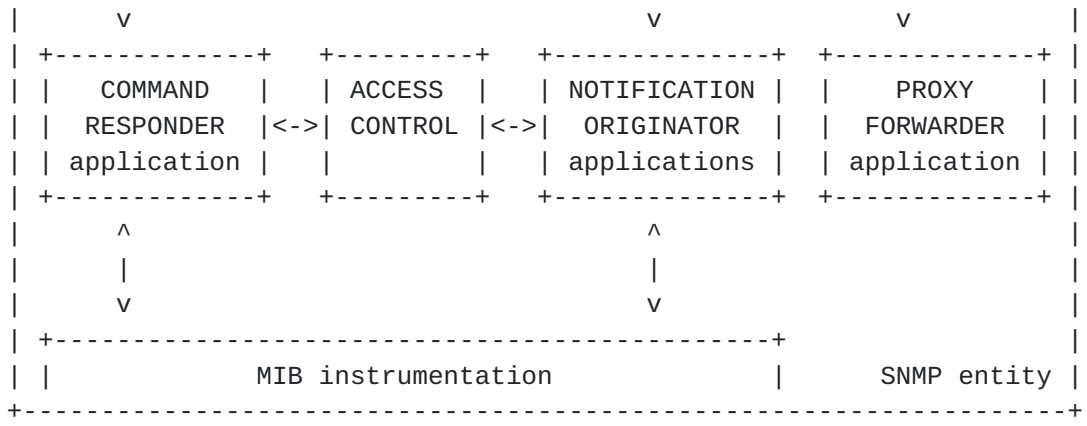
The RFC3411 architecture, and the USM approach, assume that a security model is called by a message-processing model and will perform multiple security functions. The TMSM approach performs similar functions but performs them in different places within the archtitecture, so we need to distinguish the two locations for security processing.

Transport mapping security is by its very nature a security layer which is plugged into the RFC3411 architecture between the transport layer and the message dispatcher. Conceptually, transport mapping security processing will be called from within the Transport Mapping functionality of an SNMP engine dispatcher to perform the translation of transport security parameters to/from security-model-independent parameters. This transport mapping security processor will be referred to in this document as TMSP.

Additional functionality may be performed as part of the message processing function, i.e. in the security subsystem of the RFC3411 architecture. This document will refer to message processor's security processor as the MPSP.

Thus a TMSM is composed of both a TPSP and an MPSP.



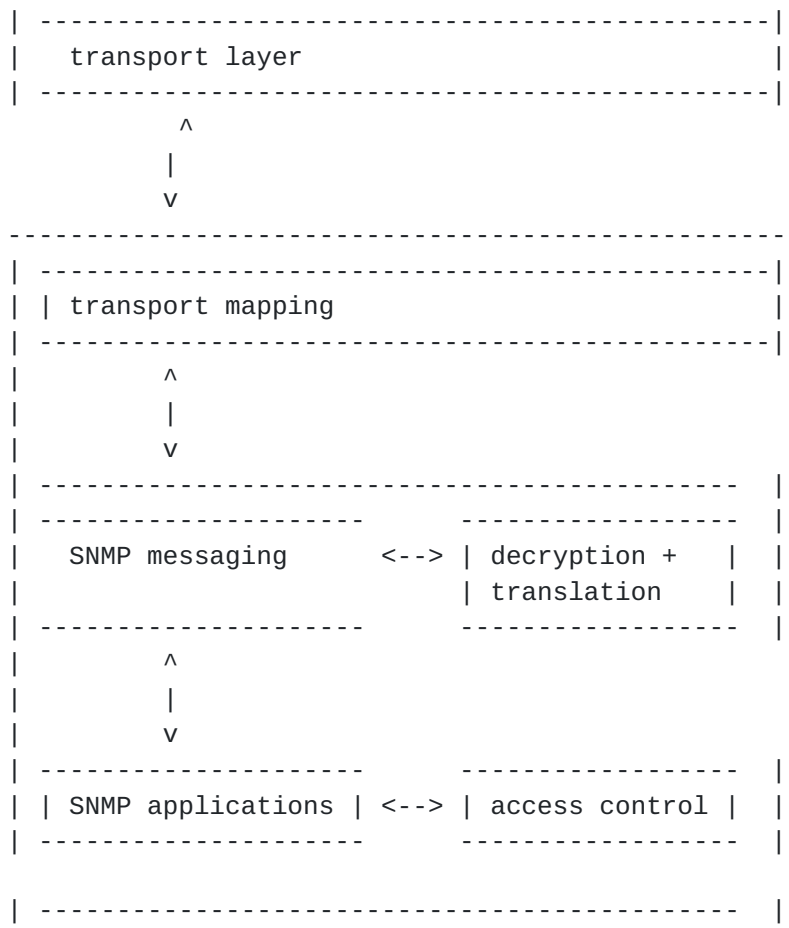


3.2.1 USM and the RFC3411 Architecture

This following diagrams illustrate the difference in the security processing done by the USM model and the security processing done by a TMSM model.

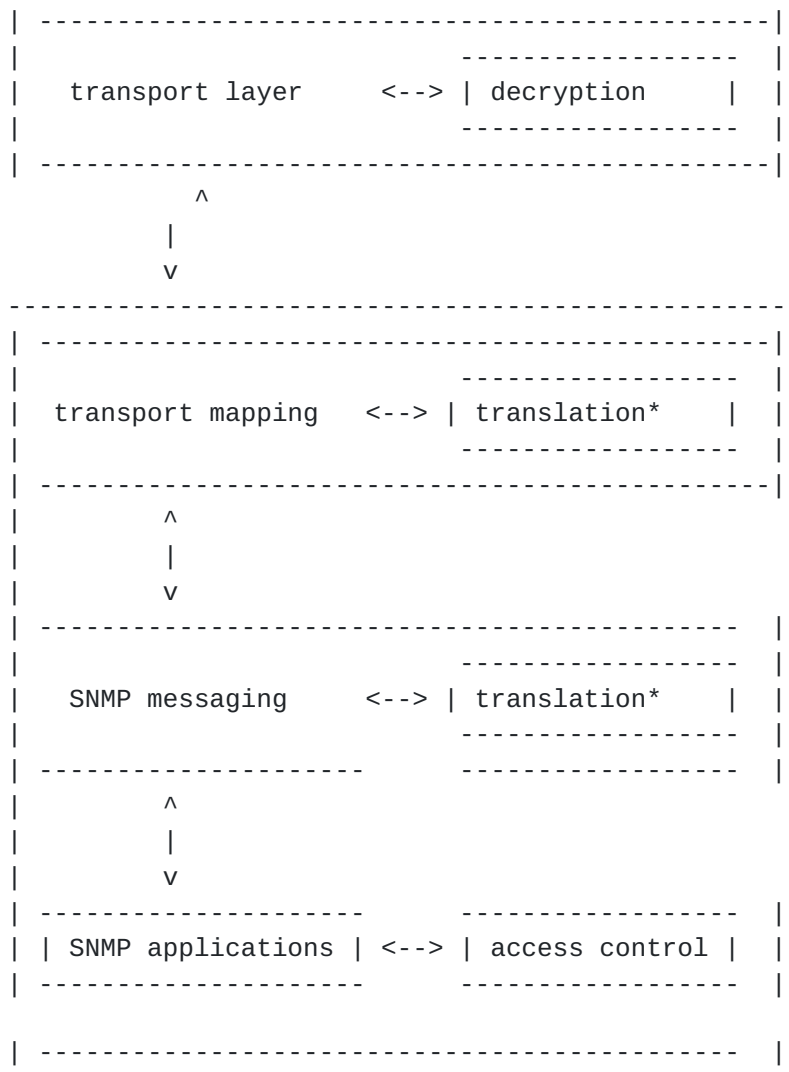
The USM security model is encapsulated by the messaging model, because the messaging model needs to (for incoming messages) 1) decode the ASN.1 (messaging model) 2) determine the SNMP security model and parameters (messaging model) 3) decrypt the encrypted portions of the message (security model) 4) translate parameters to model-independent parameters (security model) 5) determine which application should get the decrypted portions (messaging model), and 6) pass on the decrypted portions with model-independent parameters.

The USM approach uses SNMP-specific message security and parameters.



3.2.2 TMSM and the RFC3411 Architecture

In the TMSM approach, the order of the steps differ and may be handled by different subsystems: 1) decrypt the encrypted portions of the message (transport layer) 2) determine the SNMP security model and parameters (transport mapping) 3*) translate parameters to model-independent parameters (transport mapping) 4) decode the ASN.1 (messaging model) 5) determine which application should get the decrypted portions (messaging model) 6*) translate parameters to model-independent parameters (security model) 7) pass on the decrypted portions with model-independent security parameters This is largely based on having non-SNMP-specific message security and parameters. The transport mapping model might provide the translation from (e.g.) TLS user to securityName in step 3, OR The TLS user might be passed to the messaging model to pass to a TMSM security model to do the translation in step 6, if the WG decides all translations should use the same translation table (e.g. the USM MIB).



3.3 Passing Messages between Subsystems

RFC3411 defines ASIs that describe the passing of messages between subsystem within an engine, and the parameters which are expected to be passed between the subsystems. The ASIs generally pass model-independent information.

A TMSM model will establish an encrypted tunnel between the transport mappings of two SNMP engines. One transport mapping security model instance encrypts all messages, and the other transport mapping security model instance decrypts the messages.

After the transport layer tunnel is established, then SNMP messages can conceptually be sent through the tunnel from one SNMP message

dispatcher to another SNMP message dispatcher. Once the tunnel is established, multiple SNMP messages may be able to be passed through the same tunnel.

Within an engine, outgoing SNMP messages are passed unencrypted from the message dispatcher to the transport mapping, and incoming messages are passed unencrypted from the transport mapping to the message dispatcher.

3.4 Security Parameter Passing Requirement

[RFC3411] section 4 describes primitives to describe the abstract service interfaces used to conceptually pass information between the various subsystems, models and applications within the architecture.

The security parameters include a model-independent identifier of the security "principal", the security model used to perform the authentication, and which SNMP-specific security services were (should be) applied to the message (authentication and/or privacy).

In the RFC3411 architecture, the messaging model must unpack SNMP-specific security parameters from the message before calling a security model to authenticate and decrypt an incoming message, perform integrity checking, and translate model-specific security parameters into model-independent parameters. In the TMSM approach, the security -model specific parameters are not all carried in the SNMP message, and can be determined from the transport layer by the transport mapping, before the message processing begins.

[todo] For outgoing messages, it is necessary to have an MPSP because it is the MPSP that actually creates the message from it scomponent parts. Does the MPSP need to know the transport address or the actual transport security capabilities, or can this be handled in the TMSP, given the model-independent (and message-version-independent) parameters? Are there any security services provided by the MPSP for an outgoing message?

[todo] For incoming messages, is there security functionality that can only be handled after the message version is known, such as the comparison of transport security capabilities and msgFlags? Does that functionality need to know the transport address and session or just the model-independent security parameters (securityName, model, level)? Are there any SNMP-specific parameters that need to be unpacked from the message for MPSP handling? msgFlags, securityLevel, etc.?

The RFC3411 architecture has no ASI parameters for passing security information between the transport mapping and the dispatcher, and

between the dispatcher and the message processing model. If there is a need to have an MPSP called from the message processing model to, for example, verify that msgFlags and the transport security are consistent, then it will be necessary to pass the model-independent security parameters from the TPSP through to the MPSP.

There are four approaches that could be used for passing information between the TMSP and an MPSP.

1. we could define an ASI to supplement the existing ASIs, or
2. the TMSM could add a header to encapsulate the SNMP message,
3. the TMSM could utilize fields already defined in the existing SNMPv3 message, or
4. the TMSM could pass the information in an implementation-specific cache or via a MIB module.

3.4.1 Define an Abstract Service Iinterface

Abstract service interfaces [RFC3411] are defined by a set of primitives that specify the services provided and the abstract data elements that are to be passed when the services are invoked. Defining additional ASIs to pass the security and transport information from the transport mapping to a messaging security model has the advantage of being consistent with existing RFC3411/3412 practice, and helps to ensure that any TMSM proposals pass the necessary data, and do not cause side effects by creating model-specific dependencies between itself and other models or other subsystems other than those that are clearly defined by an ASI.

3.4.2 Using an encapsulating header

A header could encapsulate the SNMP message to pass necessary information from the TMSP to the dispatcher and then to a messaging security model. The message header would be included in the wholeMessage ASI parameter, and would be removed by a corresponding messaging model. This would imply the (one and only) messaging dispatcher would need to be modified to determine which SNMP message version was involved, and a new message processing model would need to be developed that knew how to extract the header from the message and pass it to the MPSP.

3.4.3 Modifying Existing Fields in an SNMP Message

[RFC3412] describes the SNMPv3 message, which contains fields to pass security related parameters. The TMSM could use these fields in an SNMPv3 message, or comparable fields in other message formats to pass information between transport mapping security models in different SNMP engines, and to pass information between a transport mapping security model and a corresponding messaging security model.

If the fields in an incoming SNMPv3 message are changed by the TMSP before passing it to the MPSP, then the TMSP will need to decode the ASN.1 message, modify the fields, and re-encode the message in ASN.1 before passing the message on to the message dispatcher or to the transport layer. This would require an intimate knowledge of the message format and message versions so the TMSP knew which fields could be modified. This would seriously violate the modularity of the architecture.

3.4.4 Using a cache

A cache mechanism could be used, into which the TMSP puts information about the security applied to an incoming message, and an MPSP extracts that information from the cache. Given that there may be multiple TM-security caches, a cache ID would need to be passed through an ASI so the MPSP knows which cache of information to consult.

The cache reference could be thought of as an additional parameter in the ASIs between the transport mapping and the messaging security model. The RFC3411 ASIs would not need to be changed since the SNMPv3 WG expected that additional parameters could be passed for value-add features of specific implementations.

This approach does create dependencies between a model-specific TPSP and a corresponding specific MPSP. If a TMSM-model-independent ASI parameter is passed, this approach would be consistent with the securityStateReference cache already being passed around in the ASI.

This document will describe a cache-based approach.

3.5 Architectural Requirements for Access Control

3.5.1 securityName Binding

For SNMP access control to function properly, the security mechanism must establish a security model identifier, a securityLevel, and a securityName, which is the security model independent identifier for a principal. The SNMPv3 message processing architecture subsystem relies on a security model, such as USM, to play a role in security that goes beyond protecting the message - it provides a mapping between the USM-specific principal to a security-model independent securityName which can be used for subsequent processing, such as for access control.

The TMSM is a two-stage security model, with a transport mapping security process (TMSP) and a message processing security process (MPSP). Depending on the design of the specific TMSM model, i.e.

which transport layer protocol is used, different features might be provided by the TMSP or by the MPSP. For example, the translation from a mechanism-specific authenticated identity to a securityName might be done by the TMSP or by the MPSP. [todo] It may be possible to define a consistent division of stages regardless of the transport layer protocol used, and a consistent division of functionality would be preferred.

The SNMP architecture distinguishes between messages with no authentication and no privacy (noAuthNoPriv), authentication without privacy (authNoPriv) and authentication with privacy (authPriv). Hence, the authentication of a transport layer identity plays an important role and must be considered by any TMSM, and user authentication must be available via the transport layer security protocol.

If the type of authentication provided by the transport layer (e.g. host-based or anonymous) is considered adequate to secure and/or encrypt the message, but inadequate to provide the desired granularity of access control (e.g. user-based), a second authentication, e.g. one provided by a AAA server, may be used to provide the authentication identity which is bound to the securityName. This approach would require a good analysis of the potential for man-in-the-middle attacks or masquerade possibilities.

3.5.2 Separation of Authentication and Authorization

A TMSM security model should take care to not violate the separation of authentication and authorization in the RFC3411 architecture.. The isAccessAllowed() primitive is used for passing security-model independent parameters between the subsystems of the architecture.

Mapping of (securityModel, securityName) to an access control policy should be handled within the access control subsystem, not the security subsystem, to be consistent with the modularity of the RFC3411 architecture. This separation was a deliberate decision of the SNMPv3 WG, to allow support for authentication protocols which did not provide authorization capabilities, and to support authorization schemes, such as VACM, that do not perform their own authentication.

An authorization model MAY require authentication by certain securityModels and a minimum securityLevel to allow access to the data.

TMSM is an enhancement for the SNMPv3 privacy and authentication provisions, but it is not a significant improvement for the authorization needs of SNMPv3. TMSM provides all the model-

independent parameters for the isAccessAllowed() primitive [RFC3411].

TMSM does not specify how the securityModel and securityName could be dynamically mapped to a VACM-style groupName. The mapping of (securityModel, securityName) to a groupName is a VACM-specific mechanism for naming an access control policy, and for tying the named policy to the addressing capabilities of the data modeling language (e.g. SMIv2), the operations supported, and other factors. Providing a binding outside the Access Control subsystem might create dependencies that could make it harder to develop alternate models of access control, such as one built on UNIX groups, Windows domains, XML hierarchies, or task-based controls. The preferred approach is to pass the model-independent security parameters via the isAccessAllowed() ASI, and perform the mapping within the access control model.

To provide support for protocols which simultaneously send information for authentication and authorization, such as RADIUS, model-specific authorization information MAY be cached or otherwise made available to the access control subsystem, e.g. via a MIB table similar to the vacmSecurityToGroupTable, so the access control subsystem can create an approrpiate binding between the model-independent securityModel and securityName and a model-specific access control policy. This may be highly undesirable, however, if it creates a dependency between a security model and an access control model, just as it is undesirable that the TMSM approach creates a dependency between a TMSP and an MPSP.

4. Integration with the SNMPv3 message format

TMSM proposals can use the SNMPv3 message format, defined in RFC3412, section 6. This seection discusses how the fields could be reused.

4.1 msgVersion

For proposals that reuse the SNMPv3 message format, this field should contain the value 3.

4.2 msgGlobalData

msgID and msgMaxSize are used identically for the TMSM models as for the USM model.

msgSecurityModel should be set to a value from the SnmpSecurityModel enumeration [RFC3411] to identify the specific TMSM model. Each standards-track TMSM model should have an enumeration assigned by IANA. Each enterprise-specific security model should have an enumeration assigned following instructions in the description of the

SnmpSecurityModel TEXTUAL-CONVENTION from RFC3411.

msgSecurityParameters would carry security information required for message security processing. It is unclear whether this field would be useful or what parameters would be carried to support security, since message security is provided by an external process, and msgSecurityParameters are not used by the access control subsystem.

RFC3412 defines two primitives, generateRequestMsg() and processIncomingMsg() which require the specification of an authoritative SNMP entity. [todo] We need to discuss what the meaning of authoritative would be in a TMSM environment, whether the specific services provided in USM security from msgSecurityParameters still are needed, and how the Message Processing model provides this information to the security model via generateRequestMsg() and processIncomingMsg() primitives. RFC3412 specifies that "The data in the msgSecurityParameters field is used exclusively by the Security Model, and the contents and format of the data is defined by the Security Model. This OCTET STRING is not interpreted by the v3MP, but is passed to the local implementation of the Security Model indicated by the msgSecurityModel field in the message."

msgFlags have the same values for the TMSM models as for the USM model. "The authFlag and privFlag fields indicate the securityLevel that was applied to the message before it was sent on the wire."

4.3 securityLevel and msgFlags

For an outgoing message, msgFlags is the requested security for the message; if a TMSM cannot provide the requested securityLevel, the model MUST describe a standard behavior that is followed for that situation. If the TMSM cannot provide at least the requested level of security, the TMSM MUST discard the request and SHOULD notify the message processing model that the request failed. [dbh: how is yet to be determined, and may be model-specific or implementation-specific.]

For an outgoing message, if the TMSM is able to provide stronger than requested security, that may be acceptable. The transport layer protocol would need to indicate to the receiver what security has been applied to the actual message. To avoid the need to mess with the ASN.1 encoding, the SNMPv3 message carries the requested msgFlags, not the actual securityLevel applied to the message. If a message format other than SNMPv3 is used, then the new message may carry the more accurate securityLevel in the SNMP message.

For an incoming message, the receiving TMSM knows what must be done to process the message based on the transport layer mechanisms. If the underlying transport security mechanisms for the receiver cannot

provide the matching securityLevel, then the message should follow the standard behaviors for the transport security mechanism, or be discarded silently.

Part of the responsibility of the TMSM is to ensure that the actual security provided by the underlying transport layer security mechanisms is configured to meet or exceed the securityLevel required by the msgFlags in the SNMP message. When the MPSP processes the incoming message, it should compare the msgFlags field to the securityLevel actually provided for the message by the transport layer security. If they differ, the MPSP should determine whether the changed securityLevel is acceptable. If not, it should discard the message. Depending on the model, the MPSP may issue a reportPDU with the XXXXXXX model-specific counter.

4.4 The tmStateReference for Passing Security Parameters

A tmStateReference is used to pass data between the TMSP and the MPSP, similar to the securityStateReference described in RFC3412. This can be envisioned as being appended to the ASIs between the TM and the MP or as being passed in an encapsulating header.

The TMSP may provide only some aspects of security, and leave some aspects to the MPSP. tmStateReference should be used to pass any parameters, in a model- and mechanism-specific format, that will be needed to coordinate the activities of the TMSP and MPSP, and the parameters subsequently passed in securityStateReference . For example, the TMSP may provide privacy and data integrity and authentication and authorization policy retrievals, or some subset of these features, depending on the features available in the transport mechanisms. A field in tmStateReference should identify which services were provided for each received message by the TMSP, the securityLevel applied to the received message, the model-specific security identity, the session identifier for session based transport security, and so on.

4.5 securityStateReference Cached Security Data

From RFC3411: "For each message received, the Security Model caches the state information such that a Response message can be generated using the same security information, even if the Local Configuration Datastore is altered between the time of the incoming request and the outgoing response.

A Message Processing Model has the responsibility for explicitly releasing the cached data if such data is no longer needed. To enable this, an abstract securityStateReference data element is passed from the Security Model to the Message Processing Model. The

cached security data may be implicitly released via the generation of a response, or explicitly released by using the stateRelease primitive, as described in RFC3411 section 4.5.1."

For the TMSM approach, the TMSP may need to provide information to the message processing model, such as the security-model-independent securityName, securityLevel, and securityModel parameters, and for responses, the messaging model may need to pass the parameters back to the TMSP. To differentiate what information needs to be provided to the message processing model by the TMSP, and vice-versa, this document will differentiate the tmStateReference provide by the TMSP from the securityStateReference provided by the MPSP. An implementation MAY use one cache and one reference to serve both functions, but an implementor must be aware of the cache-release issues to prevent the cache from being released before the transport mapping has had an opportunity to extract the information it needs.

4.5.1 Prepare an Outgoing SNMP Message

Following RFC3412, section 7.1, the SNMPv3 message processing model uses the generateResponseMsg() or generateRequestMsg() primitives, to call the MPSP. The message processing model, or the MPSP it calls, may need to put information into the tmStateReference cache for use by the TMSP, such as:

- tmSecurityStateReference - the unique identifier for the cached information
- tmTransportDomain
- tmTransportAddress
- tmSecurityModel - an indicator of which mechanisms to use
- tmSecurityName - a model-specific identifier of the security principal
- tmSecurityLevel - an indicator of which security services are requested

and may contain additional information such as

- tmSessionID
- tmSessionKey
- tmSessionMsgID

According to RFC3411, section 4.1.1, the application provides the transportDomain and transportAddress to the PDU dispatcher via the sendPDU() primitive. If we permit multiple sessions per transportAddress, then we would need to define how session identifiers get passed from the application to the PDU dispatcher (and then to the MP model).

The SNMP over TCP Transport Mapping document [RFC3430] says that TCP connections can be recreated dynamically or kept for future use and actually leaves all that to the transport mapping.

[todo] we might define a new transportDomain and transportAddress, which includes the address and session identifier. For situations where a session has not yet been established, we could pass a 0x0000 session identifier (or whatever) to indicate that a session should be established.

We might have a MIB module that records the session information for subsequent use by the applications and other subsytems, or it might be passed in the tmStateReference cache. For notifications, I assume the SNMPv3 notification tables would be a place to find the address, but I'm not sure how to identify the presumably-dynamic session identifiers. The MIB module could identify whether the session was initiated by the remote engine or initiated by the current engine, and possibly assigned a purpose (incoming request/response or outgoing notifications). First we need to decide whether to handle notifications and requests in one or two (or more) sessions, which might depend on the transport protocol we choose (the same problem netconf faced).

4.5.2 Prepare Data Elements from an Incoming SNMP Message

For an incoming message, the TMSP will need to put information from the transport mechanisms used into the tmStateReference so the MPSP can extract the information and add it conceptually to the securityStateReference.

The tmStateReference cache will likely contain at least the following information:

- tmStateReference - a unique identifier for the cached information
- tmSecurityStateReference - the unique identifier for the cached information
- tmTransportDomain
- tmTransportAddress
- tmSecurityModel - an indicator of which mechanisms to use
- tmSecurityName - a model-specific identifier of the security principal
- tmSecurityLevel - an indicator of which security services are requested
- tmAuthProtocol
- tmPrivProtocol

and may contain additional information such as

- tmSessionID
- tmSessionKey
- tmSessionMsgID

4.6 Notifications

For notifications, if the cache has been released and then session

closed, then the MPSP will request the TMSP to establish a session, populate the cache, and pass the securityStateReference to the MPSP.

[todo] We need to determine what state needs to be saved here.

5. Transport Mapping Security Model Samples

There are a number of standard protocols that could be proposed as possible solutions within the TMSM framework. Some factors should be considered when selecting a protocol for use within this framework.

Using a protocol in a manner for which is was not designed has numerous problems. The advertised security characteristics of a protocol may depend on its being used as designed; when used in other ways, it may not deliver the expected security characteristics. It is recommended that any proposed model include a discussion of the applicability statement of the protocols to be used.

5.1 TLS/TCP Transport Mapping Security Model

SNMP supports multiple transports. The preferred transport for SNMP over IP is UDP [RFC3417]. An experimental transport for SNMP over TCP is defined in [RFC3430].

TLS/TCP will create an association between the TMSM of one SNMP entity and the TMSM of another SNMP entity. The created "tunnel" may provide encryption and data integrity. Both encryption and data integrity are optional features in TLS. The TLS TMSP MUST provide authentication if auth is requested in the securityLevel of the SNMP message request (RFC3412 4.1.1). The TLS TM-security model MUST specify that the messages be encrypted if priv is requested in the securityLevel parameter of the SNMP message request (RFC3412 4.1.1).

The TLS TM-security model MUST support the TLS Handshake Protocol with mutual authentication.

5.1.1 tmStateReference for TLS

Upon establishment of a TLS session, the TMSP will cache the state information. A unique tmStateReference will be passed to the corresponding MPSP. The MPSP will pass the securityStateReference to the Message Processing Model for memory management.

The tmStateReference cache:

```
   tmStateReference
```

```
tmSecurityStateReference
tmTransportDomain = TCP/IPv4
tmTransportAddress = x.x.x.x:y
tmSecurityModel - TLS TMSM
tmSecurityName = "dbharrington"
tmSecurityLevel = "authPriv"
tmAuthProtocol = Handshake MD5
tmPrivProtocol = Handshake DES
tmSessionID = Handshake session identifier
tmSessionKey = Handshake peer certificate
tmSessionMasterSecret = master secret
tmSessionParameters = compression method, cipher spec, is-
resumable
```

5.1.2 MPSP for TLS TM-Security Model

```
messageProcessingModel   = SNMPv3
securityModel            = TLS TMSM
securityName             = tmSecurityName
securityLevel              = msgSecurityLevel
```

5.1.3 MIB Module for TLS Security

Each security model should use its own MIB module, rather than utilizing the USM MIB, to eliminate dependencies on a model that could be replaced some day. See RFC3411 section 4.1.1.

The TLS MIB module needs to provide the mapping from model-specific identity to a model independent securityName.

[todo] Module needs to be worked out once things become stable...

5.2 DTLS/UDP Transport Mapping Security Model

DTLS has been proposed as a UDP-based TLS. Transport Layer Security (TLS) [RFC2246] traditionally requires a connection-oriented transport and is usually used over TCP. Datagram Transport Layer Security (DTLS) [DTLS] provides security services equivalent to TLS for connection-less transports such as UDP.

DTLS provides all the security services needed from an SNMP architectural point of view. Although it is possible to derive a securityName from the public key certificates (e.g. the subject field), this approach requires installing certificates on all SNMP entities, leading to a certificate management problem which does not integrate well with established AAA systems. [todo] why does this not integrate well with existing AAA systems?

Another option is to run an authentication exchange which is integrated with TLS, such as Secure Remote Password with TLS [SRP-TLS]. A similar option would be to use Kerberos authentication with TLS as defined in [RFC2712].

It is important to stress that the authentication exchange must be integrated into the TLS mechanism to prevent man-in-the-middle attacks. While SASL [RFC2222] is often used on top of a TLS encrypted channel to authenticate users, this choice seems to be problematic until the mechanism to cryptographically bind SASL into the TLS mechanism has been defined.

DTLS will create an association between the TMSM of one SNMP entity and the TMSM of another SNMP entity. The created "tunnel" may provide encryption and data integrity. Both encryption and data integrity are optional features in DTLS. The DTLS TM-security model MUST provide authentication if auth is requested in the securityLevel of the SNMP message request (RFC3412 4.1.1). The TLS TM-security model MUST specify that the messages be encrypted if priv is requested in the securityLevel parameter of the SNMP message request (RFC3412 4.1.1).

The DTLS TM-security model MUST support the TLS Handshake Protocol with mutual authentication.

5.2.1 tmStateReference for DTLS

Upon establishment of a DTLS session, the TMSP will cache the state information. A unique tmStateReference will be passed to the corresponding MPSP. The MPSP will pass the securityStateReference to the Message Processing Model for memory management.

The tmStateReference cache:

```
   tmStateReference
   tmSecurityStateReference
   tmTransportDomain = UDP/IPv4
   tmTransportAddress = x.x.x.x:y
   tmSecurityModel - DTLS TMSM
   tmSecurityName = "dbharrington"
   tmSecurityLevel = "authPriv"
   tmAuthProtocol = Handshake MD5
   tmPrivProtocol = Handshake DES
   tmSessionID = Handshake session identifier
   tmSessionKey = Handshake peer certificate
   tmSessionMasterSecret = master secret
   tmSessionParameters = compression method, cipher spec, is-
   resumable
```

tmSessionSequence = epoch, sequence

[todo]
Need to discuss to what extent DTLS is a reasonable choice for SNMP interactions.
What is the status of the work to cryptographically bind SASL to DTLS?
More details need to be worked out...

5.3 SASL Transport Mapping Security Model

The Simple Authentication and Security Layer (SASL) [RFC2222] provides a hook for authentication and security mechanisms to be used in application protocols. SASL supports a number of authentication and security mechanisms, among them Kerberos via the GSSAPI mechanism.

This sample will use DIGEST-MD5 because it supports authentication, integrity checking, and confidentiality.

DIGEST-MD5 supports auth, auth with integrity, and auth with confidentiality. Since SNMPv3 assumes integrity checking is part of authentication, if msgFlags is set to authNoPriv, the qop-value should be set to auth-int; if msgFlags is authPriv, then qop-value should be auth-conf.

Realm is optional, but can be utilized by the securityModel if desired. SNMP does not use this value, but a TMSM could map the realm into SNMP processing in various ways. For example, realm and username could be concatenated to be the securityName value, e.g. helpdesk::username", or the realm could be used to specify a groupname to use in the VACM access control. This would be similar to having the securityName-to-group mapping done by the external AAA server.

5.3.1 tmStateReference for SASL DIGEST-MD5

The tmStateReference cache:
tmStateReference
tmSecurityStateReference
tmTransportDomain = TCP/IPv4
tmTransportAddress = x.x.x.x:y
tmSecurityModel - SASL TMSM
tmSecurityName = username
tmSecurityLevel = [auth-conf]
tmAuthProtocol = md5-sess

```
   tmPrivProtocol =  3des
   tmServicesProvided =
      mutual authentication,
      reauthentication,
      integrity,
      encryption
   tmParameters = "realm=helpdesk, serv-type=SNMP
```

6. Security Considerations

This document describes an architectural approach and multiple proposed configurations that would permit SNMPv3 to utilize transport layer security services. Each section containing a proposal should discuss the security considerations of that approach. [todo] expand as needed.

Perfect forward secrecy guarantees that compromise of long term secret keys does not result in disclosure of past session keys.

It is considered desirable by some industry segements that TMSM security models should utilize transport layer security that addresses perfect forward secrecy at least for encryption keys.

7. Acknowledgments

The authors would like to thank Ira McDonald, Ken Hornstein, and Nagendra Modadugu for their comments and suggestions.

8. References

8.1 Normative References

[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 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.

[RFC3417] Presuhn (Editor), R., "Transport Mappings for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3417, December 2002.

[RFC3430] Schoenwaelder, J., "Simple Network Management Protocol (SNMP) over Transmission Control Protocol (TCP) Transport Mapping", RFC 3430, December 2002.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC1510] Kohl, J. and B. Neuman, "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993.

[RFC2222] Myers, J., "Simple Authentication and Security Layer (SASL)", STD 62, RFC RFC2222, October 1997.

[DTLS] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", ID draft-rescorla-dtls-01.txt, July 2004.

8.2 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.

[RFC2712] Medvinsky, A. and M. Hur, "Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)", RFC 2712, October 1999.

[SRP-TLS] Taylor, D., Wu, T., Mavroyanopoulos, M., and T. Perrin, "Using SRP for TLS Authentication", ID draft-ietf-tls-srp-08.txt, August 2004.

[EUSM] Narayan, D., McCloghrie, K., Salowey, J., and C. Elliot, "External USM for SNMPv3", ID draft-kaushik-snmp-external-usm-00.txt, July 2004.

[NETCONF] Enns, R., "NETCONF Configuration Protocol", ID draft-ietf-netconf-prot-04.txt, October 2004.

[SSHauth] Lonvick, C., "SSH Authentication Protocol", ID draft-ietf-secsh-userauth-21.txt, June 2004.

Authors' Addresses

David Harrington
Independent
Harding Rd
Portsmouth NH
USA

Phone: +1 603 436 8634
Email: dbharrington@comcast.net

Juergen Schoenwaelder
International University Bremen
Campus Ring 1
28725 Bremen
Germany

Phone: +49 421 200-3587
Email: j.schoenwaelder@iu-bremen.de

Appendix A. Questions about msgFlags:

[todo] many of these questions can be resolved by deciding whether the TMSP or MPSP provides the service of comparing msgFlags (from inside the message) to actual capabilities of the transport layer security (external to the message). It may however be necessary to provide this service for two slightly different purposes depending on whether the message is outgoing (and may need to be checked by the TMSP when a new transport session might be created) or the message is incoming (the capabilities of the transport layer session are already known, but msgFlags has not been unpacked yet at the TMSP, so the comparison must be done at the MPSP). Of course, we really only need to compare the authflag and the privflag, i.e. the securityLevel, so if we pass the securityLevel between the two stages, then they each have the info they need to do their respective comparisons.

There have been a large number of questions about msgFlags in the TMSM approach, mostly concerning the msgFlags value and the actual security provided, and whether msgFlags can be used to initiate per-message or per-session security.

A.1 msgFlags versus actual security

Using IPSEC, SSH, or SSL/TLS to provide security services "below" the SNMP message, the use of securityName and securityLevel will differ from the USM/VACM approach to SNMP access control. VACM uses the

"securityName" and the "securityLevel" to determine if access is allowed. With the SNMPv3 message and USM security model, both securityLevel and securityName are contained in every SNMPv3 message.

Any proposal for a security model using IPSEC, SSH, or SSL/TLS needs to specify how this info is made available to the SNMPv3 message processing, and how it is used.

One specific case to consider is the relationship between the msgFlags of an SNMPv3 message, and the actual services provided by the lower layer security. For example, if a session is set up with encryption, is the priv bit always (or never) set in the msgFlags field, and is the PDU never (or always) encrypted? Do msgFlags have to match the security services provided by the lower layer, or are the msgFlags ignored and the values from the lower layer used?

> Is the securityLevel looked at before the security model gets to it.? No. the security model has two parts - the TMSP and the MPSP. The securityLevel is looked at by the TMSP before it gets to the MPSP, but both are parts of the same security model.
> Would it be legal for the security model to ignore the incoming flags and change them before passing them back up? If it changed them, it wouldn't necessarily be ignoring them. The TMSP should pass both an actual securityLevel applied to the message, and the msgFlags in the SNMP message to the MPSP for consideration related to access control.. The msgFlags parameter in the SNMP message is never changed when processing an incoming message.
> Would it be legal for the security model to ignore the outgoing flags and change them before passing them out? no; because the two stages are parts of the same security model, either the MPSP should recognize that a securityLevel cannot be met or exceeded, and reject the message during the message-build phase, or the TMSP should determine if it is possible to honor the request. It is possible to apply an increased securityLevel for an outgoing request, but the procedure to do so must be spelled out clearly in the model design.
> The security model MUST check the incoming security level flags to make sure they matched the transport session setup. and if not drop the message. Yes, mostly. Depending on the model, either the TMSP or the MPSP MUST verify that the actual processing met or exceeded the securityLevel requested by the msgFlags and that it is acceptable to the specific-model processing (or operator configuration) for this different securityLevel to be applied to the message. This is also true (especially) for outgoing messages.
> You might legally be able to have a authNoPriv message that is actually encrypted via the transport (but not the other way around of course). Yes, a TMSM could define that as the behavior (or

permit an operator to specify that is acceptable behavior) when a requested securityLevel cannot be provided, but a stronger securityLevel can be provided.

A.2 Message security versus session security

For SBSM, and for many TMSM models, securityName is specified during session setup, and associated with the session identifier. Is it possible for the request (and notification) originator to specify per message auth and encryption services, or are they are "fixed" by the transport/session model?
If a session is created as 'authPriv', then keys for encryption would still be negotiated once at the beginning of the session. But if a message is presented to the session with a security level of authNoPriv, then that message could simply be authenticated and not encrypted. Wouldn't that also have some security benefit, in that it reduces the encrypted data available to an attacker gathering packets to try and discover the encryption keys?
Some SNMP entities are resource-constrained. Adding sessions increases the need for resources, we shouldn't require two sessions when one can suffice. 2 bytes per session structure and a compare or two is much less of a resource burden than two separate sessions.
It's not just about CPU power of the device but the percentage of CPU cycles that are spent on network management. There isn't much value in using encryption for a performance management system polling PEs for performance data on thousands of interfaces every ten minutes,it just adds significant overhead to processing of the packet. Using an encrypted TLS channel for everything may not work for use cases in performance management wherein we collect massive amounts of non sensitive data at periodic intervals. Each SNMP "session" would have to negotiate two separate protection channels (authPriv and authNoPriv) and for every packet the SNMP engine will use the appropriate channel based on the desired securityLevel.
If the underlying transport layer security was configurable on a per-message basis, a TMSM could have a MIB module with configurable maxSecurityLevel and a minSecurityLevel objects to identify the range of possible levels, and not all messages sent via that session are of the same level. A session's maxSecurityLevel would identify the maximum security it could provide, and a session created with a minSecurityLevel of authPriv would reject an attempt to send an authNoPriv message.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.