TELE9752 Network Operations and Control

Week 8: Security management

SNMP = ?

Security is Not My Problem
Outline

• Security
  – services
  – mechanisms
• Security & NM
• SNMPv3 security
  – Entities
    • Engines
    • Key localisation
    • Contexts & users
  – Message format
  – User Security Model (USM)
  – View-based Access Control Model (VACM)
• Configuring Access Control Lists
Security objectives

Consider Alice sending to Bob in presence of intruder Trudy
  - e.g. A=lecturer/Mgr, B=uni records/element,
    T=someone tampering with marks or eavesdropping

Desired aspects:
• **Secrecy** (aka confidentiality & “privacy” [SNMP]): T can't understand/detect A sending to B
  - of information
  - of traffic: occurrence, timing, length, etc
• **Integrity**: B receives what A sends, unmodified by T
• **Authentication**: B knows that A sent it.
  Requires integrity, but integrity doesn't require auth
• **Timeliness**: B receives it just after A sends it. Not excessively delayed, or replayed (e.g. directive to reboot router)

& many more (e.g. anonymous voting)

TELE3119 covers trusted networks in greater depth, e.g. details about cryptographic methods & security protocols
Mechanism 1: Information security

Protect information by transforming it to have properties that hinder attacks.

**Cryptography**: Key controls transformation
- **Symmetric**: A&B use same key. Manage many keys
- **Asymmetric**: X has public+private keys. Encrypt secrets for X or authenticate X through public key
- **Initialisation Vectors (IVs)**: Random # precedes payload making msg unique; thwarts cryptanalysis

**Hashing**: Map large volume of info into smaller. To:
- Reduce processing: authenticate hash rather than whole
- Like hashing for packet classification (TELE9751) but must carefully consider collisions: stronger adversaries
- Blend info, e.g. device key = H(user key,deviceId) [M0>
- MAC = Message Authentication Code (not medium access control)
Mechanism 2: Security devices

**Physical** separation: e.g. out-of-band management

**Access control**: Programmed checks that access is permitted, e.g.
- View Based Access Control in SNMP
- **Firewalls**: Border device only passes packets that satisfy rules
  - On border of computer (e.g. personal firewall software), or network (e.g. firewall device or feature on existing device, e.g. router)
  - Rules need to be managed [3U].
- **Access control servers** Part of “Authentication, Authorization and Accounting (AAA)” [KD>

**Intrusion Detection Systems**: Spot anomalies by monitoring network activity. Like fault detection.
Security & NM are symbiotic

Security needs configuration management
• of keys
• Access Control Lists (ACLs) [Q7]
• update configuration of devices to correct bugs that create security weaknesses

Network management needs security, e.g.
Secrecy: Traffic stats reveal:
– who uses network when & to whom - concern to users
– volumes of traffic - of interest to competitors
Authentication: Configure devices => control how/if they work.
Security of mgt + Mgt of security

- Need to secure management services (upper Fig)
- Need to manage security services (lower Fig)
Tension between security & NM

- **Range of services**
  - Minimise for security
    - Exclude those with known vulnerabilities; less to concentrate on
    - Minimal set: HTTP, DNS + SMTP, IMAP, FTP
  - Increase for NM, e.g. ICMP, traceroute, ping, loose source routing, whois

- **Feedback to user** about why access failed & suggested workarounds (e.g. ICMP unreachable errors)
  - Minimise for security
  - Increase for NM
Fault management:

- **<K1>** Tension between security & NM: Elaborating on why something was refused helps F but hinders S
- Attacks and faults cause similar degradations, leading to similar protection (e.g. Integrity checks) but stronger for security **<8L>**
- Security is part of dependability
  - as are availability and reliability
  - Denial of Service impinges availability
- Monitoring/Anomaly detection: NM detects faults; Security to detect intrusions

Some security mechanisms† use OIDs, ASN.1 and BER

Same IT people often do both security and NM

† e.g. Kerberos and X.500
Outline

SNMPv3 security
Protecting SNMP itself

Before securing mgt info, consider securing the management protocol, SNMP

• e.g. Agent accepts SNMP packets, but should check that they are well-formed. Non-trivial given generality (=> complexity) of “Basic” Encoding Rules+SNMP PDUs
  – In 2002: “badly formatted packets caused the implementation to corrupt its memory maps ... with some implementations causing the managed device to enter a loop of continuous rebooting, and others allowing the device to then run injected code, thereby allowing the device to be hijacked by the attacker.”

• Mgr should monitor counters of anomalies in SNMP MIB
  <V3>:\n  \snmpInBadVersions, \snmpInBadCommunityNames, \snmpInASNParseErrs etc

Details of vulnerabilities at http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/
SNMP security

SNMP v1/2 had community strings

- Very weak security:
  - Can authenticate manager, provided monitor for brute-force attacks.
  - No protection against eavesdropping: intruder that captures one msg can then create msgs that seem authentic

=> Config usually manual, rather than unauthenticated

SNMPv1/2 set requests

“SNMPv3 is SNMPv2 plus security and administration”

security = privacy+auth (USM [ZW]) + access control (VACM [73])
admin = naming entities + we won't cover: notification destinations & proxies
SNMPv3 references

- SNMPv3 IETF STD62 = RFCs 3410-9
  - Previous iterations of RFCs: 2271-5 (originals), 2571-5
- Stallings (in order of increasing bulk)
  - “SNMPv3: A security enhancement for SNMP”, IEEE Communications Surveys
  - Ch. 8 of Network Security Essentials: Applications and Standards, 3rd edition
  - Ch. 15-17 of SNMP, SNMPv2, SNMPv3, RMON 1 and 2, 3rd edition

† the phrase “for privFlag=0, authentication was applied” should read “for authFlag=1...”
Outline

– Entities
  • Engines
  • Key localisation
  • Contexts & users
Secure SNMP with a MsgProc engine between UDP & SNMP PDUs

- Basic idea is to insert another layer between transport and SNMP PDUs (get/trap/etc)
- Familiar?

Fig 8.5 from Stallings: *Network Security Essentials*
SNMP engines

“SNMP Engine” = an instance of the layer that secures SNMP PDUs. `snmpEngine group [RFC3411]:`

- `snmpEngineID OCTET STRING (SIZE(5..32))`
  - 1st bit indicates format of remainder; one format: 4B enterprise # from OID space; 5th B: coding of remainder, remainder: IPv4/6 addr or MAC addr
- `snmpEngineBoots INTEGER`
  - # of times engine has rebooted since configuring `snmpEngineID`
- `snmpEngineTime INTEGER`
  - time since last boot, in seconds. Used to ensure timeliness. Like `sysUpTime` but SNMP engine may be reset independently of system? if `EngineTime` wraps, ++Boots
- `snmpEngineMaxMessageSize INTEGER`
  - Suggested max msg size for others to use
Key localisation

- Distributed systems require many keys => tempting to use same key for all, but compromise of one may lead to compromise of all.

- Key localisation uses hash function to blend a user-specific key with EngineID, creating a distinct key for each Engine.

Fig. 16.9 from Stallings aka “key localization”
“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.” [RFC 3411]

- “Often, a context is a physical device,
- or perhaps, a logical device, although a context can also encompass multiple devices, or a subset of a single device, or even a subset of multiple devices”
- identified by a textual contextName

• An engine may handle multiple “contexts”
  - contextEngineID = engineID

Access to contexts is restricted to approved “users”, identified by textual userName (allocated within NMS)

e.g. user “Tim” may read MAC addr of “EE&T hosts”
Forgettable details

• The previous slide glosses over details:
  • “users” are specific to the User Security Model; more generic term (also applicable to VACM) is “participant”, which is identified by a “securityName”
  • $\text{securityName} + \text{securityModel} \Rightarrow \text{group} \Rightarrow \text{groupName}$
Outline: SNMPv3 messages

Coming up: One slide about each underlined red topic, in top-to-bottom order

**SNMPv3Message** ::= SEQUENCE {
  msgVersion INTEGER ( 0 .. 2147483647 ),
  msgGlobalData **HeaderData** ::= SEQUENCE {
    msgID INTEGER (0..2147483647),
    msgMaxSize INTEGER (484..2147483647),
    msgFlags OCTET STRING (SIZE(1)),
    msgSecurityModel INTEGER (1..2147483647)
  }

  -- User Security Model
  msg[USM]SecurityParameters OCTET STRING ::= SEQUENCE {
    msgAuthoritativeEngineID OCTET STRING, -- Engine discovery
    -- Timeliness checking using EngineBoots & EngineTime
    msgAuthoritativeEngineBoots INTEGER (0..2147483647),
    msgAuthoritativeEngineTime INTEGER (0..2147483647),
    msgUserName OCTET STRING (SIZE(0..32)),
    msgAuthenticationParameters OCTET STRING,
    msgPrivacyParameters OCTET STRING
  }

  msgData ScopedPduData
}
SNMPv3 Message format

SNMPv3Message ::= SEQUENCE {
    msgVersion INTEGER ( 0 .. 2147483647 ), snmpv3
    msgGlobalData HeaderData, defined on next slide [C6]
    msgSecurityParameters OCTET STRING, depends on security
    model, e.g. UsmSecurityParameters
    msgData  ScopedPduData
}

ScopedPduData ::= CHOICE {
    plaintext    ScopedPDU,
    encryptedPDU OCTET STRING
}

ScopedPDU ::= SEQUENCE {
    contextEngineID   OCTET STRING,
    contextName       OCTET STRING,
    data              ANY e.g. get, trap etc
}

A ScopedPDU is just a PDU in a specific context (contextEngineID + contextName)

From RFC 3412

Copyright © Tim Moors 2013
HeaderData

- **msgID**: To match responses to requests. Similar to request-id, but outside encryptedPDU => can be used to determine crypto parameters?
- **msgMaxSize**: How big can responses to the sender be?
- **msgFlags (=>SecurityLevel)**: Single octet, with 3 binary flags:
  - **authFlag**: Msg contains authentication info
  - **privFlag**: Msg has been encrypted for privacy
    - privFlag=1 authFlag=0 “must NOT be used” - why? (?Because symmetric crypto implies auth?)
  - **reportableFlag**: Controls generation of Report PDUs, but Reports are not standardised[RFC3416]
- **msgSecurityModel**: 1=SNMPv1, 2=SNMPv2c, 3=USM
User Security Model (USM)

- Protects privacy, authentication & timeliness of msgs
  - Defined in RFC3414

- **Privacy** through symmetric Data Encryption Standard (DES) with an Initialization Vector
- **Authenticates** using HMAC with either MD5 or SHA-1 as the digest functions
  - A & B hold secret that they hash with msg to obtain HMAC. Secret not revealed through msg+HMAC
  - For details, see H. Krawczyk et al: “HMAC: Keyed-Hashing for Message Authentication” IETF RFC 2104

- **Timeliness** through timestamps & authentication
  - [FQ], needing msgSecurityParameters [KX] & engineDiscovery [X1]
Content of `msgSecurityParameters`:

```
UsmSecurityParameters ::= SEQUENCE {
  -- global User-based security parameters
  msgAuthoritativeEngineID     OCTET STRING,
  msgAuthoritativeEngineBoots  INTEGER (0..2147483647),
  msgAuthoritativeEngineTime   INTEGER (0..2147483647),
  msgUserName          OCTET STRING (SIZE(0..32)),

  -- authentication protocol specific parameters
  msgAuthenticationParameters  OCTET STRING,

    12B of HMAC of whole message

  -- privacy protocol specific parameters
  msgPrivacyParameters         OCTET STRING

    8B that imply the initialization vector
}
```

`msgAuthenticationParameters & msgPrivacyParameters ==
empty strings if authFlag or privFlag respectively ==0`
Engine discovery

How to determine EngineID, EngineBoots, EngineTime for 1st interaction?

• **Send Request with** `securityLevel==noAuthNoPriv & no content`
  - “The response to this message will be a Report message containing the snmpEngineID”
  Yes: A Report PDU that isn't defined in RFC3416 or elsewhere!

• **Send Request with** `authFlag==1 & msgUserName`
  - Response is another Report PDU indicating `snmpEngineBoots and snmpEngineTime`
Timeliness checking

Each engine tracks Boots & Time for other engines.

– No global clock

To ensure timeliness:

• **Sender estimates time at receiver**
  (“AuthoritativeEngine”) & includes that in message
  (msgAuthoritativeEngineBoots & ...Time)

• **Receiver rejects messages that are old**
  ...Boots!=own or |...Time - own|>150sec

  Replies to others.

  – 150sec allows for clock drift + comm. delays

• **Sender updates Boots&Time according to reply**
  – Sender records latestReceivedEngineTime &
    only updates it if < Time in reply, protecting against
    replay of earlier replies.
View-based Access Control Model (VACM)

USM protects mgt info *crossing the network*
VACM protects access to mgt info *inside an agent*

The term “view” gets used in 2 ways:
Verb: type of access (read = get, write = set, notify = trap/inform), e.g. `viewType`

Noun: *subset of objects for which access is controlled*
  e.g. “*>View-based Access Control Model*”,
  - subset defined by a family (list, in a table) of subtrees
  - subtrees are defined by OID of interior node of tree
  - can specify subtrees to allow or deny access to; e.g. allow broad subtree but deny access to a sub-subtree
Recall: `securityName` \(\approx\) `userName`
`securityModel` \(\approx\) USM
`securityLevel` \(\approx\) private / authenticated

Fig 17.2 from Stallings SNMP...
Factors that affect access

• **Who wants access?**: `securityName` *(from `msgUserName` for USM)*
  – e.g. read-only for junior mgr, read-write for senior mgr

• **How secure was their request?** `securityLevel`
  – e.g. 1: authentication: needed to write, but not to read
  – e.g. 2: require privacy when accessing sensitive info

• **securityModel**: modulates the above parameters
  – e.g. authentication:
    weak community string=>read, strong USM=>write

• **What do they want to access?**
  – `contextName`: where: e.g. which device
  – `variableName`: which object

• **What type of access do they seek?** "`viewType`"
Access control mechanism

- **vacmAccessTable**
  - indexed by Context, SecurityLevel, groupName and SecurityModel
  - provides 'names' of read, write & notify views
- **vacmViewTreeFamilyTable** defines accessible objects
  - indexed by
    - 'name' (from vacmAccessTable), and
    - Subtree: distinguishes different rows
      - Family = set of Subtrees
      - Compare with variableName being accessed
  - Each row defines whether access is allowed (==included) or denied (==excluded).
  - A variableName may match 0 or more rows.
    - Default is to deny access? =>
    - Access allowed if at least 1 row matches & all matching rows == included?
SNMPv3 mysteries

Bonus marks if you can find answers:

- RFCs describe values of `errorIndication` generated by modules (e.g. `noSuchContext`, `noGroupName`), but how are they conveyed to requester (e.g. through SNMP from agent to Manager)?
- Why is `msgMaxSize` sent in each message, rather than, say, once per BootTime? (stateless SNMP?)
- And mysteries raised on earlier slides [L3] [C6] [MQ]
Outline

Configuring Access Control Lists
**Access control**

**Access**: Interacting with a resource, e.g. Send to, read from, write to

**Control**: Limit who can access what.

Specified through rules:

Rule = location [T1] + circumstances [LX] + action [AP]

Access Control Lists of multiple rules

“one recent survey [9 [from 2007]] reported that ISPs called management of access control lists (ACLs) their “most critical” problem.” [Bellovin]
Locations

Firewall protects a trusted zone from an untrusted zone
Assume that attacks only come from untrusted zone

Default actions:
- Deny incoming traffic
- Allow outgoing traffic

Except: Internet services, e.g. Web, DNS
Accept inbound connections only for servers in a “demilitarized zone (DMZ)

Rules apply at each interface
Policies apply across all interfaces.

Copyright © Tim Moors 2013
Circumstances

Rules may depend on:

• Values of **packet bits**.
  Usually header fields; may involve deep packet inspection

• Stateful firewalls consider **past** circumstances / events
  e.g. allow HTTP response if previously saw HTTP request
  May release state after timeout, e.g. Forget connection
  after 5 minutes

Rules include matching, mismatching, range checking
# Actions

## Types of action:
- **Allow / Deny**
  - Usually silently discard
  - Might send ICMP Administratively Prohibited
- **Log**
  - Especially for serious breaches

## Access Control Lists state rules *in prioritized order*

<table>
<thead>
<tr>
<th>Incoming traffic</th>
<th>Outgoing traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log/notify-NM for very bad behaviour</td>
<td>Deny if malformed</td>
</tr>
</tbody>
</table>

**Exceptions:**
- Allow responses to our requests
- Deny bad behavior e.g. SMTP from hosts

**Default:**
- Deny
- Allow
## Sample access control list

<table>
<thead>
<tr>
<th>Action</th>
<th>src</th>
<th>dst</th>
<th>proto</th>
<th>Transport fields</th>
<th>State</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td># Allow outbound DNS requests &amp; corresponding inbound responses</td>
<td>allow</td>
<td>149.11/16</td>
<td>!149.11/16</td>
<td>UDP</td>
<td>&gt;1023</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>allow</td>
<td>!149.11/16</td>
<td>149.11/16</td>
<td>UDP</td>
<td>53</td>
<td>&gt;1023</td>
</tr>
<tr>
<td># Allow our users to connect to external web servers &amp; receive replies</td>
<td>allow</td>
<td>149.11/16</td>
<td>!149.11/16</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>allow</td>
<td>!149.11/16</td>
<td>149.11/16</td>
<td>TCP</td>
<td>80</td>
<td>&gt;1023</td>
</tr>
<tr>
<td># Block everything else!</td>
<td>deny</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Requiring external address not in our network helps detect internal routing problems. Client port >1023 stops external attempt to connect to well known port.

! means not, as in C programming language & descendents
The end of security