Logics for Authorizations and Security

Ahmad Sepahi
Outline

• Introduction
• Preliminaries on security formalization
• Logic-based policy specification languages
• Policy evaluation and verification
• Other applications of logic to security
• Conclusion
Introduction

• The expressive power & the formal foundation of logical formalism

• Complicated and intimidating

• Formal guarantees, flexibility, expressiveness, declarativeness (no programming ability)
Introduction

• Precise specification of security needs:
  • What to be protected and from whom/what?
    • Solution: sophisticated logics
  • How to specify what is *authorized* or *forbidden*?
    • Solution: horn clause
  • How to prove that systems/applications are secure?
    • Solution: Automated reasoning technique
Outline

• Introduction
• Preliminaries on security formalization
• Logic-based policy specification languages
• Policy evaluation and verification
• Other applications of logic to security
• Conclusion
Preliminaries on security formalization

• Components of logical formalization of security
  • An abstract computational model of the system
    • Turing model (abstract state machine)
    • Can not capture OS bugs
  • A security policy
    • Sets of constraints
    • Syntactic representation
  • A formal definition of security properties
    • Specify the security goals
Outline

- Introduction
- Preliminaries on security formalization
- Logic-based policy specification languages
- Policy evaluation and verification
- Other applications of logic to security
- Conclusion
Logic-based policy specification languages

- Clean and unambiguous semantic, suitable for implementation validation
- Expressive enough to formulate all the policies in the literature
- High level of abstraction, close to natural language
Logic-based policy specification languages

• Dynamic policies

• Hierarchies, inheritance and exceptions

• Message control

• Policy composition framework

• Credential based access control
Dynamic policies

• Time dependent policies
• Creation and removal of users, objects
  • HRU
    • No fixed finite bound
    • Undecidable in general
• Two approaches
  • Temporal authorization bases
  • Active rules
Dynamic policies

Temporal authorization bases

- Expressive enough for typical real world
  - ([begin, end], p, A <op> α)

- Includes
  - Periodic authorization
    - (subject, object, action, sign, grantor)
    - Temporal expression: <[begin, end],p>
    - <[1/1/2000, 1/1/2001], 9 a.m. to 12 a.m.>
  - Temporal operators
    - Whenever
    - As long as
    - Upon
Dynamic policies
Active Rules

- Called role triggers
- Dynamic changes concern roles
  - Static
  - Enable/disable
  - No creation/deletion
  - Decidable
Dynamic policies
Active Rules

• Syntax
  • Event expression
    • $p_1: \text{enable } R$, $p_2: \text{disable } R$
    • VH: enable doctor on day duty
  • Status expression
    • $\text{Enabled } R$, $\neg \text{enabled } R$
  • Role triggers
    • $S_1, ..., S_n, E_1, ..., E_m \rightarrow p: E_0 \text{ after } \Delta t$
    • $\text{Enable doctor on day duty } \rightarrow H: \text{enable nurse on day duty after 1}$
Logic-based policy specification languages

- Dynamic policies
- Hierarchies, inheritance and exceptions
- Message control
- Policy composition framework
- Credential based access control
Hierarchies, inheritance and exceptions

• Users(objects) can be collected in groups (classes)

• The authorizations granted to a group(class) apply to all its members

• Role hierarchies are triples \((X,Y,\leq)\)
Hierarchies, inheritance and exceptions

- Hierarchy of authorization

  - E.g., \((s, o, a) \leq (s', o', a')\) iff \(s \leq s'\), \(o \leq o'\), \(a \leq a'\)

- \((s, o, a)\) is more specific than \((s', o', a')\)

- \((s, o, a)\) is inherited from \((s', o', a')\)
Hierarchies, inheritance and exceptions

• Exception deals with conflicts
  • \((s, o, a, +)\) is inherited from \((s, o', a, +)\)
  • \((s, o, a, -)\) is inherited from \((s', o, a, -)\)

• Simplest conflict resolution methods
  • Denial takes precedence
  • Permission takes precedence
Hierarchies, inheritance and exceptions

• Different approaches:
  • Flexible Authorization Framework (FAF)
  • Hierarchical temporal authorization model
  • Ordered logic programs
Logic-based policy specification languages

- Dynamic policies
- Hierarchies, inheritance and exceptions
- Message control
- Policy composition framework
- Credential based access control
Message control

- Send(s, m, t)

- \( m := (\text{subject, object, action}) \)

- Policies consists sets of laws

- Laws map send(s, m, t) to \((m’, t’)\) or fail
Message control

• Laws are composed of several rules

• $R_1 : \text{send}(S, ^M, T) \rightarrow \text{isa}(T, \text{module}) \& T.\text{owner}=S \& \text{deliver}(^M, T)$

• $R_2 : \text{send}(S, @M, T) \rightarrow \text{isa}(S, \text{module}) \& \text{isa}(T, \text{module}) \& \text{deliver}(@M, T)$
Logic-based policy specification languages

- Dynamic policies

- Hierarchies, inheritance and exceptions

- Message control

- Policy composition framework

- Credential based access control
Policy composition Framework

- Multiple requirement $\rightarrow$ combining different security policy

- Policy composition approaches
  - Constraint based approaches
  - Deontic approach
  - Algebraic approach
Policy composition Framework
Constraint based approaches

• Labeling users and objects

• Security levels belongs to a finite partially ordered set by a relation $\leq$

• No read up $\Rightarrow l' \leq l$

• No write down $\Rightarrow l \leq l''$
Policy composition Framework
Deontic approach

• Modal logic

• Modal operators
  • O : Obligatory
  • P : permitted
  • F : forbidden
Policy composition Framework

Algebraic approach

- Expressiveness
- Formal semantics
- Controlled interference
- Support of unknown policy
- Heterogeneous policy support
- Support of different abstraction levels
Logic-based policy specification languages

- Dynamic policies
- Hierarchies, inheritance and exceptions
- Message control
- Policy composition framework
- Credential based access control
Credential based access control

- Use of remote resources and services
- Digital certificates or credential
  - What should be communicated?
  - How to communicate?
- Using logic based policy language for
  - Establishing protection requirements on the server
  - Restricting credentials and declarations release
Outline

- Introduction
- Preliminaries on security formalization
- Logic-based policy specification languages
- Policy evaluation and verification
- Other applications of logic to security
- Conclusion
Policy evaluation and verification

• Automated or semi automated policy verification tools
  • System’s availability
  • No access is ever denied
  • Can not access data without owner’s explicit authorization
Outline

• Introduction
• Preliminaries on security formalization
• Logic-based policy specification languages
• Policy evaluation and verification
• Other applications of logic to security
• Conclusion
Other applications of logic to security

- Trust management
- Model checking for protocol verification
- Intrusion detection
Outline

• Introduction
• Preliminaries on security formalization
• Logic-based policy specification languages
• Policy evaluation and verification
• Other applications of logic to security
• Conclusion
Conclusion

• Application of different logics to the specification and enforcement of security policies

• Logic based formulations helped making progress in the formalization of security problems and their solutions

• Logic based policy languages provide the flexibility and expressiveness
References
