An Authorization Logic for Reasoning with Explicit Time

Henry DeYoung
Advised by Frank Pfenning

Senior Thesis Program
Department of Computer Science
Carnegie Mellon University

Thesis Mid-semester Progress, Fall 2007

Outline

Outline

Outline

Outline

A Need for Specification of Access Control Policies

Problem:

- How can we specify and enforce precise access control policies?

A Need for Specification of Access Control Policies

Problem:

- How can we specify and enforce precise access control policies?

Example:

- A browser extension for automatically completing web registration forms needs to access the user's e-mail address, a piece of personal data.

A Need for Specification of Access Control Policies

Problem:

- How can we specify and enforce precise access control policies?

Example:

- A browser extension for automatically completing web registration forms needs to access the user's e-mail address, a piece of personal data.
- Two faulty policy extremes:
  - Preventing the extension from reading any personal data makes it useless.
  - Allowing the extension to read all personal data is unsafe.

A Need for Specification of Access Control Policies

Problem:

- How can we specify and enforce precise access control policies?

Example:

- A browser extension for automatically completing web registration forms needs to access the user's e-mail address, a piece of personal data.
- Two faulty policy extremes:
  - Preventing the extension from reading any personal data makes it useless.
  - Allowing the extension to read all personal data is unsafe.
- We need a *precise* policy in between extremes.

A Need for Specification of Access Control Policies

Problem:

- How can we specify and enforce precise access control policies?

Example:

- A browser extension for automatically completing web registration forms needs to access the user's e-mail address, a piece of personal data.
- Two faulty policy extremes:
  - Preventing the extension from reading any personal data makes it useless.
  - Allowing the extension to read all personal data is unsafe.
- We need a *precise* policy in between extremes.

Approach:

- Take advantage of the mathematical precision of logic.

An Example: Office Access

Setting:

- Three principals:

An Example: Office Access

Setting:

- Three principals:
  - Alice, a professor.

An Example: Office Access

Setting:

- Three principals:
  - Alice, a professor.
  - Bob, one of Alice's graduate students.

An Example: Office Access

Setting:

- Three principals:
  - Alice, a professor.
  - Bob, one of Alice's graduate students.
  - admin, the administrator that controls office access.

An Example: Office Access

Setting:

- Three principals:
  - Alice, a professor.
  - Bob, one of Alice's graduate students.
  - admin, the administrator that controls office access.
- Alice is out of town on 10/30/07.
- Bob needs a document from Alice's office.

An Example: Office Access

Setting:

- Three principals:
  - Alice, a professor.
  - Bob, one of Alice's graduate students.
  - admin, the administrator that controls office access.
- Alice is out of town on 10/30/07.
- Bob needs a document from Alice's office.

Important Policy:

- admin trusts Alice's access control decisions about her own office.

A Simple Access Request

- Alice signs a certificate allowing Bob to enter her office.

A Simple Access Request

- Alice signs a certificate allowing Bob to enter her office.
- At 1:00pm, Bob attempts to enter Alice's office.

A Simple Access Request

- Alice signs a certificate allowing Bob to enter her office.
- At 1:00pm, Bob attempts to enter Alice's office.
- Before the door will unlock, Bob must logically *prove* that admin allows him to enter Alice's office.

A Simple Access Request

- Alice signs a certificate allowing Bob to enter her office.
- At 1:00pm, Bob attempts to enter Alice's office.
- Before the door will unlock, Bob must logically *prove* that admin allows him to enter Alice's office.

Bob's Proof:

A Simple Access Request

- Alice signs a certificate allowing Bob to enter her office.
- At 1:00pm, Bob attempts to enter Alice's office.
- Before the door will unlock, Bob must logically *prove* that admin allows him to enter Alice's office.

Bob's Proof:

1. "Alice allows me to enter her office."

A Simple Access Request

- Alice signs a certificate allowing Bob to enter her office.
- At 1:00pm, Bob attempts to enter Alice's office.
- Before the door will unlock, Bob must logically *prove* that admin allows him to enter Alice's office.

Bob's Proof:

1. "Alice allows me to enter her office."
2. "admin trusts Alice's decisions about entry to her office."

A Simple Access Request

- Alice signs a certificate allowing Bob to enter her office.
- At 1:00pm, Bob attempts to enter Alice's office.
- Before the door will unlock, Bob must logically *prove* that admin allows him to enter Alice's office.

Bob's Proof:

1. "Alice allows me to enter her office."
2. "admin trusts Alice's decisions about entry to her office."
3. "Therefore, admin allows me to enter Alice's office."

A Simple Access Request

- Alice signs a certificate allowing Bob to enter her office.
- At 1:00pm, Bob attempts to enter Alice's office.
- Before the door will unlock, Bob must logically *prove* that admin allows him to enter Alice's office.

Bob's Proof:

1. "Alice allows me to enter her office."
2. "admin trusts Alice's decisions about entry to her office."
3. "Therefore, admin allows me to enter Alice's office."

- Bob's proof is correct; the door unlocks.

One-time Access: Linearity

Problem:

- The previous system allows Bob to enter Alice's office an *unlimited* number of times—probably not what Alice wants.

One-time Access: Linearity

Problem:

- The previous system allows Bob to enter Alice's office an *unlimited* number of times—probably not what Alice wants.

Solution:

- The logic must therefore express single-use resources, called *linear* assumptions.

One-time Access: Linearity

Problem:

- The previous system allows Bob to enter Alice's office an *unlimited* number of times—probably not what Alice wants.

Solution:

- The logic must therefore express single-use resources, called *linear* assumptions.
  - If Alice's certificate is linear, Bob's first correct proof will consume it.

One-time Access: Linearity

Problem:

- The previous system allows Bob to enter Alice's office an *unlimited* number of times—probably not what Alice wants.

Solution:

- The logic must therefore express single-use resources, called *linear* assumptions.
  - If Alice's certificate is linear, Bob's first correct proof will consume it.
  - Bob cannot enter Alice's office a second time—he cannot construct a correct proof because the certificate is now gone.

One-time Access: Linearity

Problem:

- The previous system allows Bob to enter Alice's office an *unlimited* number of times—probably not what Alice wants.

Solution:

- The logic must therefore express single-use resources, called *linear* assumptions.
  - If Alice's certificate is linear, Bob's first correct proof will consume it.
  - Bob cannot enter Alice's office a second time—he cannot construct a correct proof because the certificate is now gone.
- Previous work [2] has addressed linearity in an authorization logic.

Time-limited Access

Problem:

- With linearity, Bob can enter Alice's office only once, but on any day.

Time-limited Access

Problem:

- With linearity, Bob can enter Alice's office only once, but on any day.
- However, Alice wants to allow Bob to enter only once and *only* on 10/30/07.

Time-limited Access

Problem:

- With linearity, Bob can enter Alice's office only once, but on any day.
- However, Alice wants to allow Bob to enter only once and *only* on 10/30/07.
- Therefore, both linearity and an explicit notion of time are needed.

Time-limited Access

Problem:

- With linearity, Bob can enter Alice's office only once, but on any day.
- However, Alice wants to allow Bob to enter only once and *only* on 10/30/07.
- Therefore, both linearity and an explicit notion of time are needed.

Many other security policies are time-dependent.

Time-limited Access

Problem:

- With linearity, Bob can enter Alice's office only once, but on any day.
- However, Alice wants to allow Bob to enter only once and *only* on 10/30/07.
- Therefore, both linearity and an explicit notion of time are needed.

Many other security policies are time-dependent.

Thesis Question:

- How can we add an explicit notion of time to an authorization logic, while preserving linearity?

Thesis Project Goals

Goals:

1. Design an authorization logic for time-dependent policies.
2. Prove various metatheorems, including admissibility of cut.
3. Submit a paper for publication in a conference proceedings.
4. Implement a syntactic proof-checker for the logic.
5. Mechanically verify the logic's metatheory in Twelf.
6. Possibly extend the logic to express information flow policies.

Thesis Project Goals

Goals:

1. Design an authorization logic for time-dependent policies. ✓
2. Prove various metatheorems, including admissibility of cut.
3. Submit a paper for publication in a conference proceedings.
4. Implement a syntactic proof-checker for the logic.
5. Mechanically verify the logic's metatheory in Twelf.
6. Possibly extend the logic to express information flow policies.

Thesis Project Goals

Goals:

1. Design an authorization logic for time-dependent policies. ✓
2. Prove various metatheorems, including admissibility of cut. ✓
3. Submit a paper for publication in a conference proceedings.
4. Implement a syntactic proof-checker for the logic.
5. Mechanically verify the logic's metatheory in Twelf.
6. Possibly extend the logic to express information flow policies.

Thesis Project Goals

Goals:

1. Design an authorization logic for time-dependent policies. ✓
2. Prove various metatheorems, including admissibility of cut. ✓
3. Submit a paper for publication in a conference proceedings. ✓
4. Implement a syntactic proof-checker for the logic.
5. Mechanically verify the logic's metatheory in Twelf.
6. Possibly extend the logic to express information flow policies.

Thesis Project Goals

Goals:

1. Design an authorization logic for time-dependent policies. ✓
2. Prove various metatheorems, including admissibility of cut. ✓
3. Submit a paper for publication in a conference proceedings. ✓
4. Implement a syntactic proof-checker for the logic. ?
5. Mechanically verify the logic's metatheory in Twelf.
6. Possibly extend the logic to express information flow policies.

Thesis Project Goals

Goals:

1. Design an authorization logic for time-dependent policies. ✓
2. Prove various metatheorems, including admissibility of cut. ✓
3. Submit a paper for publication in a conference proceedings. ✓
4. Implement a syntactic proof-checker for the logic. ?
5. Mechanically verify the logic's metatheory in Twelf. ?
6. Possibly extend the logic to express information flow policies.

Thesis Project Goals

Goals:

1. Design an authorization logic for time-dependent policies. ✓
2. Prove various metatheorems, including admissibility of cut. ✓
3. Submit a paper for publication in a conference proceedings. ✓
4. Implement a syntactic proof-checker for the logic. ?
5. Mechanically verify the logic's metatheory in Twelf. ?
6. Possibly extend the logic to express information flow policies. ?

Evaluation of Progress

1. Correctness of desirable metatheorems.
   - Admissibility of cut
   - Interval subsumption

Evaluation of Progress

1. Correctness of desirable metatheorems.
   - Admissibility of cut
   - Interval subsumption
2. Ability to model example time-dependent policies.
   - Office access
   - Journal review and publication
   - Course filesystem (e.g., Blackboard)
   - Kerberos-like protocol
   - Prescription filling

Summary

Summary:

Summary

Summary:

- A system for precisely specifying access control policies is needed.

Summary

Summary:

- A system for precisely specifying access control policies is needed.
- Many policies are time-dependent, so an authorization logic must express time. This is my thesis project.

References

K. Bowers, L. Bauer, D. Garg, F. Pfenning, and M. Reiter
Consumable Credentials in Logic-Based Access-Control Systems
In *Proceedings of the 14th Annual Network and Distributed System Security Symposium*, February 2007.

D. Garg, L. Bauer, K. Bowers, F. Pfenning, and M. Reiter.
A Linear Logic of Authorization and Knowledge.
In *Proceedings of the 11th European Symposium on Research in Computer Security*, pages 297–312, September 2006.