# TELE9752 Network Operations and Control

## Lecture 6 / Week 8: Security management

**SNMP** = ?

**S**ecurity is **N**ot **M**y **P**roblem

# 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 the 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 <5C])

& many more (e.g. anonymous voting)

TELE3119 covers trusted networks in greater depth, e.g. details about cryptographic mechanisms & 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. Many=N(N-1) keys. Fast.
- ***Asymmetric***: X has public+private keys. Encrypt secrets for X or authenticate X through public key. 2N keys. Slow.
- **Initialisation Vectors (IVs)**: Random # precedes payload making msg unique; thwarts cryptanalysis

**Hashing**: Map large amount 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)

# Hashing examples

**Crude (very weak) hash function**: text -> 4 digit #

Count # of vowels, consonants, digits, punctuation in text;
use last digit of each count for hash digit.

"TELE9752 rocks!" -> 3642

**Applications**:

- **Integrity check**: "WXYZ9752 rocks!" -> 1842<>3642
  - Anybody knowing hash function could change hash also
  - "TELE9752 sucks!" has same hash value (collision)
- **Authentication**: Shared secret abracadabra [5vowels, 6consonants] appended before hashing. Value=8(1)242
- **Blending** ("key localisation" [M0>): Different secret for different sites: google=33, facebook=44, unsw=13





2!!!

# Mechanism 2: Security devices

**Physical** separation: e.g. out-of-band management <4U]

**Access control**: Programmed checks that access is permitted, e.g.

- View Based Access Control in SNMP [73>
- **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.

# Outline

# Security & NM are symbiotic

**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: Device config => control how/if they work

**Security needs configuration management**

- of keys
- Access Control Lists (ACLs) [Q7>
- update configuration of devices to correct bugs that create security weaknesses

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

Same IT people often do both security and NM



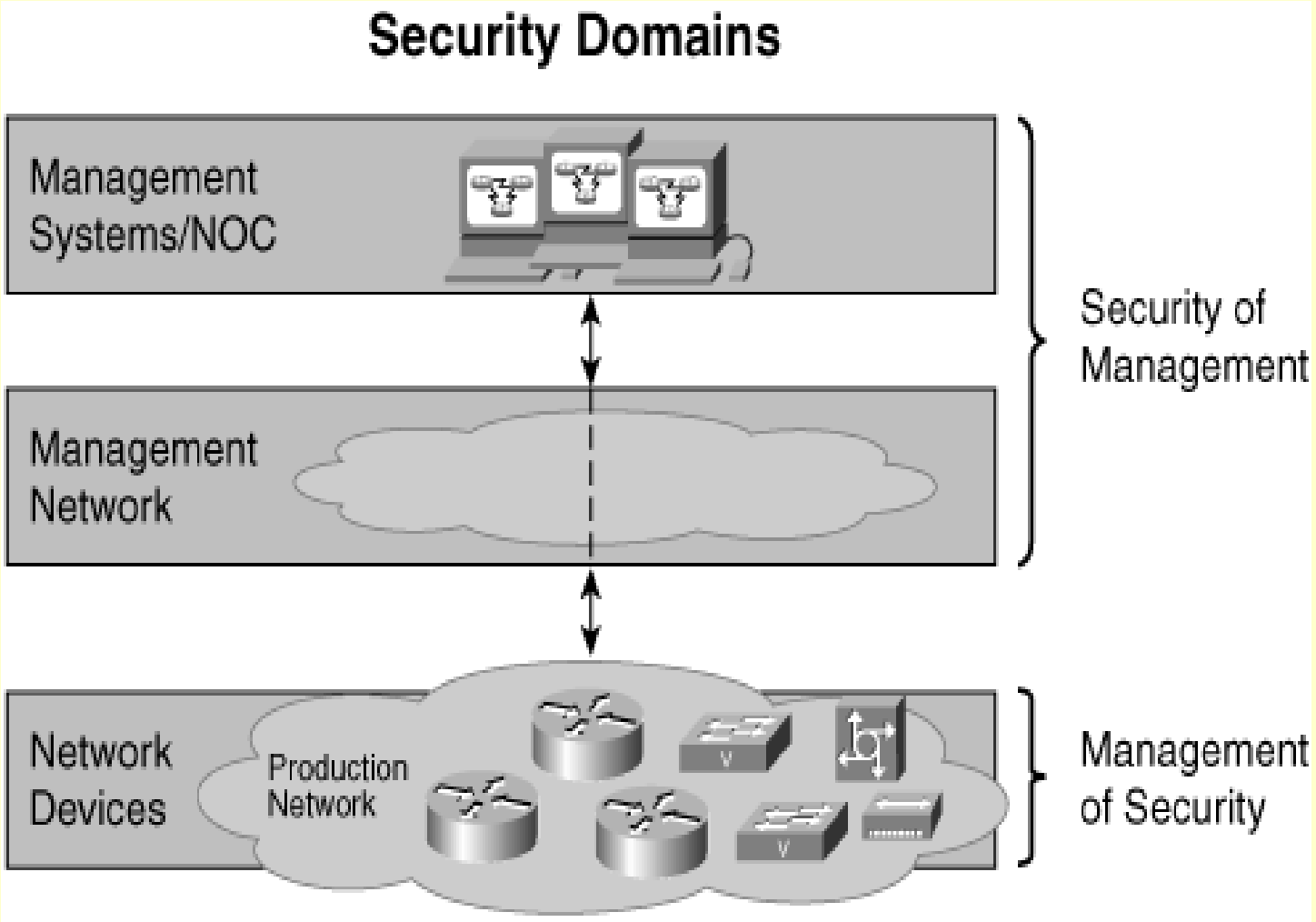


23U

† e.g. Kerberos and X.500

# Security of mgt + Mgt of security

- Need to secure management services (upper Fig)
- Need to manage security services (lower Fig)




Text in figure: NOC
Production network

# Tension between security & NM

- Range of services
  - Minimise for security
    - Exclude those with known vulnerabilities; less to worry about
    - 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

# FCAPS links

**FCAPS topics are interrelated**

e.g. Unavailability due to fault (F) or Denial of Service (S)

**Each topic  will have a slide relating it to other topics**

Avoid duplication by topics only referring to preceding topics => No links for this lecture

**Link slides in future lectures:**

Fault [1U>

Configuration [F9>

Accounting [4R>

Performance [YH>

# Outline

SNMPv3 security

# Protecting SNMP itself

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

- e.g. Agent should **check that incoming SNMP packets 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]: `snmpInBadVersions`, `snmpInBadCommunityNames`, `snmpInASNParseErrs` etc

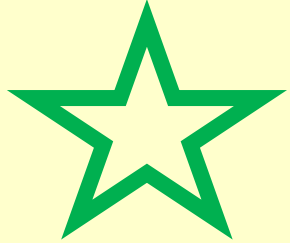
Details of vulnerabilities at http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/





218 Summary at http://www.potaroo.net/ispcol/2002-04/2002-04-snmp.html

# 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)
  - "Security Comes to SNMP: The New SNMPv3 Proposed Internet Standards", *Internet Protocol Journal*, 1(3):2-12†
  - "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?

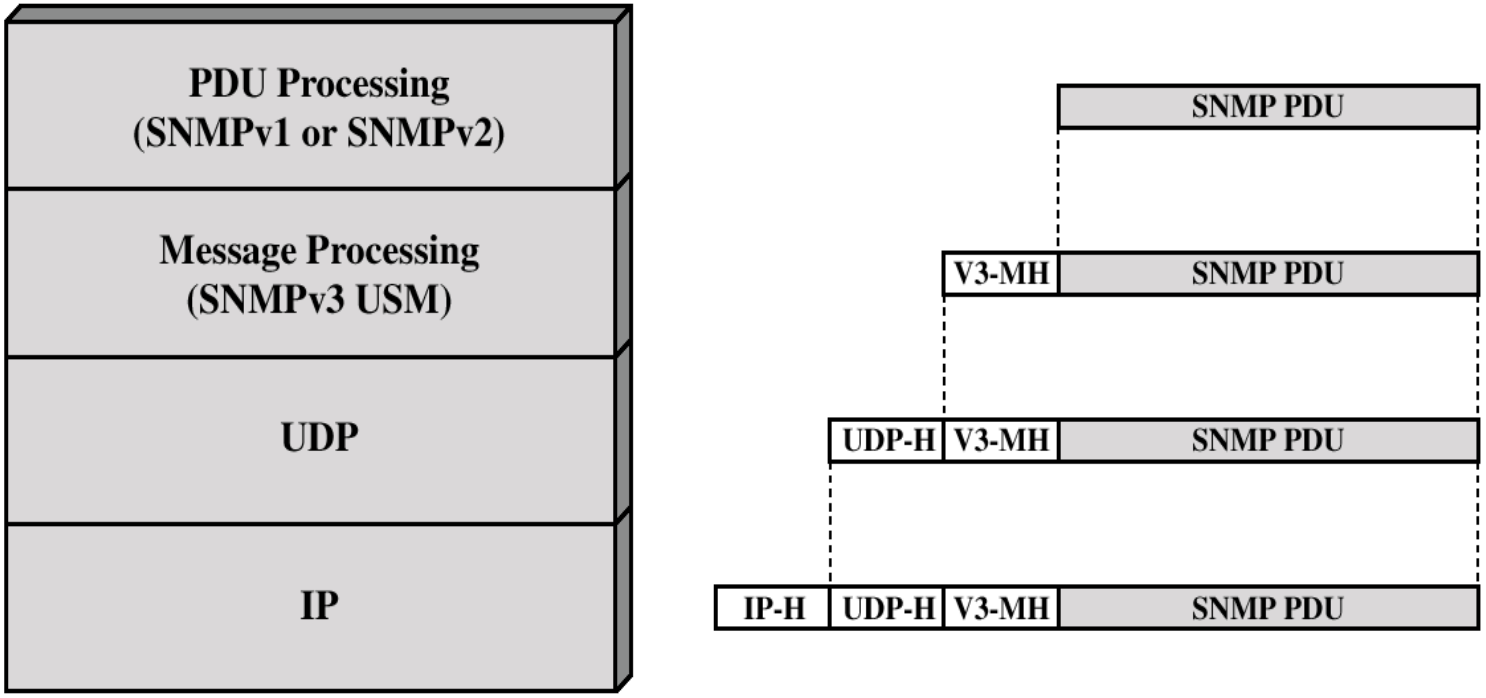
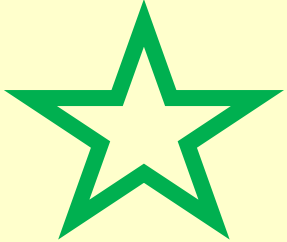



Fig 8.5 from Stallings: *Network Security Essentials*

# 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`

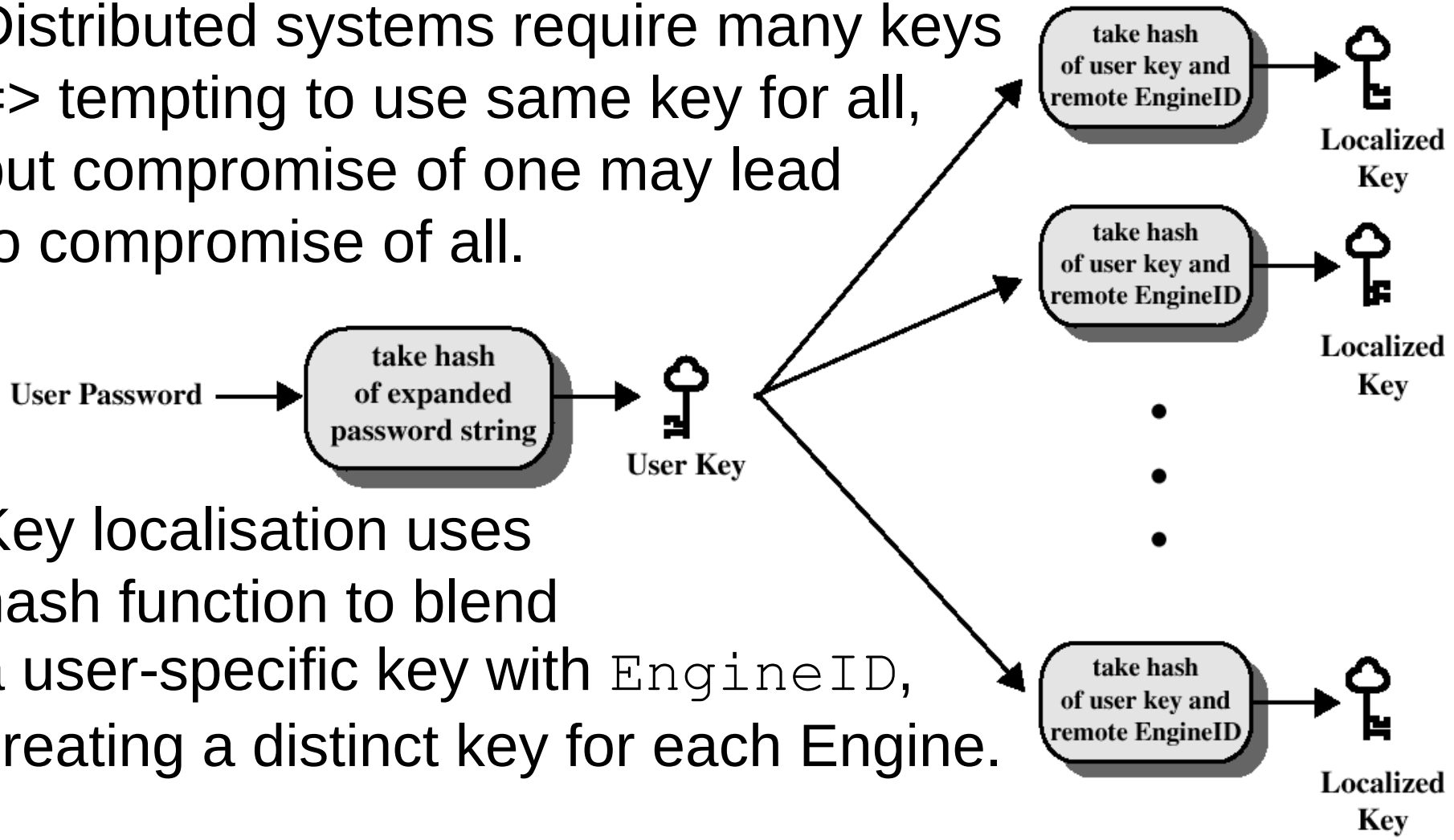
- Suggested max msg size for others to use



† An ASN.1 INTEGER would never wrap. "INTEGER" on this slide is short for the actual RFC text "**INTEGER** (0..2147483647)" and the bound on that INTEGER enables wrapping, albeit after 64 years

# 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"

# Contexts and users

**"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 that are used later [Z0>:

- “users” are specific to the User Security Model; more generic term (also applicable to VACM) is “participant”, which is identified by a “`securityName`”
- `securityName` + `securityModel` => “group” => `groupName`

# 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





252

# 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
}
```

# HeaderData

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

- `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..2147483
     msgAuthoritativeEngineTime   INTEGER (0..2147483
     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*

```
}
```

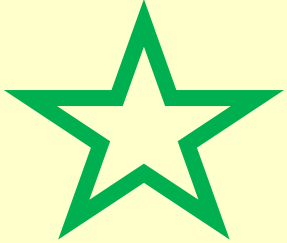
`msgAuthenticationParameters & msgPrivacyParameters` == empty strings if `authFlag` on `privFlag` respectively ==0

# Engine discovery

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

- Send Get 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 Get 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





† Sender could be NMS sending a request to a device, or a device sending response or trap to NMS

# Outline

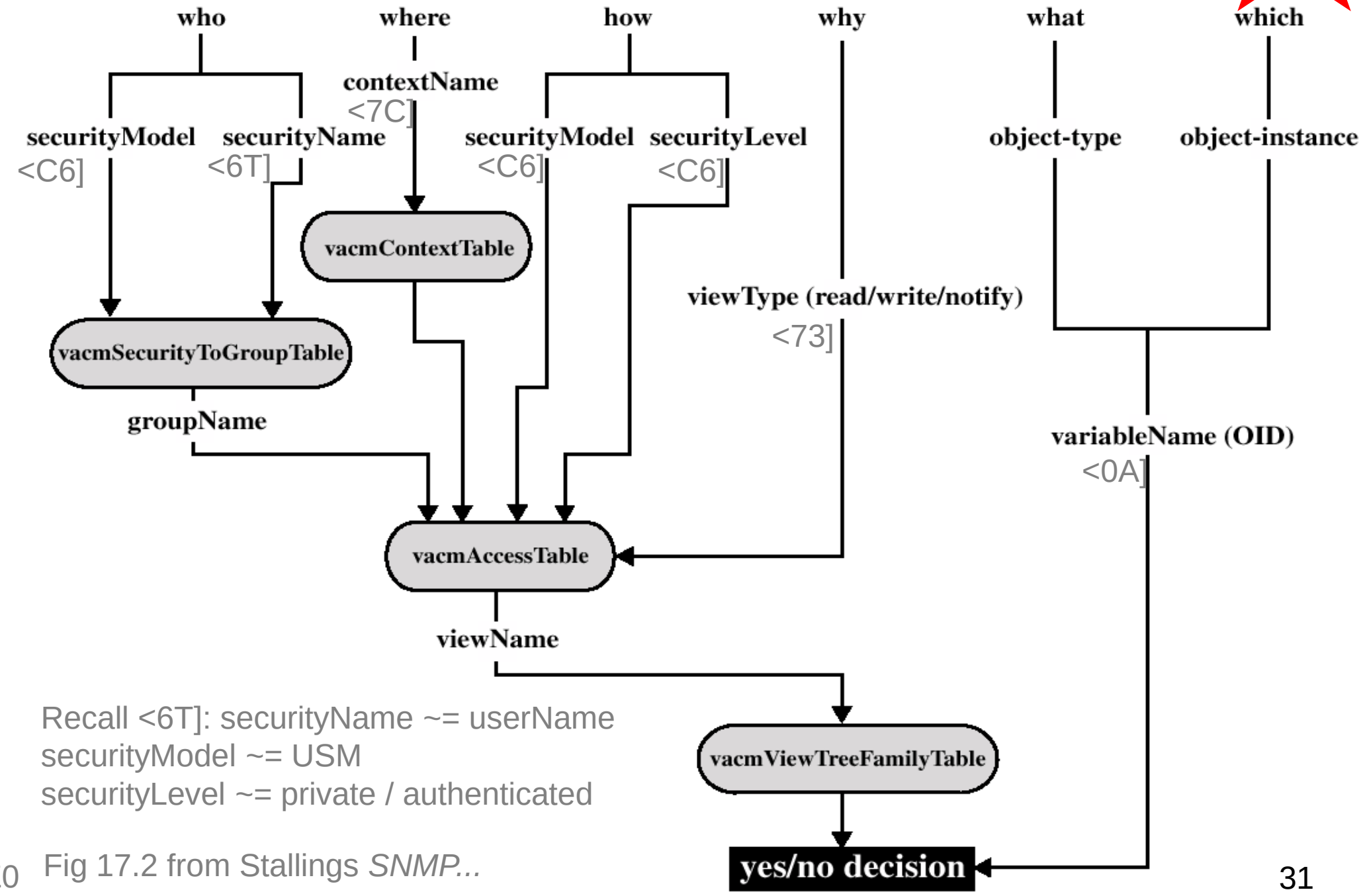
# View-based Access Control Model (VACM)

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

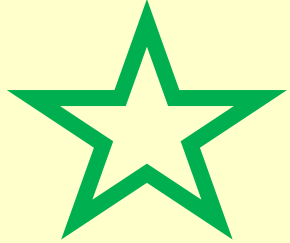
# Pictorial: Factors that affect access




Recall <6T]: securityName ~= userName
securityModel ~= USM
securityLevel ~= 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

- `vacm`**`AccessTable`**
  - indexed by Context, `SecurityLevel`, `groupName` and `SecurityModel`
  - provides 'names' of read, write & notify views
- `vacm`**`ViewTreeFamily`**`Table` 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

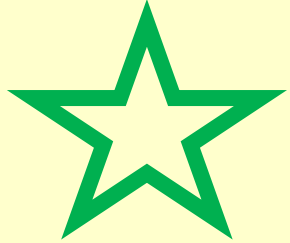
Bonus marks if you can find answers:

- 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.

Previously saw VACM of access to MIB objects. More generally: access to network or other resource.

Specified through rules:

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

Access Control Lists of multiple rules

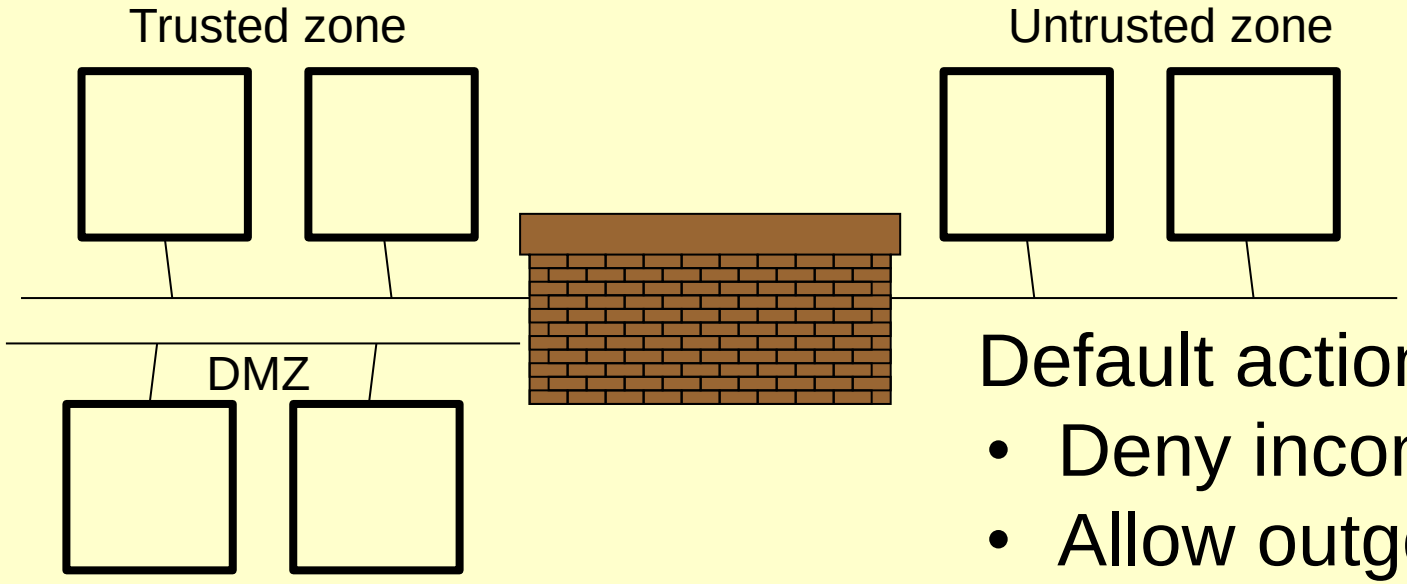
"one recent survey [9 [from 2007]] reported that ISPs called management of access control lists (ACLs) their "most critical" problem." [Bellovin]

-> configuration management [WC>

A. Wool, "A quantitative study of firewall configuration errors," *IEEE Computer*

# 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

Rules apply at each interface

**Policies** apply across all interfaces.

Accept inbound connections only for public servers (web, DNS) in a "demilitarized zone (DMZ)"

# 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 / capture / redirect
  - Especially for serious breaches

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

| | Incoming traffic | Outgoing traffic |
|---|---|---|
| | Log/notify-NM for very bad behaviour | |
| Exceptions: | Deny if malformed | |
| | Allow responses to our requests | Deny bad behavior e.g. SMTP from hosts |
| Default: | Deny | Allow |

# Sample access control list

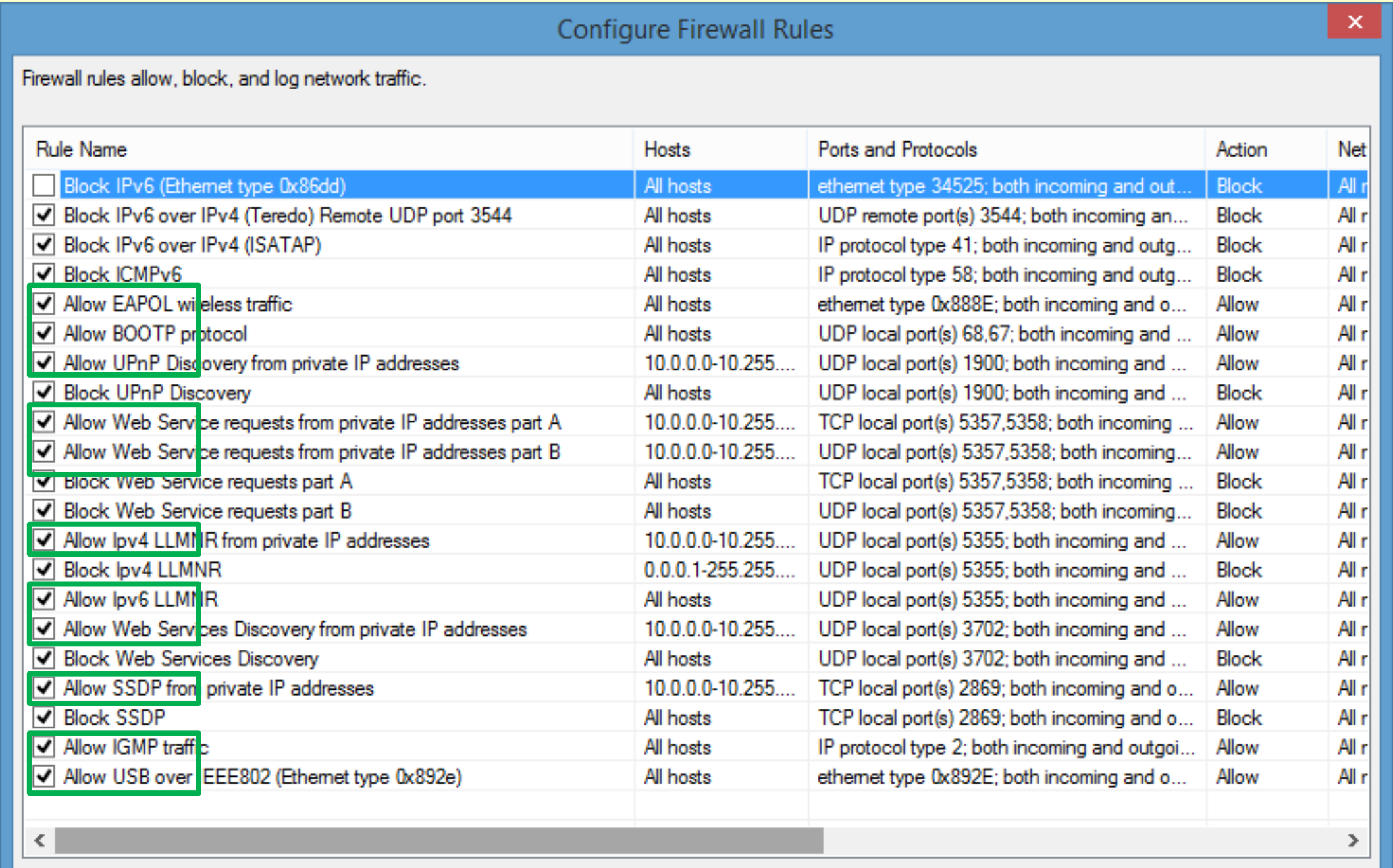
| | IP fields | | | Transport fields | | | State |
|---|---|---|---|---|---|---|---|
| **Action** | **src** | **dst** | **proto** | **src** | **dst** | **flags** | **estab'd** |
| *# Allow outbound DNS requests & corresponding inbound responses* | | | | | | | |
| allow | 149.11/16 | !149.11/16 | UDP | >1023 | 53 | X | X |
| allow | !149.11/16 | 149.11/16 | UDP | 53 | >1023 | X | Y |
| *# Allow our users to connect to external web servers & receive replies* | | | | | | | |
| allow | 149.11/16 | !149.11/16 | TCP | >1023 | 80 | X | X |
| allow | !149.11/16 | 149.11/16 | TCP | 80 | >1023 | ACK=1 | Y |
| *# Block everything else!* | | | | | | | |
| deny | * | * | * | * | * | * | X |

! means not, as in C programming language & descendents

Requiring external address not in our network helps detect internal routing problems.

client port >1023 stops external attempt to connect to well known port

# e.g. default Symantec PC firewall rules



# Things to think about

- Critical thinking:
  - Consider how improve security may impinge usability.
  - Security flaws garner publicity, but how many remedies are proven to themselves be flawless?
- Engineering methods: Perfect security often involves high overhead, so it suffices to just make attacks “computationally infeasible”. In terms of engineering cost/benefit analysis, the attack becomes no longer worthwhile for the attacker.
- Links to other areas: Deliberate security attacks are very similar to failures (e.g. type of cause, effect, defence) except that they need stronger defences.
- Independent learning: A good textbook to learn more about this subject is “Network Security: Private Communications in a Public World”





2!!!

# The end of security