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

Gwangju Institute of Science and Technology

SHIN, Wook
# Table of Contents

1. Introduction and Motivation

3. **Extended Role Based Access Control (E-RBAC)**
   - Concept and Model

5. **Coloured Petri Net Formal Model for E-RBAC**
   - Formal specification and verification

7. **Implementation & Performance Evaluation**
   - The implementation example on an Embedded System
   - Performance Evaluation

9. Conclusion
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]
TOS and Access Control

- What is the Trusted Operating System (TOS)?

TOS := traditional OS services + security services
Access Control

- Access Control: the core function of the TOS
Current Access Controls

- The process of access control
  - Gather Access Control Information (ACI) at the moment of each access
  - Make a decision based on the ACI
  - Discard the ACI

```cpp
program P (arga, argb) {
    ...
    load_data(arga);
    load_data(argb);
    calculate;
    save_data(argc);
    load_data(argc);
    print;
    print;
    ...
}
```
Insufficiency of Current Access Controls

- Current access controls cannot block some kinds of attacks
Summary of the Motivation

- Current access control process is insufficient
  - We need a stronger method

- We propose an extended access control
  - Extend the vision and the functionality of the concept of access control based on the sequence of operations
# Table of Contents

1. Introduction and Motivation

3. **Extended Role Based Access Control (E-RBAC)**  
   – Concept and Model of E-RBAC

5. Coloured Petri Net Formal Model for E-RBAC  
   – Formal specification and verification

7. Implementation & Performance Evaluation  
   – The implementation example on an Embedded System  
   – Performance Evaluation

9. Conclusion
Role Based Access Control

- The Core of RBAC
  - Abstractions
  - Interface for giving additional constraints

- Our Extended Features are base on the abstractions
RBAC v.s. E-RBAC

- **RBAC**, one of the traditional access controls
  - No component to express execution sequences

- Subject- and object-abstractions are mixed in one abstraction layer

- **E-RBAC**
  - Components to express execution sequences
    - Ordering information
    - Identification information

  Abstractions are distinguished
  - Subject abstraction does not need the properties
    - Overhead for implementations

* Introducing the concept of negative permission
Additional Constraints of E-RBAC

• 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

![Diagram showing Extended RBAC components: USERS, ROLES, BEHAVS, PERMS, SA, AR, OA, PC, PR, PU, PH, SESSIONS]

- ■ : Core E-RBAC
- ■+■ : Constrained E-RBAC
Core E-RBAC

- Core E-RBAC Model
  - USERS, ROLES, BEHAVS, and PERMS
  - SESSIONS
  - SA ⊆ USERS × ROLES
  - OA ⊆ BEHAVS × PERMS
  - AR ⊆ ROLES × BEHAVS

- assigned_users: (r: ROLES) → 2^{USERS}, the mapping from a role r onto a set of users
  - Formally: assigned_agents(r) = \{ a ∈ USERS | (a, r) ∈ SA\}

- assigned_permisions: (b: BEHAVS) → 2^{PERMS}, the mapping of behavior b onto a set of permissions
  - Formally: assigned_permissions(b) = \{ p ∈ PERMS | (p, b) ∈ OA\}
Core E-RBAC

- agent_session(a: AGENTS) → s (s: SESSIONS), the mapping of agent a onto a session
- session_role(s: SESSIONS) → r (r: ROLES), the mapping of session s onto a role

- assigned.behaviors: (r: ROLES) → 2^{BEHAVS}, the mapping of role r onto a set of behaviors
  - Formally: assigned.behaviors(r) = \{ b ∈ BEHAVS | (r, b) ∈ AR\}

- avail_session_permissions: (s: SESSIONS) → 2^{PERMS}, the mapping from a session s onto a set of permissions
  - Formally: avail_session_permissions(s) = \bigcup_{b ∈ assigned.behaviors(r)} assigned_permissions(b)
    (,where r ∈ session_roles(s))
Constrained E-RBAC

• PR Components
  – PR(Procedural Restrictions)
    • PU(Procedural Unit)
      – Behavior × Execution Order × Repetition
    • PC(Procedural Constraint)
      – PU × Identification Property
  • PH(Procedural History)
    – N × Session × Role × Behavior × Order in PC × PC
Modeling Behaviors

- Normal and Attack 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>
Modeling Behaviors with PR

- Normal and Attack behaviors can be expressed with PR elements

![Diagram of PR elements for Log Mgmt and Send Mail](image)
# Table of Contents

1. Introduction and Motivation

3. **Extended Role Based Access Control (E-RBAC)**
   - Concept and Model of E-RBAC

5. **Coloured Petri Net Formal Model for E-RBAC**
   - Formal specification and verification

7. **Implementation & Performance Evaluation**
   - The implementation example on an Embedded System
   - Performance Evaluation

9. Conclusion
Security Administration

- Protect a system
  Security policy → Implementation → Security administration

- Defense a fortress
  Elaborate plans → Build barriers and traps → Assign soldiers

- Security administration
  - Wrong configuration gives rise to security flaws
    - unauthorized access, denial of service
  
  - Finding faults in a configuration
    - Manual or trial-and-error is almost impossible
    - Mathematical proof will be helpful
Formal Methods

- Formal methods in TOS developments
  - for correct design and implementation
    - Specifying requirements and systems
    - Specifying security policies, models, and implemented systems
    - Determining how well a specification meets the requirements
  
  - for security administration
    - check the correctness of system configuration before it is applied to a real system
E-RBAC Models are insufficient

- The Core-/Constrained-E-RBAC models
  - follow the standard RBAC model[9]
  - describe the concept of the extended method without ambiguity

- but, it is hard to
  - specify the requirements which consist of partially ordered operations
  - test the system configuration automatically

Based on set notations
- Efficient to descript, and calculate authorities
  - \( R_{u1} = R_{sub1} \cup R_{sub2} \)
  - \( \text{avail_session_permissions}(s) = \bigcup_{b \in \text{assigned_behaviors}(r)} \text{assigned_permissions}(b) \)

(where \( r \in \text{session_roles}(s) \))
A new model for E-RBAC

- A new model for E-RBAC should express
  - the traditional access control information
    - Inductive verification (or proof-based) techniques
  - the execution sequence of operations
    - Model checking (or model-based) techniques

- We define
  - a new model based on Coloured Petri Nets (CPN) formalism
Coloured Petri Nets

- Coloured Petri Nets (CPN)
  - Modeling concurrent systems
  - The state machine based formalism, but also supports
    - Type definitions: Token, Place have data type (color)
    - Type Manipulations: Transition, Arc have expression
  - Other advantages [7]
    - CPN support hierarchical structures
    - CPN have computer tools supporting their drawing, simulation, and formal analysis
Constrained CPN

• Constrained Coloured Petri Net (CCPN)
  – The CPN formal model for E-RBAC

  – 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
Constrained CPN

- Constrained CPN (CCPN)

Enable condition: \( \text{sec}_{-}\text{rcv} \land \text{sec}_{-}\text{trans} \land \text{sec}_{-}\text{snd} \)

\( R(\text{tok}_{\text{in}}) \cap R(\text{i}_p) = \emptyset \)

\( R(\text{tok}_{\text{out}}) \cap R(\text{o}_p) = \emptyset \)
CCPN Example

- Overall Diagram

\[(\text{behav} \leftrightarrow \text{mailex}) \land (\text{behav} \leftrightarrow \text{pwaq})]\]

- if ADF(rtok,rp)
  - then 1\(^\ast\)(uid, sid, oid, mode, rtok, rp, behav)
  - else empty

- Token

- GenOPs

- Ready

- Operation under positive constraints

- Positive
  - PositiveRestriction1

- Operation under negative constraints

- Negative
  - NegativeRestriction1

- Detected

- attack trials

- Token
CCPN Example

- Positive PC
CCPN Example

- Negative PC
Testing a configuration with CPN

• Analysis
  – Simulation
  – Formal Analysis

• Example 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\}
Simulation Example

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

• Analysis by Simulation: A Negative PC Example
  – The attack sequence is detected correctly
  • \{mail execution\-unlink\-link\}
Formal Analysis

- Analysis by Formalism
  - Liveness
  - Liveness check for the transition of attack detection

Formal analysis results

Modified configuration (Prohibit the unlink operation)

Liveness check of this transition
# Table of Contents

1. **Introduction and Motivation**

3. **Extended Role Based Access Control (E-RBAC)**
   - Concept and Model of E-RBAC

5. **Coloured Petri Net Formal Model for E-RBAC**
   - Formal specification and verification

7. **Implementation & Performance Evaluation**
   - The implementation example on an Embedded System
   - Performance Evaluation

8. **Conclusion**
Implementation Environments

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

• For traditional Access Control (Core E-RBAC)
  – Process
    • Permission vector (Information of roles)
  – File
    • Permission vector (Information of roles)
  – ADF
    • Comparing the set of roles

• For behavior traces (Constrained E-RBAC)
  – Process
    • Information of current states
  – ADF
    • Calculate next states from current states and current action
Detection Example

- Attack Program

```
# Sunihill@themis.gist.ac.kr: /home/sunihill

drwxr-xr-x 2 0 0 12288 Sep 2 2004 lost+found
drwxr-xr-x 2 0 0 1024 Jul 8 2003 mnt
dr-xr-xr-x 2 0 0 0 Apr 17 16:55 proc
drwxr-xr-x 2 0 0 1024 Jul 8 2003 root
drwxr-xr-x 2 0 0 1024 Jul 8 2003 sbin
drwxr-xr-x 2 0 0 1024 Apr 17 16:55 tmp
drwxr-xr-x 3 0 0 1024 Sep 2 2004 usr
drwxr-xr-x 5 0 0 1024 Jul 8 2003 var

# /bin/themis_forkattack
attack starts
fork to raceI'm the child
1. Exec sendmail
   Sending mails...
   To... johndoe@dummy.net
   I'm the parent, child has pid 10
2. unlink
2. link

DEBUG> An attack is detected

-->attack finished
pid 9: failed 4096
# Messages: hello

[열어][환성][두벌식]
```
Performance Test

• Performance Measurement
  – Time costs of the execution of a simple program
  – Time costs of the execution of a file copy (512 bytes)
  – 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%
Table of Contents

1. Introduction and Motivation

3. Extended Role Based Access Control (E-RBAC)
   – Concept and Model of E-RBAC

5. Coloured Petri Net Formal Model for E-RBAC
   – Formal specification and verification

7. Implementation & Performance Evaluation
   – The implementation example on an Embedded System
     – Performance Evaluation

9. Conclusion
Works for the Extension

Problems
- Short-sighted control

Extension of Access Control
- Deny attacks more effectively
- Precise Authorization
- Have no formal model for specification/verification
- Performance overhead

CCPN Model
- Verify a system

Implementation
- Show an Implementation Example
- Test the performance overhead
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
Future Work

• Security for Distributed Systems
  – Active Network, Sensor Network, Grid Network

• VPC (Virtual Private Computing)
Thank you very much
Appendix: IT Layers

- IDS (x) => TOS (O)

Intrusion detection at the level of access control

- The reasons for putting security mechanisms into the lower layers[3]:
  - Higher assurance of security
  - Lower performance overheads.
Appendix: TOS, TCB, RM

• Trusted operating systems implement the concept of Trusted Computing Base (TCB)
  – TCB provides trusted environment introducing a reference monitor (RM) as the central figure [8]
    • Reference monitor mediates all accesses of a system
  – Security kernels of trusted operating systems implement the concept of reference monitoring [2]
Appendix: Disadvantages from a single abstraction layer

- Orders and identification properties are not needed for the subject abstractions such as ‘Branch Manager’, ‘Account Manager’, and ‘Teller’
- The fields and manipulation functions for the properties in implementation
  - storage overhead
  - implementation burdens
  - Semantically awkward
- Mixed abstractions in a single abstraction layer brings administrative confusions
Appendix: Petri Nets

- PN and CPN

- Petri Net

- Coloured Petri Net
Appendix: Constrained CPN

- Constrained CPN (CCPN)

Enable condition: $\text{sec_rcv} \land \text{sec_trans} \land \text{sec_snd}$
Appendix: Overheads more detail

• More Detail

About 0.01 sec (10msec) overhead for 1 execution

Overheads
- 3 level state chase
- The reaction for an attack detection

<table>
<thead>
<tr>
<th>Execution</th>
<th>Original Kernel</th>
<th>Modified Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>user CPU time</td>
<td>System CPU time</td>
</tr>
<tr>
<td>100</td>
<td>0.19</td>
<td>6.29</td>
</tr>
<tr>
<td>200</td>
<td>0.37</td>
<td>13.9</td>
</tr>
<tr>
<td>300</td>
<td>0.69</td>
<td>20.78</td>
</tr>
<tr>
<td>400</td>
<td>0.85</td>
<td>27.8</td>
</tr>
<tr>
<td>500</td>
<td>0.95</td>
<td>34.84</td>
</tr>
<tr>
<td>600</td>
<td>1.12</td>
<td>41.82</td>
</tr>
<tr>
<td>700</td>
<td>1.23</td>
<td>48.84</td>
</tr>
<tr>
<td>800</td>
<td>1.55</td>
<td>55.67</td>
</tr>
<tr>
<td>900</td>
<td>1.85</td>
<td>62.58</td>
</tr>
<tr>
<td>1000</td>
<td>1.94</td>
<td>69.54</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fork attack</th>
<th>Original Kernel</th>
<th>Modified Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>user CPU tIm</td>
<td>System CPU tIm</td>
</tr>
<tr>
<td>100</td>
<td>0.25</td>
<td>7.26</td>
</tr>
<tr>
<td>200</td>
<td>0.31</td>
<td>14.49</td>
</tr>
<tr>
<td>300</td>
<td>0.74</td>
<td>21.53</td>
</tr>
<tr>
<td>400</td>
<td>1.1</td>
<td>28.51</td>
</tr>
<tr>
<td>500</td>
<td>1.24</td>
<td>35.93</td>
</tr>
<tr>
<td>600</td>
<td>1.5</td>
<td>42.97</td>
</tr>
<tr>
<td>700</td>
<td>1.69</td>
<td>50.25</td>
</tr>
<tr>
<td>800</td>
<td>2.02</td>
<td>57.58</td>
</tr>
<tr>
<td>900</td>
<td>2.42</td>
<td>64.65</td>
</tr>
<tr>
<td>1000</td>
<td>2.48</td>
<td>71.94</td>
</tr>
</tbody>
</table>
## Appendix: Access Matrix

- Harrison-Ruzzo-Ullman Model
- Every subject is an object, too

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>O₁</th>
<th>O₂</th>
<th>O₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>Control</td>
<td>Own</td>
<td>Suspend Resume</td>
<td>Own</td>
<td>Own</td>
<td></td>
</tr>
<tr>
<td>S₂</td>
<td></td>
<td>Control</td>
<td></td>
<td>Extend</td>
<td>Own</td>
<td></td>
</tr>
<tr>
<td>S₃</td>
<td></td>
<td></td>
<td>Control</td>
<td>Read Write</td>
<td>Write</td>
<td>Read</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Access Matrix in HRU model**
Appendix: Previous RBAC Extensions

- TBAC: Extending RBAC to enforce Separation of Duty
  - Separation of Duty
    - Defined between two or more roles
    - Roles are defined as transactions
  - The main difference with SOD policies
    - They cannot control each user’s behaviors
Appendix: RBAC Modeling of CCPN

- RBAC representation of CCPN
  - Represented with one place and one transition
  - Tokens as many as users

Only traditional access conditions will be checked
Appendix: The Reasons of Embedded OS

• To reflect the current trends of researches
  – Embedded operating systems for the digital appliances

• The advantage of E-RBAC in the embedded systems
  – Some embedded systems have not enough resources to run IDS systems
  – E-RBAC introduce intrusion detection technique as well the traditional access control at kernel-level
    • Performance is high
    • Resource overhead is low
Appendix: Pros. & Cons. of E-RBAC

• Advantages
  – Blocks various attacks consist of ordinary operations
  – Precise authorization

• Disadvantages
  – Overheads due to the additional constraints

• Number of relations
  – # of relations in RBAC: |U| * |R| + |R| * |P|
  – # of relations in E-RBAC: |U| * |R’| + |R’| * |B’| + |B’| * |P|
    • to have small # relations, |B| < |P|/2