An Approach to Correctness of Security and Operational Business Policies

V.R. Karimi, D.D. Cowan, P.S.C. Alencar

Cheriton School of Computer Science
University of Waterloo
Several incidents of lack of compliance in real systems have been reported in the literature.

These incidents indicate the need for an assurance that these errors do not happen.

Testing reveals many existing errors in software systems, but errors still remain undetected even in safety-critical and vital economic systems.

Testing is not enough.

Therefore, this talk is about one critical aspect of information systems assurance namely a thorough analysis of access control policies in different business domains.
Access Control Policies and Rules

Access control policies determine which users have access to what objects and operations and under what constraints.

The following policy in a banking systems consists of a single rule.

**Example:** A teller is permitted to modify deposit accounts.

Another example of these policies can be the following two sentences that are normally joined with a combining scheme. A combining scheme is described shortly.

The following policy consists of multiple rules.

**Example:** A teller is permitted to modify deposit accounts; a customer service rep is permitted to create, delete, or modify deposit accounts.
The Approach

The approach uses the Resource-Event-Agent (REA) model that is developed in the business and accounting community.

Rules are expressed in REA and are combined by a combining scheme using a state machine conforming a model.

\[
\text{Model} = \text{Rules and a Combining Scheme}
\]

This model can be encoded using a software tool such as the SPIN model checker.
Then, different questions about this model can be checked. These questions are referred as properties of the model.

The following figure shows the approach:

For instance, a property to be checked in a model can be as follows:

A teller can always deposit into a customer account.
Rule Representation

Example: A teller is permitted to modify deposit accounts.

This example can also be represented as follows:

modify(teller,depositAccount)  →  permit

A rule can be expressed as an implication, which consists of a premise and a conclusion, as

premise-rule  →  conclusion-rule  or
p-rule  →  q-rule

p-rule and q-rule are used as a short form of premise-rule and conclusion-rule when diagrams are presented to avoid clutter.

Example: A teller is permitted to modify deposit accounts.

In this example, teller, deposit accounts, and modify are included in premise-rule, and is permitted is included in the conclusion-rule.
Combining Rules

Some policy standards use different schemes or algorithms; e.g., the first-applicable scheme or algorithm, that describe the combination and evaluation of rules.

As the name of this scheme indicates, the first applicable rule in a collection of rules takes precedence, and the result of this first applicable rule holds.

A procedural description of this scheme follows:

- Rules are evaluated in the same order listed in a policy.
- If a rule evaluates to “permit” or “deny”, then the result is permit or deny; no more rules are examined.
- If none of the rules applies, then the evaluation returns “not applicable (NA).”
Combining Rules using Diagrams

To describe the combination of rules, we use a diagrammatic representation called a state diagram or state machine.

A state machine is a diagram consisting of nodes (or states) represented by square rectangles with names inside.

For instance, $s_{10}$, and $s_{11}$ are the names of two nodes (or states) as shown below:

A transition from one node to the other is shown by a line. For instance, a transition from node $s_{10}$ to $s_{11}$ when a timeout occurs is shown by a line from the first state to the latter as follows:
Combination of Policies using State Machines

An overview of the combination of policies using state machines and the previously described first-applicable scheme is shown next.

\[ s_0 \]

\[ [p\text{-rule}_1 = \text{true}] \quad [p\text{-rule}_1 = \text{false}] \]

\[ s_{11} \]

\[ s_{10} \]
Combination of Policies using State Machines

An overview of the combination of policies using state machines and the previously described first-applicable scheme is shown next.
Combination of Policies using State Machines

An overview of the combination of policies using state machines and the previously described first-applicable scheme is shown next.
Combination of Policies using State Machines

An overview of the combination of policies using state machines and the previously described first-applicable scheme is shown next.

![State Machine Diagram]

- Transition from $S_{00}$ to $S_{11}$ with $p\text{-rule}_1 = \text{true}$.
- Transition from $S_{00}$ to $S_{10}$ with $p\text{-rule}_1 = \text{false}$.
- Transition from $S_{11}$ with $q\text{-rule}_1 = \text{permit}$.
- Transition from $S_{10}$ with $p\text{-rule}_2 = \text{true}$.
- Transition from $S_{10}$ with $p\text{-rule}_2 = \text{false}$.
- Transition from $S_{21}$ with $q\text{-rule}_2 = \text{deny}$.
- Transition from $S_{20}$ with $q\text{-rule}_2 = \text{permit}$.
A Program for Combining Rules

It is possible to use a program description to present the information provided in the previous diagram as follow:

\[
\text{for } i = \text{rule}_1 \text{ to rule}_n \text{ do} \\
\quad \text{if premise-rule}_i \text{ does not hold} \\
\quad \quad \text{continue evaluation} \\
\quad \text{else} \\
\quad \quad \text{if premise-rule}_i \text{ holds} \\
\quad \quad \quad \text{then the consequence rule}_i \text{ applies (either permit or deny)} \\
\quad \quad \quad \text{exit evaluation} \\
\quad \text{end-if} \\
\text{end-if}
\]

Program is needed to automate the checking of large number of rules.

Then, different questions can be formed to check whether the desired properties are hold.
Conclusion

We have focused on a critical aspect of information systems assurance that can be applied to different business domains (e.g., ERP).

We have proposed an approach to describing security and operational business policies and verifying their correctness with respect to a set of properties.

The method is based on the REA business modeling language to construct definitions of security and operational business rules.