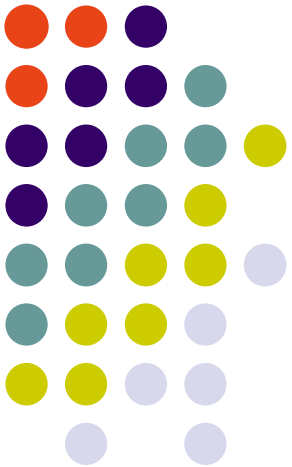
Security policy systems and their consistency problems

Luigi Logrippo, Kamel Adi

Université du Québec en Outaouais

luigi@uqo.ca



Research at the UQO
Computer Science and Engineering

- Canada Research Chair in Photonics
- University Research Chair in Distributed Computing
- Security Laboratory
  - My colleague Kamel Adi
  - We organized 3 international workshops on Access control and security in the past couple of years
- We have also research labs in multimedia, robotics, hardware design, wireless communications, etc.
- Master's and PhD program

Our Claim

- Formal auditing and formal verification of security policies is becoming possible by the use of logic tools

Formal auditing of accounts

Has a company made 17,000 if it has had incomes of 15,000 and 5,000, and loss of 3,000?

$15,000.00+
$ 5,000.00+
$ 3,000.00-

$17,000.00

Formal auditing of security policies

Can nurses gain access to accounting information in a hospital?

Only employees in the accounting department can access acct info

Nurses are in the patient care department

No employees can be in both patient care and accounting

No nurse can gain accept to accounting information

This is formally verifiable if enterprise policies are expressed in a suitable formal language

Policy languages for customizing enterprise systems

- Policy languages are already being used in order to express enterprise policies in many applications. Examples:
  - Firewalls
  - Access control (e.g. language XACML, SiteMinder)
  - Security models (Bell-LaPadula, Chinese Wall, RBAC…)
  - Identity identification
  - Telecommunications features, call control
  - Telecommunications routers
  - Web services orchestration and choreography (e.g. language BPEL)
  - E-commerce

Policy sources

- Enterprise policies can be
  - formalized in computer language, or
  - be in printed regulations, or
  - be in the ‘memory’ of the enterprise
- Policies can originate in several different places in an enterprise
  - Can be attached to administrative units or roles
- Can be created by agreements, internal or external
- Policies come and go as the enterprise evolves

- We'll start with a simpler problem
  - Policy consistency

Policy consistency

- Policies must be *mutually consistent* in order to work together
  - Inconsistency of policies can lead to inability to take a decision in a particular situation
  - Or to the *wrong* decision being taken
- Research issues:
  - How to *check* policy consistency
  - How to *maintain* policy consistency

Approach

- Consistency is a logical problem
  - Tools to perform ‘efficient’ consistency checking are becoming available
- Translate policies in logic formalism
- Use logic tools to check consistency:
  - Model checkers
  - Theorem provers
- Once an inconsistency has been identified, it must be reported to user to check if it is *intended*

Required: a formal model of the enterprise

- Of the organization structure
  - Of all policies in an organization
  - Not too much to ask?
    - partial models can be sufficient in many cases

Results:

1. XACML case study

- XACML: A language to specify access control constraints to computing resources
  - To files, equipment, web services …
- OASIS standard
- Some functionalities of XACML are present in SiteMinder, to which our approach may apply

Access Control Applications

- Access control is becoming very important in the presence of strict accessibility and privacy laws
  - And the possibility of remote access via web services
- Example: in a hospital
  - *Statistician* can access files for reading patient data, but not names
  - *Physician* can access for read/write most info for her patients
    - But not billing info
  - *Patient* can read selected information on her record
  - *Accountant* can only access accounting info
  - Etc. etc.

Example of XACML Rule

```
<Rule RuleId="Rule1" Effect="Permit">
    <Description> A professor can perform any action on the file of the course she teaches </Description>
        <Target>
            <Subjects>
                <Subject>
                    <SubjectMatch MatchId="string-equal">
                        <AttributeValue DataType="string"> Professor </AttributeValue>
                        <SubjectAttributeDesignator AttributeId="Role" DataType="string"/>
                    </SubjectMatch>
                </Subject>
            </Subjects>
            <Resources><AnyResource/></Resources>
            <Actions> <AnyAction/> </Actions>
        </Target>
        <condition>
            <Apply FunctionId="function:string-is-in">
                    <Apply FunctionId="function:string-one-and-only">
                        <ResourceAttributeDesignator AttributeId="courses" Datatype="string"/>
                    </Apply>
                    <SubjectAttributeDesignator AttributeId="taught_courses" Datatype="string"/>
            </Apply>
        </condition>
    </Rule>
```

Checking Access Control Policies

- Policies can be translated in logic notation, and consistency checks can be run

Possible relationships between targets

- Disjunction
- Non-empty intersection
- Inclusion
- Equality

- Of:
  - Subjects
  - Resources
  - Actions

Disjunction

- There is disjunction between targets if subjects, resources, and actions are all distinct

- Disjunction between the targets of two policy sets, policies, or rules means that
  - there is no case in which both are applicable
    - OK, then

Non-empty Intersection

- There is ***nonempty intersection*** between two targets if:
  - The set of subjects intersect **and**
  - The set of resources intersect **and**
  - The set of actions intersect
- Example:
  - (deny) (Executives) (databases A, B, C) (action read, write)
  - (accept) (Employees) (database B, D) (action write)
    - Executives, being employees, have conflicting write rights on B

How to fix?

- Non-empty intersections must be reported to user
- Indicate that some relationships may not have been thought out
- But the following are interesting special cases

Inclusion of target T1 in target T2

(special case of nonempty intersection)

- There is ***inclusion*** between two targets if:
  - The set of subjects of T1 **is included** in the set of subjects of T2 **and**
  - The set of resources of T1 **is included** in the set of resources of T2 **and**
  - The set of actions of T1 **is included** in the set of actions of T2
- Example:
  - (accept) (executives) (database B) (action write)
  - (deny) (employees) (database A,B,C) (any action)

Is there something to fix?

- The included case can be wanted as an exception but:
  - Check that the exception is really wanted
    - (accept) (executives) (database B) (action write)
    - (deny) (employees) (database A,B,C) (any action)

Equality
(special case of nonempty intersection)

- Subjects, target, action are all the same
  - Almost certainly, an error

Proof of concept:
2. Firewall case study

- Contrary to XACML, where all rules must be considered, in firewalls rules are considered top-down,
  - The first applicable rule is used, all the following ones are skipped
  - So in principle inconsistencies are impossible, however if there are two rules that lead to different decisions, one of them may have been included by error
  - Hence it is a safe strategy to look for inconsistencies among rules

Domain intersections, again

- As in the previous case, domain intersections are symptoms of possible errors, but the order is important

Shadowing

- A rule is preceded by a more general one with different outcome
  - Refuse all packets from port 25
  - Accept all packets from port 25 but from domain CA
    - (the second rule is ineffective)

How to fix

- Either take out the second rule,
- Or if the second rule is a desired exception, move it ahead

Generalization

- A rule is followed by a more general one but with a different outcome
  - Accept all packets from port 25 and from domain CA
  - Refuse all packets from port 25
    - (the first rule is an exception wrt to the first one)
    - (it must be checked whether the exception is desired)

Is there something to fix?

- Check that the exception is really wanted

Other cases…

- There are other cases of nonempty intersection between the domains of applicability of two rules
- They should be detected and pointed out to user because they could be the result of user errors

User Interfaces for policy creation

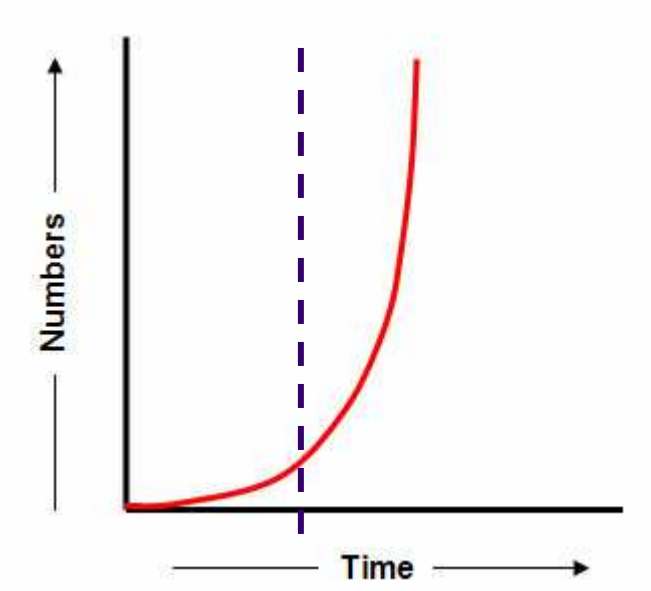
- Ideally, policies should be created and modified directly by privacy specialists
  - Often people with limited computing expertise
- These people should be able to
  - Enter policies by clicking on boxes
    - Or in English-like language
  - Obtain diagnostics in clear
  - Exercise different cases
    - What will happen if Joe wants to write on file X?
- Need of appropriate GUIs

Decisions and user intentions

- Since systems can have many rules, the existence of different rules for the same cases indicates the possibility of user error
  - Some user *intention* which is not properly represented

Difficulty of the detection problem

- In principle, the detection of inconsistencies is a *constraint satisfaction problem*
  - Exponential complexity
- However the examples we have seen have few variables, thus problem is still tractable



Research tools to be considered

- Coq, a theorem-proving tool developed at INRIA
- Alloy, a model-checker developed at MIT
- Z-Eves, Canadian product, much used in research
- Constraint programming languages, e.g. Constraint Prolog
- Implementation in standard programming languages such as C is possible, but requires more work

Security agreements

- WS-Policy Framework and other related standards establish rules by which parties can agree on security rules for a transaction
- Each party will advertise own security policy wrt signatures, encryption, etc.
- Are the requirements of the various parties mutually consistent?
- Are the requirements of one a special case of the requirements of the other
- Are there ways that the requirements can be reconciled?

Privacy and secrecy models

- Main privacy/secrecy models known:
  - Bell-LaPadula
  - Chinese Wall
  - RBAC
  - Delegation models
- Can they be reconciled?
  - Our work is showing that they can be combined together
  - However conditions should be added in each case

E.g. effects of models on *delegation*

- Chinese Wall is being used
  - A and B are in different domains
  - So they cannot *delegate* to each other
- Bell-LaPadula is being used
  - A can delegate to B if and only if
    - A and B have equal clearance level
    - B has higher clearance level than A
- These facts we can prove automatically by using our automated system
- They are for themselves fairly simple, however much more complex properties can also be proven.

Many applications

- These ideas are of very general application, not at all limited to the case studies we have mentioned

- E.g. in the case of identity management, we can have sets of policies that can be partially inconsistent

Conclusions

- Strict accessibility and privacy laws will require increasing attention to this area
- Security can be compromised by inappropriately formulated access control rules
- Formal methods will allow **formal auditing and verification** of accessibility and privacy policies, as precise as formal accounting of financial records!
- Tools to do this are available, but must be adapted