Path-based Access Control for Enterprise Networks

Matthew Burnside and Angelos D. Keromytis
Columbia University

ISC 2008
9/16/08

Overview

- Motivation
  - Access control policy mechanisms in current usage are flawed
- Goal
  - New paradigm for enterprise-scale security policies

Organization

- Background and problem
- Solution 1: Graph-based
- Solution 2: KeyNote
- Evaluation
- Conclusion

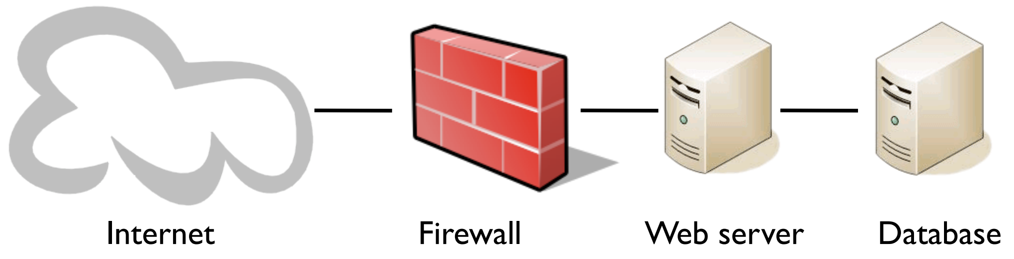
Access control history

- Formalized by Lampson
  1. User makes a request
  2. Access-control mechanism consults security policy
  3. Makes decision
  4. Goes inactive
- Gatekeeper model

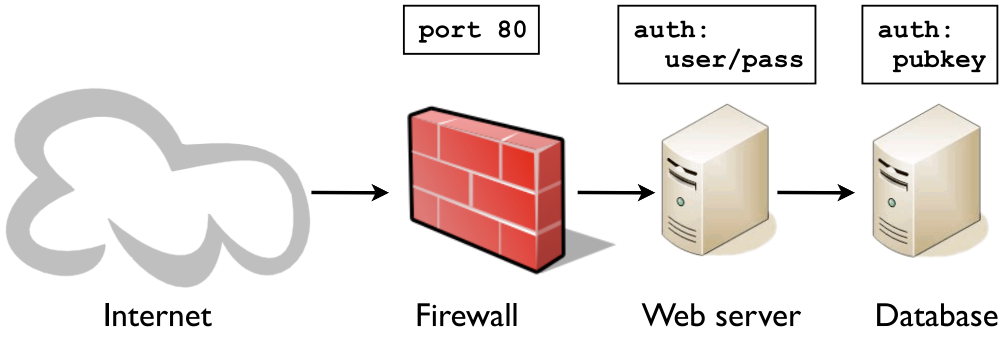
Enterprise-scale policy

- PolicyMaker takes a unified approach to describing policies and trust relationships.
- STRONGMAN showed how to scale policy distribution.
    - *Neither considers dynamic interactions.*

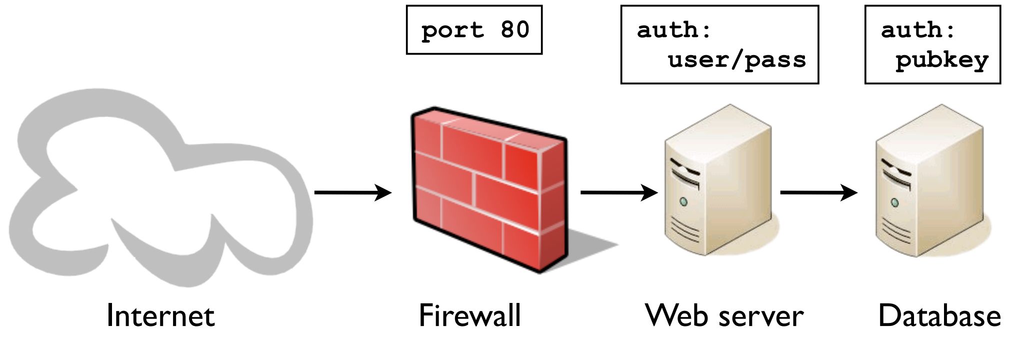
A simple network



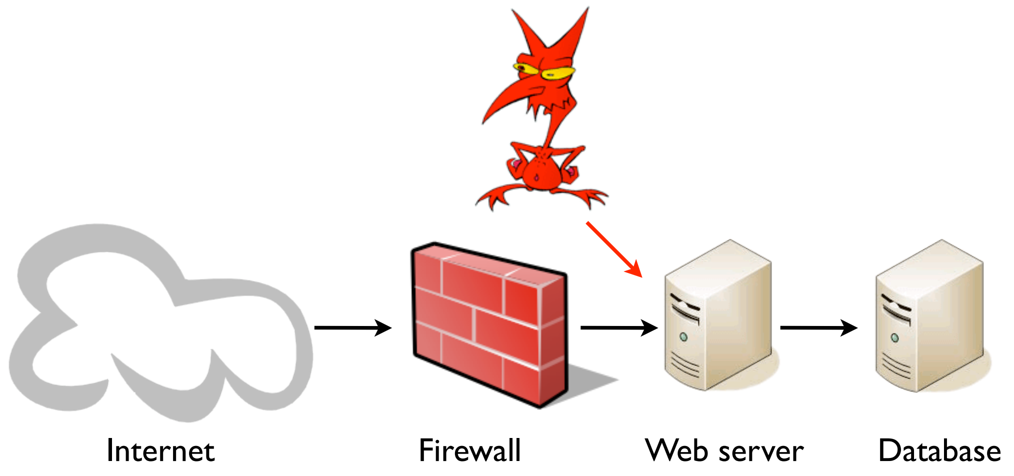
A simple policy



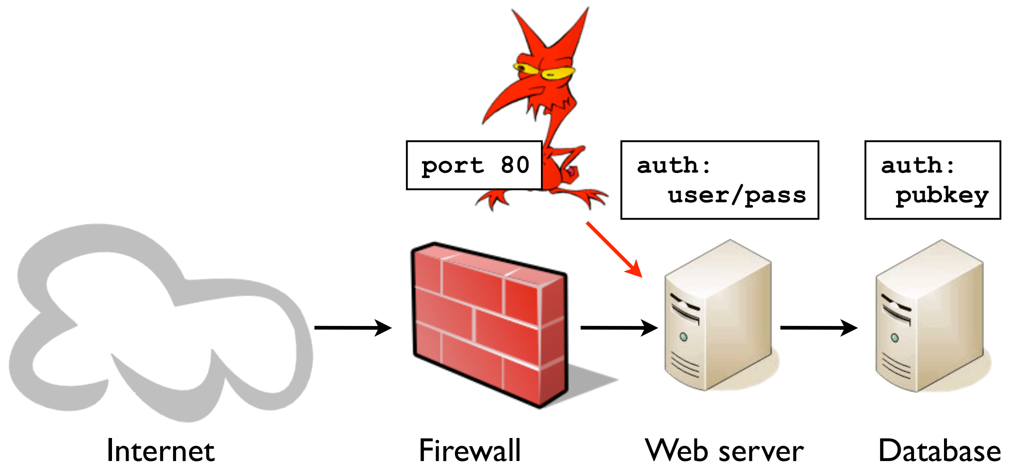
Global policy violation



Global policy violation



Global policy violation



A flawed model

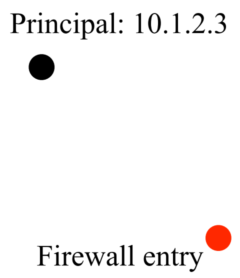
- Attack violates sysadmin’s initial assumptions about the network.
- *Insight*: global policy enforcement requires dynamic interaction between access control components.

Solution 1: Graph-based

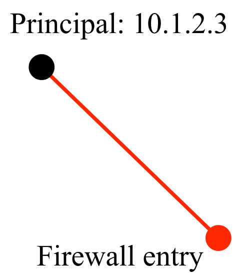
- Model network requests like function call graphs
- Define policies as paths through the graphs

Example session

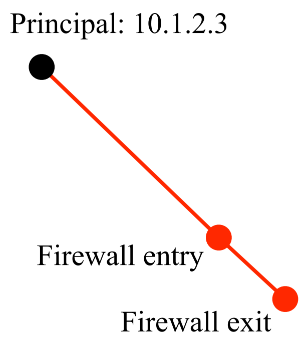
Example session



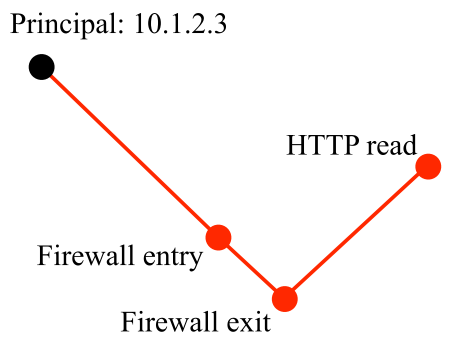
Example session



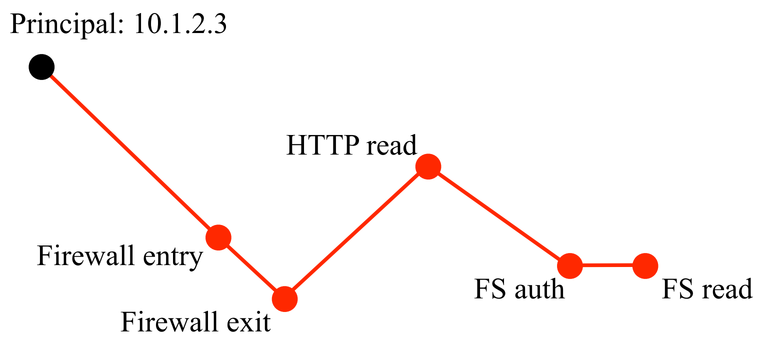
Example session



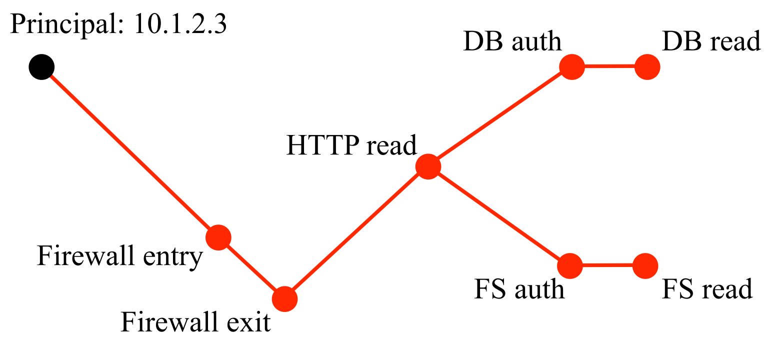
Example session



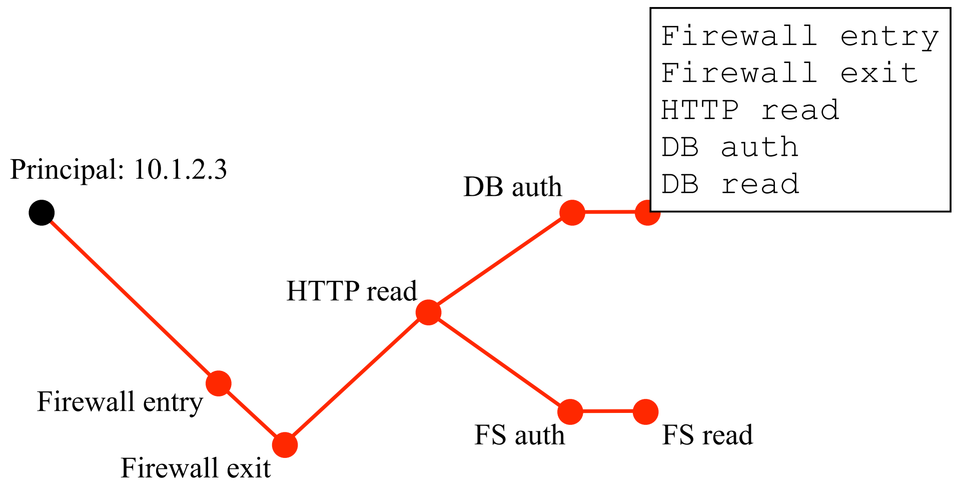
Example session



Example session



Defining a policy



```
Firewall entry
Firewall exit
HTTP read
DB auth
DB read
```

Solution 2: KeyNote-based

- Model network requests like function call graphs
- Define policies as *certificate chains* representing paths through the graphs
- Prevents an adversary from modifying the inherited chain.

KeyNote overview

- Five components (Defined in RFC2704)
  - Actions: operations with security consequences
  - Principals
  - Policy language
  - Credentials: allow principals to delegate authorization to other principals
  - Compliance checker: return yay or nay (policy compliance value), given a requested action, a policy, and a set of credentials

Policy assertions

```
KeyNote-Version: 2
Local-Constants: Alice="RSA:a8ce1212"
Authorizer: "POLICY"
Licensees: Alice
Conditions: (app_domain=="FTP") && (@size < 1GB);
```

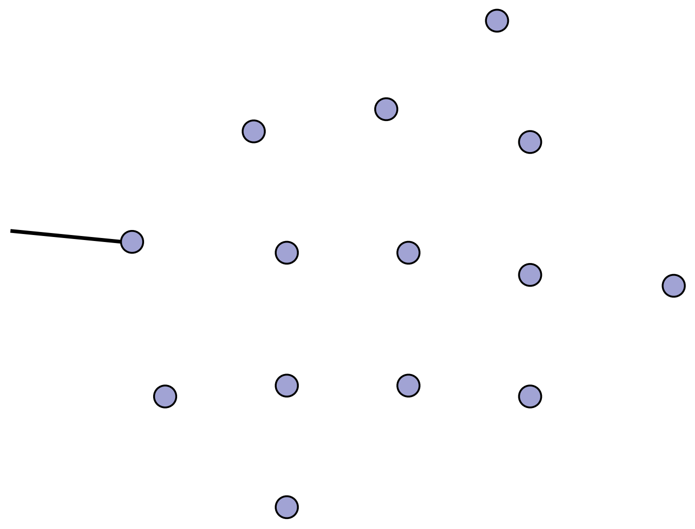
- Policies and credentials are called assertions
- A special principal, called POLICY, is the root of trust

Credential assertions

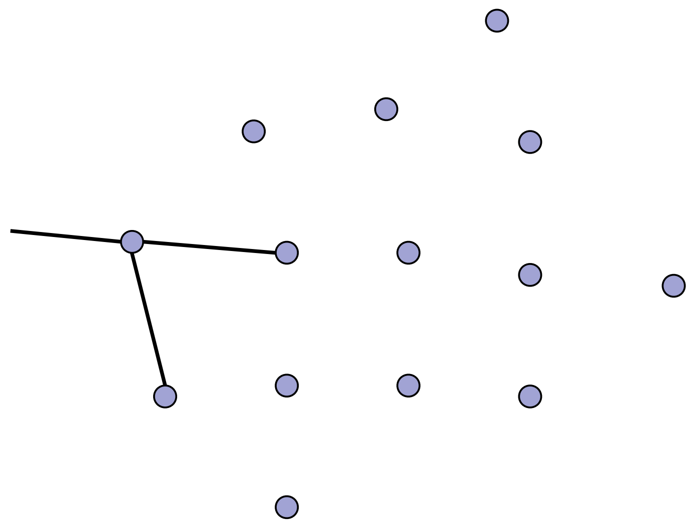
```
KeyNote-Version: 2
Local-Constants: Alice="RSA:a8ce1212"
                 Bob="RSA:8787fefe"
Authorizer: Alice
Licensees: Bob
Conditions: (app_domain == "FTP") &&
            (address == "cs.columbia.edu");
Signature: "RSA-SHA1:a1a2b3b4"
```

- Allows delegation of trust from principal to principal

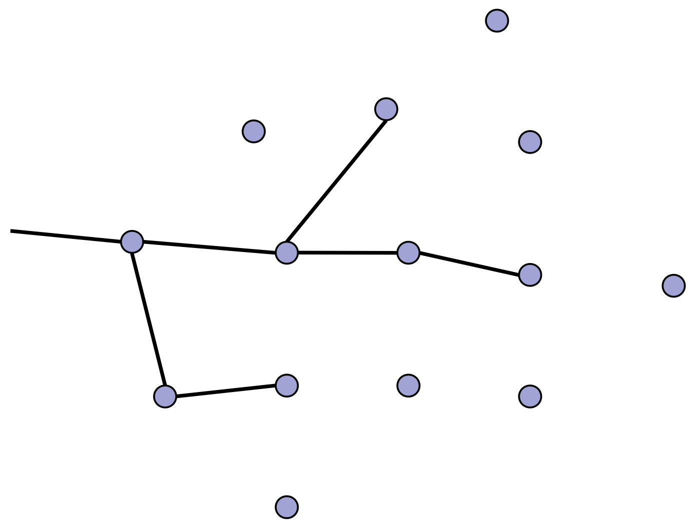
Path-based access control



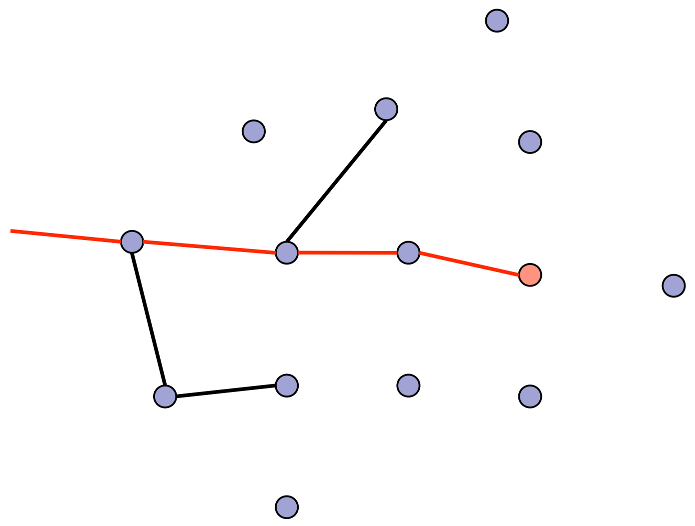
Path-based access control



Path-based access control



Path-based access control

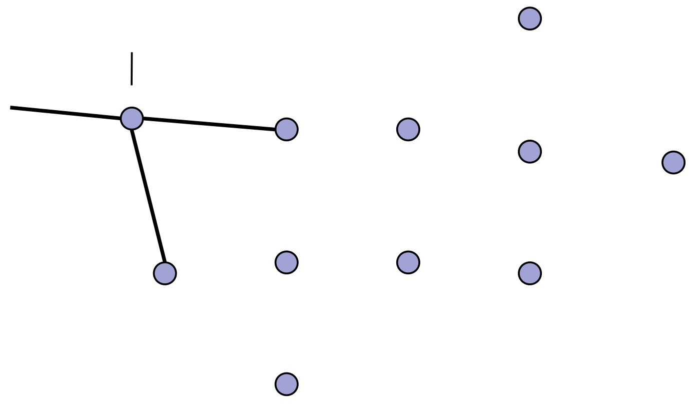


Events are assertions

```
KeyNote-Version: 2
Comment: Forward request to web server
Local-Constants: FW_key = "RSA:acdfa1df"
                 WEB_key = "RSA:deadbeef"
Authorizer: FW_key
Licensees: WEB_key
Signature: "RSA-SHA1:f00f2244"
Conditions: …
```

Generating an event

```
KeyNote-Version: 2
Comment: Forward request to web server
Local-Constants: FW_key = "RSA:acdfa1df"
                 WEB_key = "RSA:deadbeef"
Authorizer: FW_key
Licensees: WEB_key
Signature: "RSA-SHA1:f00f2244"
Conditions: …
```



Building the assertion path

- The request propagates through the network, and correlation sensors generate assertions.
- Each assertion is forwarded to the next hop along with the request.

Building the assertion path

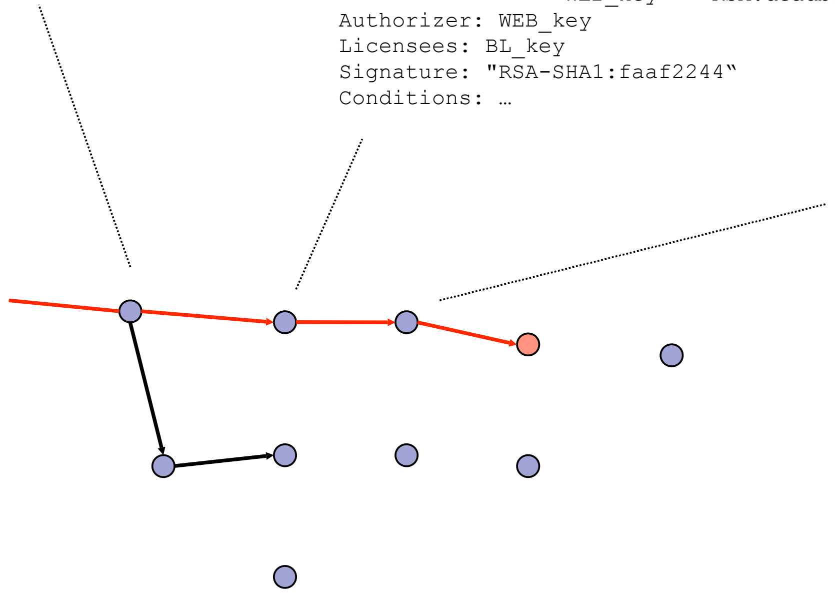
- Assertion set forms a certificate chain from the entry point to the receiving node!

Example chain

```
KeyNote-Version: 2
Comment: Forward request to web server
Local-Constants: FW_key = "RSA:acdfa1df"
                 WEB_key = "RSA:deadbeef"
Authorizer: FW_key
Licensees: WEB_key
Signature: "RSA-SHA1:f00f2244"
Conditions: …
```

```
KeyNote-Version: 2
Comment: Web server to business logic
Local-Constants: BL_key = "RSA:1111a1df"
                 WEB_key = "RSA:deadbeef"
Authorizer: WEB_key
Licensees: BL_key
Signature: "RSA-SHA1:faaf2244"
Conditions: …
```

```
KeyNote-Version: 2
Comment: Forward request to DB
Local-Constants: BL_key = "RSA:1111a1df"
                 DB_key = "RSA:feeffeef"
Authorizer: BL_key
Licensees: DB_key
Signature: "RSA-SHA1:abab2244"
Conditions: …
```



Policy evaluation

- Leverage the KeyNote compliance checker
  - Is the chain complete?
  - Is the chain correct?

- KeyNote compliance checker returns yay or nay.

Evaluation

Mechanism	**Transfer time**	**Overhead**	**Overhead/ node**
Vanilla	162ms	-	-
Graph	317ms	155ms	52ms
KeyNote	1120ms	958ms	319ms

Request a 1M file, averaged over 25 trials, across a 3-node network.

Conclusion

- Enhance the current access control paradigm to protect against a new class of attacks.
- Any questions?