snmp v3 (one more time)

Network Mgmt/Sec.

aka

- is three really the charm?

Outline

- intro
- architecture/structure -
- more protocol
- user-based security
  - authentication/encryption/key management
- view-based access control (brief)

bibliography, RFC-wise (jan/98)

- 2271 - Architecture
- 2272 - Message Processing/Dispatch
- 2273 - v3 applications (functional parts)
- 2274 - User-Based Security Model
- 2275 - View-Based Access Control Model

so what is it?

- v2 +
  - protocol security, i.e., authentication/confidentiality/key management
    - » this is the **User-Based Security model**
    - » **note: confidentiality is called privacy**
  - an enhanced access-control model
    - » based on MIB views
    - » and groups
    - » this is the **View-Based Access Control model**

protocol overview

- roughly (v3 wrapper, (v2/v1 PDU))

snmp v3 part

global hdr	sec. model hdr	scoped pdu hdr	snmp pdu (v2 bits)

crypto-fied part

pdu part may be authenticated OR auth/encrypted

architectural elements/dictionary

- snmpEngineId - a string that uniquely defines a manager or agent or combo, note == contextEngineId, may have many contexts
- contextEngineId - identity with a context
- contextName - parameter to access control subsystem, set of MIB objects, string
  - objects can be lumped into named sets with different rights associated (readonly, writeable)

we have new terms

- scopedPDU - PDU with contextEngineID, contextName
- snmpSecurityModel - which sec. model
  - v1, v2c, USM are possible
- snmpSecurityLevel
  - noAuthnoPriv, authNoPriv, authPriv
- principal - for whom do we do all this crud
  - people ultimately but processes in real life
- securityName - string representing principal

snmpEngineId something like:

- 12 bytes long acc. to AgentID, rfc 1910 or
- (4 bytes of private enterprise number) + more bytes with byte 5 indicating:
  - IPv4 address (thus 6,7,8,9) (byte 5 = 1)
  - IPv6 address (add 16)
  - MAC
  - text (max is 27)

generalized abstract architecture

applications (snmp of course) - forms pdus, get, response, etc

snmp engine (id is snmpEngineId) - does subsystem processing including security, access control

applications include

- **command generator**, *does get/getnext/getbulk/set and processes response received
- **command responder**, recvs get, etc., and sends response
- **notification recv/originator** (traps/informs)

snmp engine parts

- **dispatcher** - i/f to apps/network/and other snmp engine parts
- **message processing**- tie the v3 message together from sub-systems including:
  – and might do v1/v2/v3 messages ...
- **security subsystem** - user-based sec. model
- **access control** - view-based model

access-control subsystem primtitive (e.g.,)

- isAccessAllowed() parameters include:
  – securityModel INPUT (e.g., USM)
  – securityName INPUT (principal)
  – securityLevel IN (e.g., auth only)
  – viewType IN (read/write/notify view)
  – contextName IN (contains variableName)
  – variableName IN (OID ...)
  – statusInformation returned - OK or boo hiss

application MIBs for v3

- SNMP-TARGET-MIB defines objects for defining who to send notifications and/or proxy messages to
- SNMP-NOTIFICATION-MIB defines objects for remote config of notifications
- SNMP-PROXY-MIB contains info for remote config of proxy

SNMP-TARGET-MIB

- snmpTargetObjects contains:
  - snmpTargetSpinLock(1)
  - snmpTargetAddrTable(2)
    - » use UDP with IP addr Y, timeout T, retry R etc.
  - snmpTargetParamsTable(3)
    - » v1/v2/v3, securityModel, securityName, securityLevel
  - and more ...

NOTIFICATION MIB

- three tables including:
  – snmpNotifyTable - select who to notify from previous snmpTargetAddrTable
  – snmpNotifyFilterProfileTable - associate filters with a particular target
  – snmpNotifyfilterTable - define filters

protocol overview

- roughly (v3 wrapper, (v2/v1 PDU))

snmp v3 part

global hdr	sec. model hdr	scoped pdu hdr	snmp pdu (v2 bits)

crypto-fied part

pdu part may be authenticated OR auth/encrypted

header contains 5 fields (still TLVs):

- msgVersion (3 for v3) - INTEGER
- msgID (request ID) - INTEGER
- msgMaxSize - max segment size e.g., response
- msgFlags - flag bits,
  - reportableFlag (bit 1) - if set, report may be sent, if not set, report may NOT be set (used if message encrypted but not decodable)
  - privFlag = 1, was encrypted
  - authFlag = 1, was authenticated

cont.

- msgSecurityModel - 1 for v1, 2 for v2c, 3 for USM

scoped pdu has 3 parts

- context ID (engine ID) and
- context name
- used in access control

USM header part

- msgAuthoritativeEngineId - likely the agent's engine Id
- **msgAuthoritativeEngineBoots** - # of reboots
- **msgAuthoritativeEngineTime** - # of secs since reboot
- msgUserName - principal name for msg
- msgAuthenticationParameters - hash
- msgPrivacyParameters - IV

USM - authoritative engine

- means receiver of get/getnext/getbulk/set/inform
- OR sender of trap/response
- time in message consists of 2-tuple
  – (boot count N, time since boot)
- based on clock in authoritative engine
  – non-authoritative msg sender must estimate time in authoritative engine

initialization protocol exists

- 2-step discovery mechanism exists to allow one to obtain info about other entities (agents)
- nonauth. engine sends request with no security and user name "initial", recvs **snmpEngineId** back
- if authentication used, sends authenticated request to get back **boots** and **time** values

USM functionality (security model)

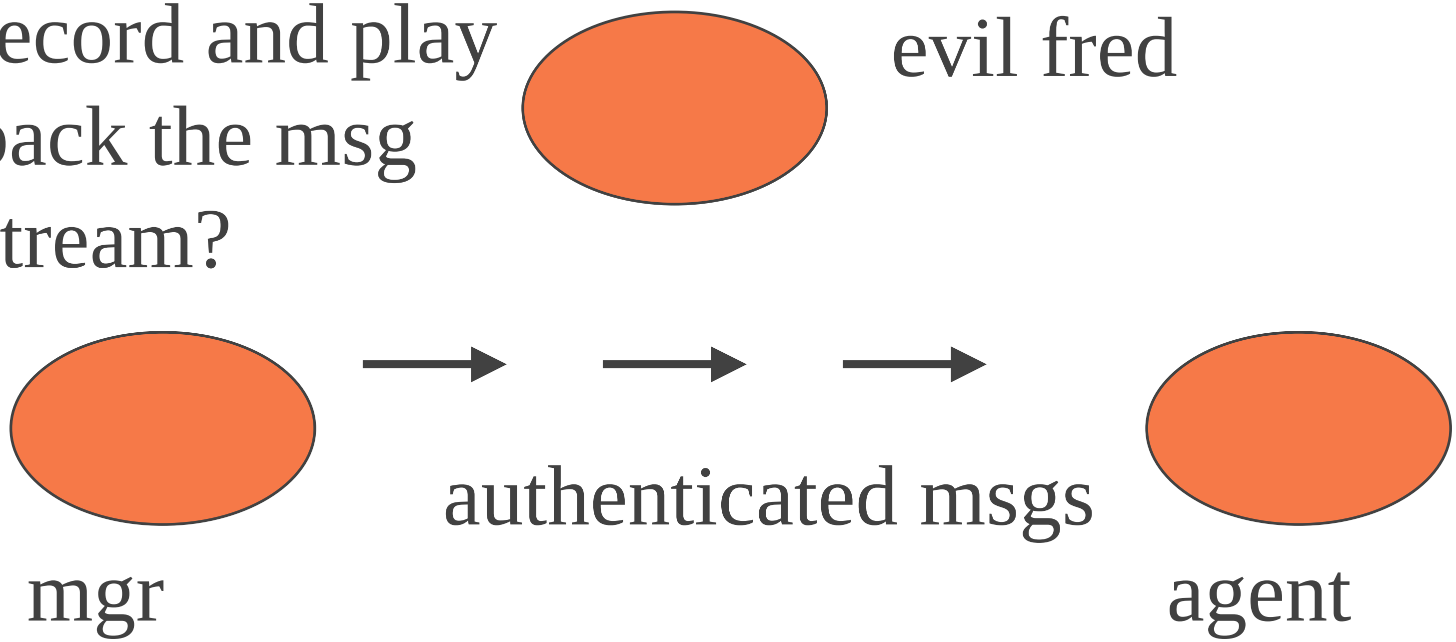
- crypto
- anti-replay and time
- usmUser group
- key management
  - password to hash key mechanism
  - key localization
  - key update

USM security overview

- denial of service attacks - NO help here
  - probably such an attack is more fundamental than just an attack on SNMP
- traffic analysis - NO
- note that authentication is strong
  - encryption if DES based not so strong
  - may need better algorithm or folding of SNMP across WAN inside IPSEC
- anti-replay features exist (time)

anti-replay

could fred
record and play
back the msg
stream?



time-replay

- (boot count, ticks since boot) in message
- allows recv to judge if message is "too old", can send notInTimeWindow error
- non-auth. engine must keep track of auth. engines time
- auth. engine must keep time window and track timeliness of messages (so does non-auth engine)
- roughly boot count must match and msg must be within 150 seconds of stored time

crypto

- authentication algorithms
  - HMAC-MD5-96
    - » 128 bit authKey/output truncated to 96 bit hash
  - HMAC-SHA-96
    - » 160 bit key/output again truncated to 96 bits
- encryption
  - DES-CBC (still a revolutionary act)
    - » 56 bit in 8 octets, least significant bit ignored

usmUser group

- information about local and remote principals
- spinlock plus usmUserTable which includes:
  - usmUserEngineID - authoritative engine ID (local/remote)
  - usmUserName - principal name, e.g., “bob”
  - usmUserSecurityName - securityName (same as above)
  - usmUserCloneFrom - pointer to another conceptual row - used to point to keys and clone keys
  - usmUserAuthProtocol - none, hmac-md5 (def), h-sha

cont.

- usmUserAuthKeyChange - byte string with keyChange syntax. causes keyChange hash function to occur in terms of key update
- usmUserOwnAuthKeyChange - user can change only own key
- usmUserPrivProtocol - none (def), DES
- usmUserPrivKeyChange - drive key change
- usmUserOwnPrivKeyChange
- usmUserPublic - not clear

processing

- to send
  - encrypt 1st then authenticate
  - before authentication, store time values
- to recv
  - note securityEngineID/securityName used for lookup in usmUserTable
  - perform authentication check
  - then perform decrypt if needed

key management

- it is assumed that there are separate keys for authentication and encryption (by definition)
- each principal has a pair of keys
- keys cannot be obtained via SNMP (gets)
  - duh ...
- **key localization** and **password to key** allow one user to access many agents with one set of keys

password to key

- human-readable password mapped by function into desired crypto keys
- RFC 2274 defines algorithm
- passwords should not be poor of course
- password is duplicated to produce string of 2 ** 20 bytes (slow down brute-force attack)
- if MD5 used, take md5 hash to form digest

key localization

- keys are localized by taking hashed user key and hashing it again with remote EngineID value
- f (user key(hashed), EngineID) -> digest/key
- this is called localized key
- upshot: use localized key - if attacker captures on wire, && breaks it, can only attack one agent

key update

- initial keys must “somehow” be installed out of band
  - assume manually
- post initial installation, key change or update process can be initiated by user
- done by using hash function + XOR
- f(keyOld, random bits, etc)., where the keys are not sent, but random bits are transmitted
- hash function is one way, if new key learned, old key not derivable

view-based access control model

- group
  - has groupname
  - list of principals (securityName) + securityModel with same access rights
  - securityModel e.g., USM or SNMPv1 community
- context
  - basically a MIB view (defined by tree and filter)
  - has name

and the point ...

- we might wish to restrict access on engine Y to
  - user Z can read anything but can’t write anything
  - user X can write but has to have a auth key with USM
    - » or stronger ... encrypted too
  - notify privileges must be considered as well (e.g., traps must be authenticated)

scoped pdu in v3 headers

- contextEngineId - which application gets it
- contextName - the MIB view via a simple string name
- and yes, the actual SNMP PDU (presumably a V2 PDU)

VBACM elements

- msgUserName/msgSecurityModel mapped into groupName
- contextEngineID - who carries it out
- contextName - what MIB objects
- msgAuthenticationParameters - checked out by USM before it gets to VACM
- access control based on groupName, contextName, security Model, securityLevel (from msgFlags)
- MIB view determined by vacmAccessTable and PDU operation type (read/write/notify)