Fail-Secure Access Control

Petar Tsankov, Srdjan Marinovic, Mohammad Torabi Dashti, David Basin

Institute of Information Security
ETH Zurich
All access control systems must operate in the presence of failures.
System Model

Subjects → PDP (Policy Decision Point) → Resources
System Model

Subjects

Submit *requests* and credentials

PDP

Policy Decision Point

Resources
System Model

Subjects submit requests and credentials. The Policy Decision Point (PDP) evaluates a declarative policy: RBAC, JAAS.
System Model

- Database
- Time server
- Policy Information Points

Subjects
- Submit requests and credentials

PDP
- Policy Decision Point

Evaluates a declarative policy:
- RBAC, JAAS
System Model

Evaluate a declarative policy:
- RBAC, JAAS

Subjects submit requests and credentials.

Policy Decision Point evaluates a declarative policy:
- RBAC, JAAS

Possible failures:
- Database
- Time server

Policy Information Points
The Problem with Failure Handling

Access control requirements

Declarative policy

PDP

* valid

* invalid
The Problem with Failure Handling

Policy Analysis

Access control requirements

Declarative policy

PDP

✓ ✔ ×
The Problem with Failure Handling

Policy Analysis

Access control requirements

Declarative policy

Affects how access decisions are made

PDP

Failure handling
The Problem with Failure Handling

Policy Analysis

Access control requirements

Declarative policy

Fail-security requirements

Describe how failures should be handed

Are all requirements met?

PDP

Failure handling

Describe how failures should be handled
The Problem with Failure Handling

Policy Analysis

Access control requirements + Declarative policy + failure handling

Fail-security requirements

PDP

Failure handling

derive
The Problem with Failure Handling

Policy Analysis

Access control requirements

Declarative policy

Fail-Security Analysis

Fail-security requirements

failure handling

derive

PDP

Failure handling

✓ ✗
Contributions

- Defining attacker model
- Specifying access policy with failure handling
- Verifying fail-security requirements
Attacker Model
System and Attacker Model

- Subjects
- Database
- Time server
- PDP
- Resources

Policy Information Points

remote queries
The attacker can selectively cause any remote query to fail.
Contributions

- Defining attacker model
- Specifying access policy with failure handling
- Verifying fail-security requirements
Running Example
Running Example

Authorizations in Grids

Owner → Bob → Fred → Valid delegation
Running Example

Authorizations in Grids

Owner → Bob → Fred

→ Valid delegation
→ Revoked delegation
Running Example

Authorizations in Grids

Owner -> Bob -> Fred

- - - - - 

Valid delegation
Revoked delegation
The PDP fails to check if the delegation is revoked
Running Example

Authorizations in Grids

Owner → Bob → Fred

- - - - Valid delegation
- - - - Revoked delegation
- - - - The PDP fails to check if the delegation is revoked

Fail-security requirement

If the PDP fails to check whether a delegation chain is valid, grant access iff the subject is a direct delegate
Running Example

Authorizations in Grids

Owner → Bob → Fred

- → Valid delegation
- → Revoked delegation
- → The PDP fails to check if the delegation is revoked

Fail-security requirement

If the PDP fails to check whether a delegation chain is valid, grant access iff the subject is a direct delegate.
Building a Grid PDP

The **policy** and **the failure handling** are split

Policy

PDP with failure handling
Building a Grid PDP

The policy and the failure handling are split

Policy

auth(X) :- owner(X)
auth(X) :- auth(Y), del(Y,X)

PDP with failure handling

auth(User user, List delegations):
policyEngine.add(Policy)
for del in delegations:
  try:
    if server.isNotRevoked(del):
      policyEngine.add(del)
  catch (Exception e):
    if del.issuer == owner:
      policyEngine.add(del)
return policyEngine.auth(user)
Building a Grid PDP

The **policy** and the **failure handling** are split

**Policy**

- auth(X) :- owner(X)
- auth(X) :- auth(Y), del(Y,X)

**PDP with failure handling**

- auth(User user, List delegations):
  - policyEngine.add(\textit{Policy})
  - for del in delegations:
    - try:
      - if server.isNotRevoked(del):
        - policyEngine.add(del)
    - catch (Exception e):
      - if del.issuer == owner:
        - policyEngine.add(del)
  - return policyEngine.auth(user)

Is this failure handler correct?

Does the PDP (**policy** + **failure handling**) meet the **fail-security requirement**?
Specifying Access Control With Failure Handling
PDP Specification

- **Declarative policy** ➔ e.g. RBAC, ACLs
- **Failure-handling model** ➔ e.g. documentation

PDP specification
PDP Specification

**Declarative policy** ➔ e.g. RBAC, ACLs

**Failure-handling model** ➔ e.g. documentation

How to write this?

PDP specification
BelLog

A many-valued logic-programming language

Can encode the state-of-the-art policy languages

Can encode failure handling:

➔ **Denote failures with**

➔ Derive failure-handling operators

Truth space

Truth ordering

\[t\] \[\perp\] \[f\]
BelLog

A many-valued logic-programming language

Can encode the state-of-the-art policy languages

Can encode failure handling:

- Denote failures with
- Derive failure-handling operators

Truth space

Truth ordering

Represent failures
Specifying the Running Example

**Policy**

\[
\text{pol}(X) :- \text{owner}(X) \\
\text{pol}(X) :- \text{pol}(Y), \text{valid-del}(Y, X)
\]

\[
\text{pol}(X) \leftarrow \text{owner}(X) \\
\text{pol}(X) \leftarrow \text{pol}(Y) \land \text{valid-del}(Y, X)
\]

to BelLog

**Failure handling**

```
try:
    ...
catch (Exception e)
    if del.issuer == owner:
        policyEngine.add(del)
```

to BelLog

\[
\text{valid-del}(X, Y) \leftarrow (\text{del}(X, Y) \triangleright \text{owner}(X))
\]

**Common failure-handling idioms:**

➔ Catch  ➔ Fallback  ➔ Propagate
Verifying Fail-Security Requirements
Verifying Fail-Security Properties

Analyze all failure scenarios that the attacker can cause.

Analyzer

PDP specification
Fail-security property

Secure
Attack-scenario
Verifying Fail-Security Properties

How to write this?

Analyze all failure scenarios that the attacker can cause
Fail-Security Properties

Fail-security requirement

If the PDP fails to check whether a delegation chain is valid, grant access iff the subject is a direct delegate.
Fail-Security Properties

**Fail-security requirement**

If the PDP fails to check whether a delegation chain is valid, grant access iff the subject is a direct delegate

\[ \varphi \rightarrow (S_{pdp} \Leftrightarrow S) \]
Fail-Security Properties

Fail-security requirement

If the PDP fails to check whether a delegation chain is valid, grant access iff the subject is a direct delegate.

A condition that defines failure-scenarios

\[ \varphi \rightarrow (S_{pdp} \iff S) \]
Fail-Security Properties

Fail-security requirement

If the PDP fails to check whether a delegation chain is valid, grant access iff the subject is a direct delegate.

A condition that defines failure-scenarios

\[ \varphi \rightarrow (S_{pdp} \iff S) \]

PDP specification
Fail-Security Properties

Fail-security requirement

If the PDP fails to check whether a delegation chain is valid, grant access iff the subject is a direct delegate.

A condition that defines failure-scenarios

PDP behavior described by the fail-security requirement

\[ \varphi \rightarrow (S_{pdp} \Leftrightarrow S) \]
Running Example Attack

Valid delegation
Revoked delegation
The PDP failed to check if the delegation is revoked
Running Example Attack

The PDP **fails** to check this delegation

The PDP **successfully** checks this delegation

Fred

Valid delegation
Revoked delegation
The PDP failed to check if the delegation is revoked
Running Example Attack

The PDP **fails** to check this delegation

The PDP **successfully** checks this delegation

try:
    ...
    catch (Exception e)
    if del.issuer == owner:
        policyEngine.add(del)
What I Skipped Today

- Failure-handling idioms
- Algorithmic complexity
- Tools:
  - PDP Simulator (http://bellog.org)
  - PDP Analyzer (http://goo.gl/JzKKxk)
- More examples
Summary of Contributions
Summary of Contributions

Raised the problem of failure handling in access control
Summary of Contributions

Raised the problem of failure handling in access control

Defined an attacker that can selectively cause failures
Summary of Contributions

- Raised the problem of failure handling in access control
- Defined an attacker that can selectively cause failures
- Defined idioms for specifying PDPs with failure handling
Summary of Contributions

- Raised the problem of failure handling in access control
- Defined an attacker that can selectively cause failures
- Defined idioms for specifying PDPs with failure handling
- Developed tools for automated analysis of fail-security properties