Extended Role Based Access Control for Trusted Operating Systems and its Coloured Petri Net Model

Gwangju Institute of Science and Technology

SHIN, Wook
Trusted Operating System (TOS)

- App. level security solutions can be bypassed [1]
  - Intrusion Detection System (IDS) and Firewall are executed in application level

- TOS is an even more fundamental security solution

"Without TOS, all security efforts result in Fortress built upon sand" [2]
Insufficiency of Current Access Controls

- Current Access Controls
  - Control Accesses Based on Instant Access Information
  - They cannot block some kinds of attacks
    Ex) race condition attacks

```
program P (arga, argb) {
  ...
  load_data(arga);
  load_data(argb);
  calculate;
  save_data(argc);
  load_data(argc);
  print;
  print;
  ...
}
```
Additional Constraints of E-RBAC

• We propose an extended access control
  – Extend the vision and the functionality of the concept of access control based on the sequence of operations

• Subject Abstraction and Object Abstraction
  – Roles: a set of users (subject-abstraction)
    Ex) Secretaries := {John, Michael, Tom}
  – Behaviors: a set of permissions (object-abstraction)
    Ex) FileOpSet := { f_open, f_close, f_read, f_write}

• Operations in E-RBAC
  – expressed in the Behavior layer
    • Permitted operations without procedural restrictions
    • Prohibited operations without procedural restrictions
    • Permitted execution sequences of operations (Positive procedural constraints, Positive PC)
    • Prohibited execution sequences of operations (Negative PC)
Extended-Role Based Access Control

- Extended RBAC (E-RBAC)
  - Core E-RBAC
  - Constrained E-RBAC

- The Conceptual Diagram
Modeling Behaviors

- Normal and Dangerous behaviors can be described

<table>
<thead>
<tr>
<th>Positive</th>
<th>The permitted behavioral patterns of log file management</th>
</tr>
</thead>
<tbody>
<tr>
<td>(lf, open)</td>
<td>(lf, read)</td>
</tr>
<tr>
<td>(lf, open)</td>
<td>(lf, write)</td>
</tr>
<tr>
<td>(lf, open)</td>
<td>(lf, close)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Negative</th>
<th>The prohibited behavioral pattern of the race condition attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s1, execute)</td>
<td>(f1, unlink)</td>
</tr>
</tbody>
</table>
A Formal Model for E-RBAC

• We need a formal method to specify and verify security configuration of an E-RBAC system

• We define
  – a new formal model Constrained Coloured Petri Nets (CCPN)
    • based on Coloured Petri Nets (CPN) formalism
    • CCPN describes access matrix information and procedural information at the same time
Constrained CPN (CCPN)

- Main Components of Coloured Petri Net (CCPN)
  - Additional Component: Access Matrix
    - row: subjects
    - column: objects
    - entry: permissions
  - Interpretation: CPN Components are interpreted as AC entities
    - Tokens: Access Subjects
    - Places: Access Objects
    - Transitions: AEFs (Access Enforcement Function)
  - Modified Enable Condition
Testing a configuration with CPN

- We can test security related properties by
  - Simulation
  - Formal Analysis

- Example System Configuration
  - USERS = \{u_1, ..., u_i\}
  - ROLES = \{SysAdmin, User, r_1, ..., r_j\}
  - Objects = \{logfile, mail_prg, file_1, ..., file_k\}
  - Modes = \{read, write, open, close, execute, link, unlink\}
  - Behaviors = \{ExecuteMailProgram, AccessLogFiles, b_1, ..., b_n\}

- Using the formal method, we can correct security configuration errors
Simulation Example

- Analysis by Simulation: A Positive PC Example
  - The sets of execution sequences are performed well
  - \{open-read*-close\} or \{open-write*-close\}
Formal Analysis

- Analysis by Formalism
  - Liveness
    - Liveness check for the transition of attack detection
    - We can find and remove the possibility of being attacked

Formal analysis results

Liveness check of this transition

Remove a dangerous operation

Modified configuration (Prohibit the unlink operation)

Original configuration

Security Research Group
An Implementation

- The Implementation Environment
  - IFC-ETK100: An Embedded Board
  - CPU: SE3208 (32 bit EISC Processor)
  - Memory:
    - 4M ROM, 4M Flash, 16M SDRAM
  - OS: uClinux-2.4.19

- The Implementation Result
  - Successfully detects race condition attacks
Performance Test

• Performance Measurement
  – Time costs of the execution of a simple program
  – Time costs of the execution of a file copy (512bytes)
  – Time costs of the execution of a simple program that have procedural constraints

• Results

Our system: 10 % overhead

Overhead of other systems
- A current TOS implementation (SELinux): 5%
- A current application level IDS solution (Snort): 10%
Conclusion

• The achievements
  – Extended RBAC Model
    • The vision and function of access control are extended
    • The attacks which consist of ordinary operations are denied
  – CPN Model for E-RBAC
    • Hybrid model for access control
    • Helpful for security administration
  – Trusted Embedded OS
    • E-RBAC can be implemented with reasonable overheads
Bibliography


