

Trust Management, Compliance Checking & Security Policy

Matt Blaze

AT&T Laboratories

mab@research.att.com

joint work with: Joan Feigenbaum,
John Ioannidis, Angelos Keromytis,
Jack Lacy, and Martin Strauss

What's a security policy for?

- Maybe it *generates* actions, events, etc.
 - e.g., "policy enabled" / "policy driven" whatever
 - policy generates *actual behavior* of system
 - nice idea, but big, hard to be sure about
- Maybe it *protects resources*
 - policy specifies what's allowed
 - security layer checks for compliance with policy
 - prevents things that don't comply from happening
 - smaller problem, focuses security in one place
- Here we focus on the latter

31 January 2001

Trust Management

2

Trust Management

- Nothing to do with finance or faith in boss
- For answering questions of the form: "Should I perform this (dangerous) action?"
- Systematic approach to managing
 - security policies, credentials, trust relationships
- Term coined in 1996
 - Blaze, Feigenbaum, Lacy, "Decentralized Trust Management." Oakland, 1996.

31 January 2001

Trust Management

3

Trust Management: Compliance Checking

- Provides advice to applications on whether "dangerous" actions should be permitted
- Compliance checker uses local policy & signed credentials in making these decisions
 - guarantees that only actions that conform to policy will be approved
- As long as all dangerous actions are checked with the compliance checker, we know the security policy is being followed

31 January 2001

Trust Management

4

Distributed/Decentralized Policy

- Ideally, the policy is in one place, specified by one person
- More often, different parts of the policy come from different places
 - delegation of authorization
 - different administrators for different services
 - multiple requirements for access
- You may not even be able to look at the whole policy in one place
- Scale here means complexity & distribution

31 January 2001

Trust Management

5

Policies and credentials do similar things

- A *policy* tells *who* is trusted to do *what*
 - *who* might be a public key
 - *what* is some potentially "dangerous" action
 - spend money, claim to be "matt blaze", access a document
- A *credential* delegates trust to *someone else*
 - *someone else* might also be a public key (e.g., a CA)
- Distributed systems blur the line between policies and credentials
 - a credential is a policy signed by someone trusted

31 January 2001

Trust Management

6

The Big Lie: Public Key Infrastructure

- Why don't certificates and PKIs solve everything?
 - applications want an answer to this question:
 - "is this the correct public key for this purpose?"
 - current applications need ad hoc mechanism
 - PKI systems quietly restate this by answering another question instead:
 - "who owns this public key?"
 - X.509 certificates are good at doing this
- The two questions aren't quite the same...

31 January 2001

Trust Management

7

Trust Management Theology

- Separate mechanism from policy
 - application-specific data, general mechanisms
 - certificate-based systems get this backwards!
- Use a general language for writing application-specific policies and credentials
- Interpreter for this language can serve as a compliance checker that applications call to test whether an action is allowed based on policy & credentials

31 January 2001

Trust Management

8

Trust Management Elements

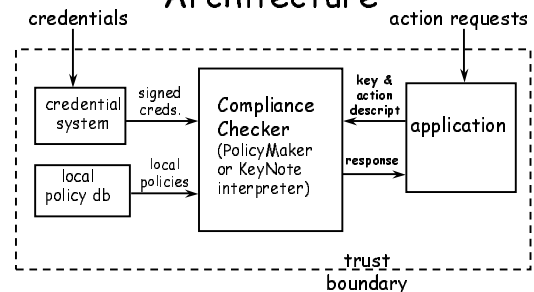
- A language for *Actions*
 - operations with security consequences for applications
- A naming scheme for *Principals*
 - entities that can be authorized to request actions
- A language for *Policies*
 - govern the actions that principals are authorized for
- A language for *Credentials*
 - allow principals to delegate authorization
- A *Compliance Checker* and interface
 - service that determines whether a requested action should be allowed, based on policy and a set of credentials

31 January 2001

Trust Management

9

Trust Management Architecture



31 January 2001

Trust Management

10

Trust Management Languages

- PolicyMaker
 - Blaze, Feigenbaum, Lacy, 1996
 - Compliance checking semantics formalized in Blaze, Feigenbaum, Strauss, 1998
 - very general, designed more for study than use
- KeyNote
 - Blaze, Feigenbaum, Ioannidis, Keromytis 1997
 - defined in RFC 2704
 - designed to be used, especially in Internet apps
- Both share same basic semantic structure

31 January 2001

Trust Management

11

Basic Language Element: Assertions

- Authorize principals to perform actions
- Policies consist of a collection of assertions
- An assertion contains two basic parts
 - a principal identifier (key or key expression)
 - an action predicate
- principal is authorized to perform actions that pass the action predicate
- Assertions can also be signed by keys
 - signed assertions are credentials

31 January 2001

Trust Management

12

Assertion Syntax

- *principal is-authorized-for predicate* {signed-by authorizer}
 - the keys in *principal* are authorized for actions that pass the predicate, according to authorizer
- Actual KeyNote assertion syntax:
 - Authorizer:** <keyword POLICY or assertion signer's public key>
 - Licensees:** <principal, tests signer keys>
 - Conditions:** <trust-expr, tests action attribs>
 - Signature:** <encoding of signature, for signed credentials>

31 January 2001

Trust Management

13

Assertion Examples

- Policies (local and trusted)
 - *angelos* is allowed to authorize transactions up to \$10,000
 - *ji* is allowed to authorize email from **@research.att.com*
- Credentials (signed by someone):
 - *jisays* that *mab* is allowed to authorize email from *"mab@research.att.com"*
 - *angelos* says that *mab* together with *ji* is allowed to authorize transaction up to \$5000
- *mab, ji, angelos* can be given as public keys

31 January 2001

Trust Management

14

Describing Actions

- Action semantics are unknown by KeyNote itself
 - interpreted only by assertion predicates
- Set of freeform attribute/value pairs
- Associated with a key that describes who is requesting the action
- Attribute/values are available to and evaluated by assertion predicates

31 January 2001

Trust Management

15

Compliance Checking Semantics

- An action is approved if any policy assertion approves it
- An assertion is considered to approve an action if its predicate passes and either
 - the action was directly requested by the assertion's licensees
 - the action was approved by some other assertion signed by the licensees

31 January 2001

Trust Management

16

Compliance Checking Process

- Application collects appropriate assertions
 - local, trusted root policy assertions
 - credentials signed by someone else
- Application forms action description
 - collection of free-form attributes
 - associated with principal identifier (key or ID)
- Compliance checker finds "compliance value"
 - evaluates actions against conditions in assertions forming graph between root policy and requestor
 - binary: approved, disapproved, or multi-valued

31 January 2001

Trust Management

17

Assertion Monotonicity

- Adding an assertion can never cause an approved action to become disapproved
- Deleting an assertion will never cause a disapproved action to become approved
- Nothing is allowed unless a policy assertion explicitly allows it

31 January 2001

Trust Management

18

Implications of Monotonicity

- Safe for distributed systems
 - missing assertions can't cause policy violations
- The set of approving assertions constitutes "proof of compliance"
 - client can collect appropriate signed assertions and present them to server
- No such thing as a "conflict"
 - if an action can be approved, it's approved

31 January 2001

Trust Management

19

Limitations of Monotonicity

- Some policies don't seem monotonic
 - everything is allowed unless I say otherwise
 - don't accept email from angelos
- Revocation is non-monotonic
 - but a revocation service need not be...
- In practice, many non-monotonic policies can be specified as monotonic
 - the cost of systems to support non-monotonic evaluation is so high that it's worth avoiding

31 January 2001

Trust Management

20

Some (Real) Example Applications

- Network-layer security (IPSEC)
- Offline electronic micropayments

Network Layer Security (IPSEC)

- Protects (encrypts / authenticates) at network layer
- Usually based on *datagram encapsulation*
 - IPSEC is an example
 - Packets are protected and put inside packets
 - On receipt, protected packets are extracted, authenticated, decrypted, then processed normally
- Extremely versatile
 - Any network node (host) can be a security endpoint
 - Many configurations (end-end, link, net-net)
 - Basic building block for VPNs

31 January 2001

Trust Management

22

Security Associations in IPSEC

- The *Security Association (SA)* is the basic data structure describing how to handle secure packets
 - indexes crypto keys
 - defines classes of packets to which it applies
 - establishes an IPSEC "channel"
- Incoming packets contain SA identifier
 - whether to accept a packet is a policy decision
- Outgoing packets protected under SA selected by host according to policy

31 January 2001

Trust Management

23

Policy in Network-Layer Security

- A host's *policy* determines whether traffic can be exchanged with other hosts and networks
- Governs:
 - With which other hosts data is exchanged
 - is there a key to encrypt/authenticate this packet?
 - What kind of data can be exchanged
 - does sending or receiving this data from this host conform to security policy?

31 January 2001

Trust Management

24

Network Policy is Really a Hierarchy of Policies

- Packet Policy: how to treat each packet?
 - should packet be encrypted/sent/accepted?
 - enforced by packet filter associated with SA
- SA Policy: what filter to associate with SA?
 - should filter be applied to an SA?
 - no standard way to determine this
- Meta-Policy: does policy comply with "policy"
 - organization policy may be high-level, informal, ill-specified

31 January 2001

Trust Management

25

Some Observations

- Packet-level policy must be fast
 - enforced on every incoming / outgoing packet
 - filter limited to header pattern matching
- SA policy can be a bit slower
 - enforced when SA is created, (e.g., key xchg)
 - might be complex, based on credentials and dynamic relationships
 - this is a classic *trust management problem*

31 January 2001

Trust Management

26

An IPSEC Trust Management Architecture

- IPSEC SA contains a simple packet filter
 - pattern matches input & output packet header
 - faster than the encryption operation
- SA creation controlled by KeyNote
 - Action describes filter
 - KeyNote credentials serve as certificates, allow remote policy management
 - Fast enough to evaluate as part of key exchange

31 January 2001

Trust Management

27

Example KeyNote Policy

```
Authorizer: "POLICY"  
Licencees: "DSA:1f203faa2babd11ffe"  
Conditions: application=="network"  
            && (protocol=="tcp" || protocol=="udp")  
            && source == "192.11.255.255"  
            && dest == "135.205.89.255";
```

31 January 2001

Trust Management

28

KeyNote Delegation

```
Authorizer: "POLICY"  
Licencees: "DSA:1f203faa2babd11ffe"  
Conditions: application=="network"  
            && (protocol=="tcp" || protocol=="udp");
```

```
Authorizer: "DSA:1f203faa2babd11ffe"  
Licencees: "DSA:23dd11ff12efcafef"  
Conditions: source == "192.11.255.255"  
            && dest == "135.205.89.255";  
Signature: "093a3134ffa38172200333110a2bc"
```

31 January 2001

Trust Management

29

Benefits

- Extensible
 - policy not hard-coded
- Human-readable
- Expressible
 - can represent VPNs, enforce secure DNS, remote access, access control lists, etc.
- Can serve as basis for comprehensive security policy at other layers
- Serves as intermediate layer between organization policy and packet filter implementation

31 January 2001

Trust Management

30

Another Example: KeyNote Microchecks

- Basic idea is to use trust management system to encode risk management strategy
- Customers issued short-lived KeyNote credential that describes the circumstances under which offline payments are guaranteed
 - e.g., newspapers and sodas costing less than \$1.50, no more than \$5.00 per vendor per day

Status

- IETF RFC defines the KeyNote language
RFC 2704
- No intellectual property issues
- We have a free implementation of KeyNote:
<http://www.crypto.com/trustmgt>
- The KeyNote group has a mail alias:
keynote@research.att.com
- There's also a mailing list
- These slides can be found in
<http://www.crypto.com/talks/>

Open Problems

- Policy based on provable properties
- Credential optimization
- Policy and credential distribution
- Interaction between security layers
- Policy discovery and negotiation
-