Security Research Group, Dept. of Info. & Comm.
Gwangju Institute of Science and Technology

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

Gwangju Institute of Science and Technology

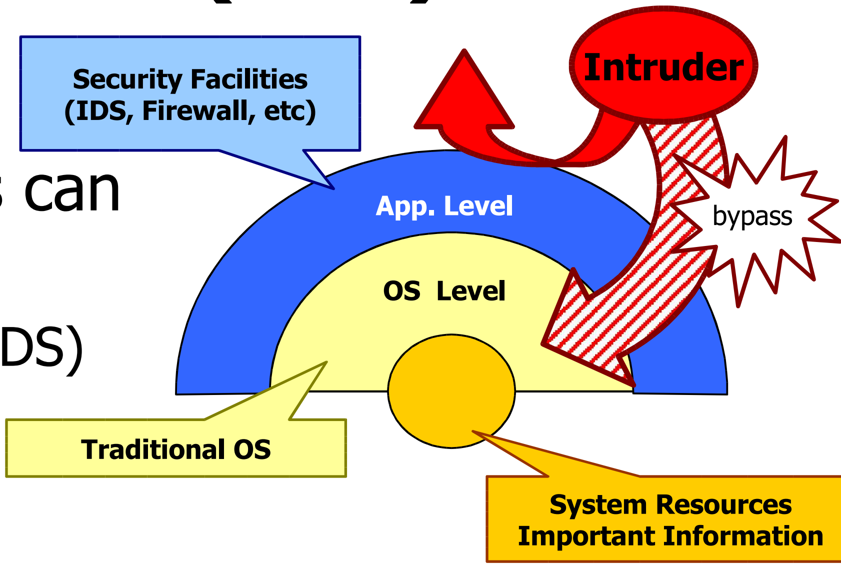
SHIN, Wook

Dept. of Info. & Comm., GIST, 1 Oryong-dong Buk-gu Gwangju 500-712, Korea

Table of Contents

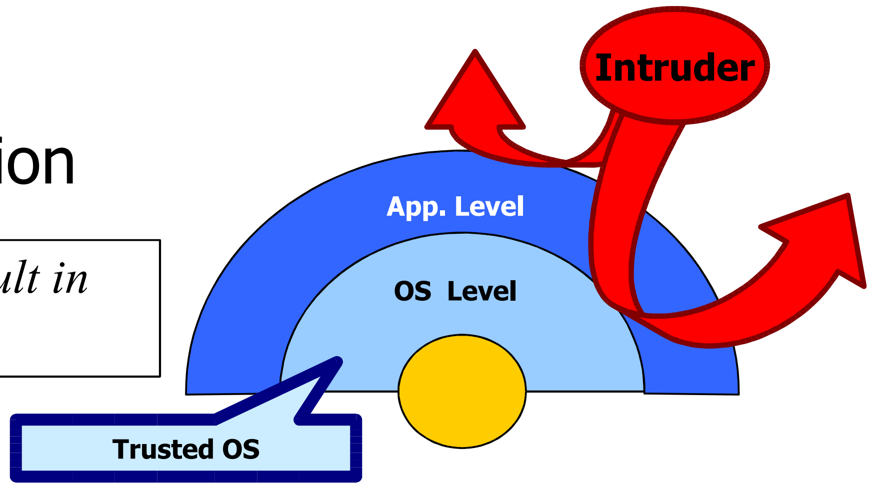
Trusted Operating System (TOS)

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



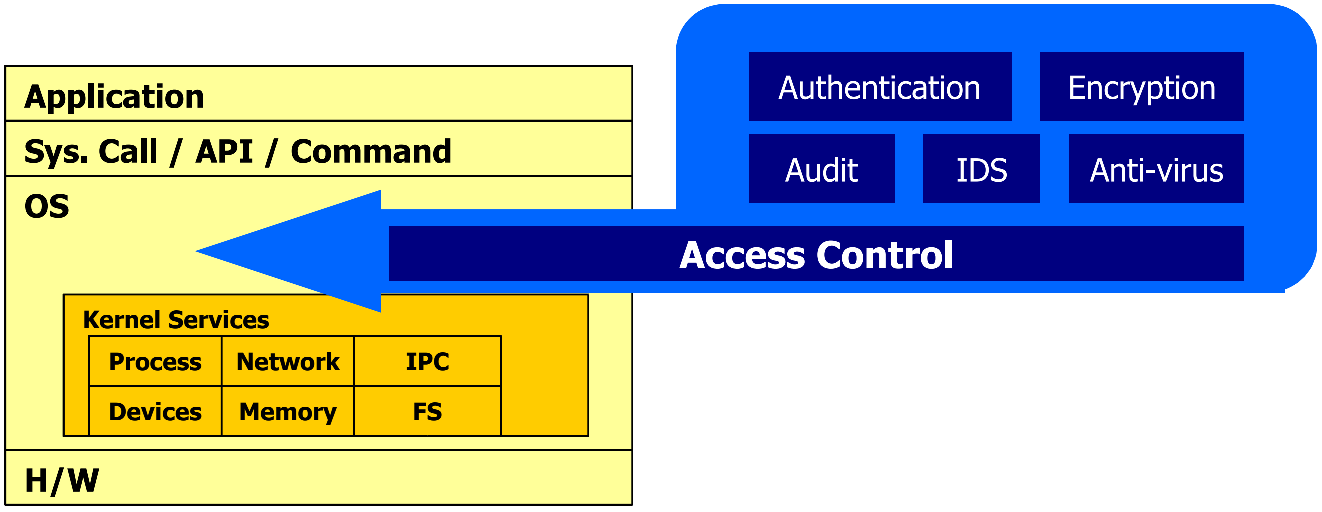
- TOS is an even more fundamental security solution

> *"Without TOS, all security efforts result in **Fortress built upon sand**"*[2]



TOS and Access Control

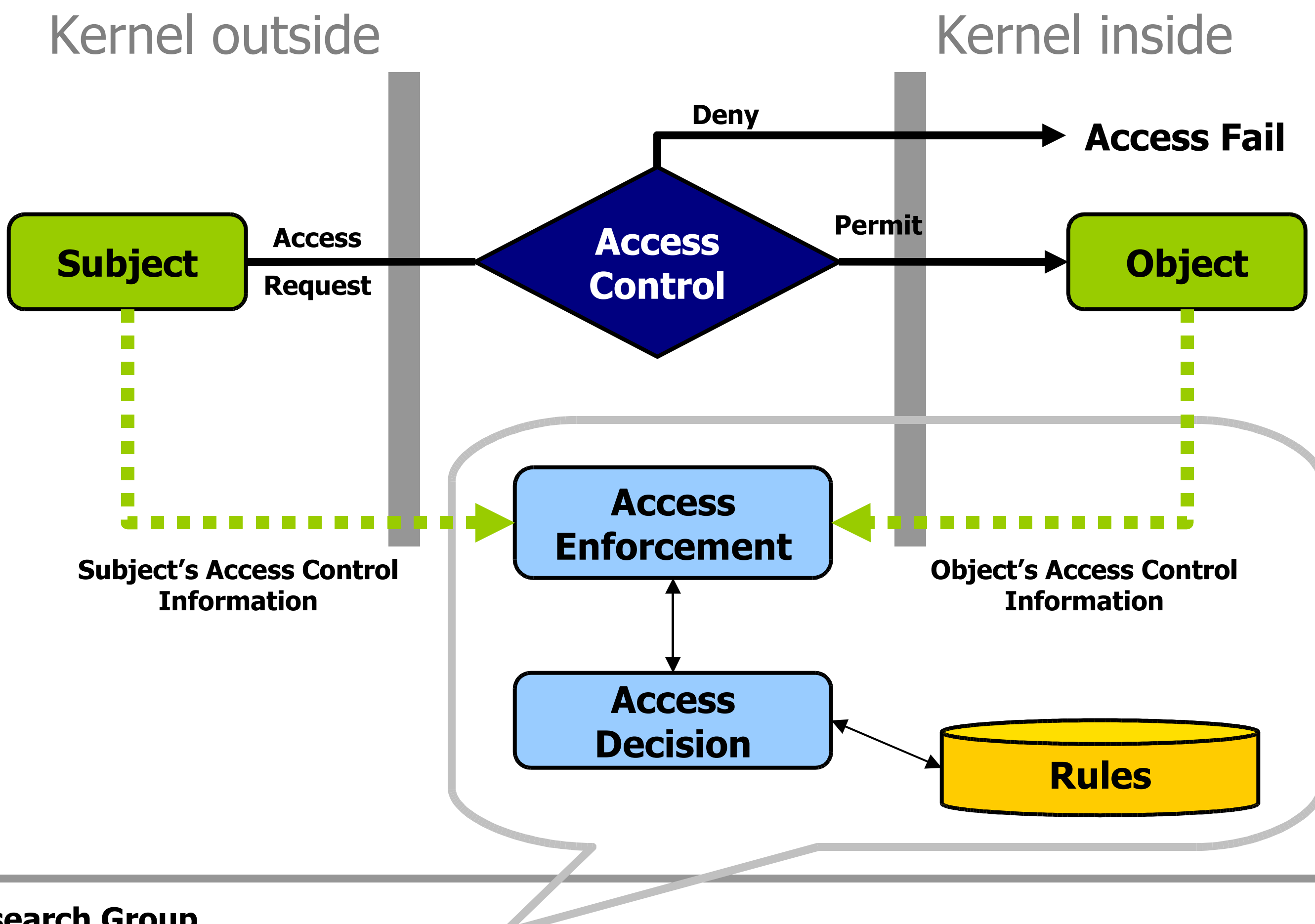
- What is the Trusted Operating System (TOS)?



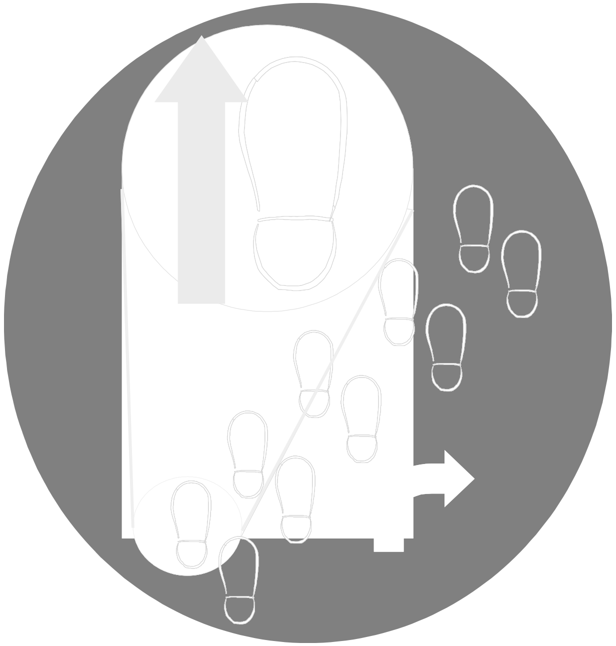
TOS := traditional OS services + security services

Access Control

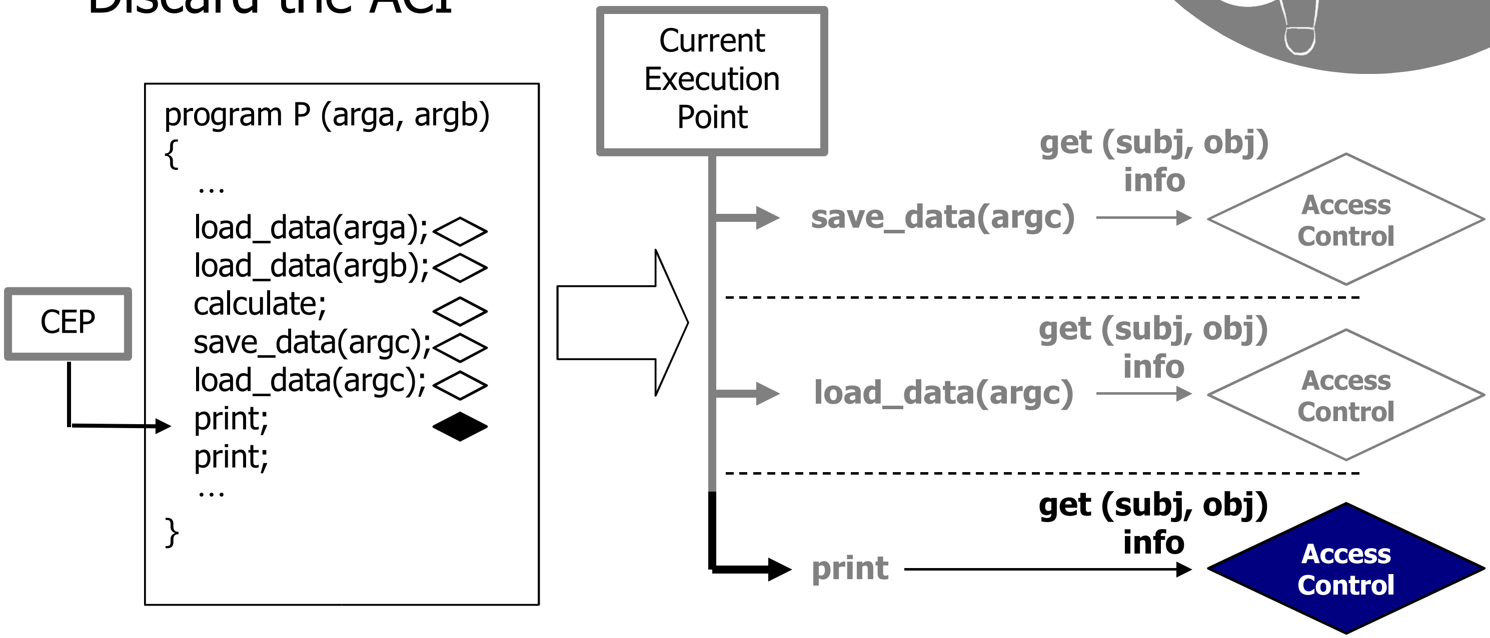
- Access Control: the core function of the TOS



Current Access Controls

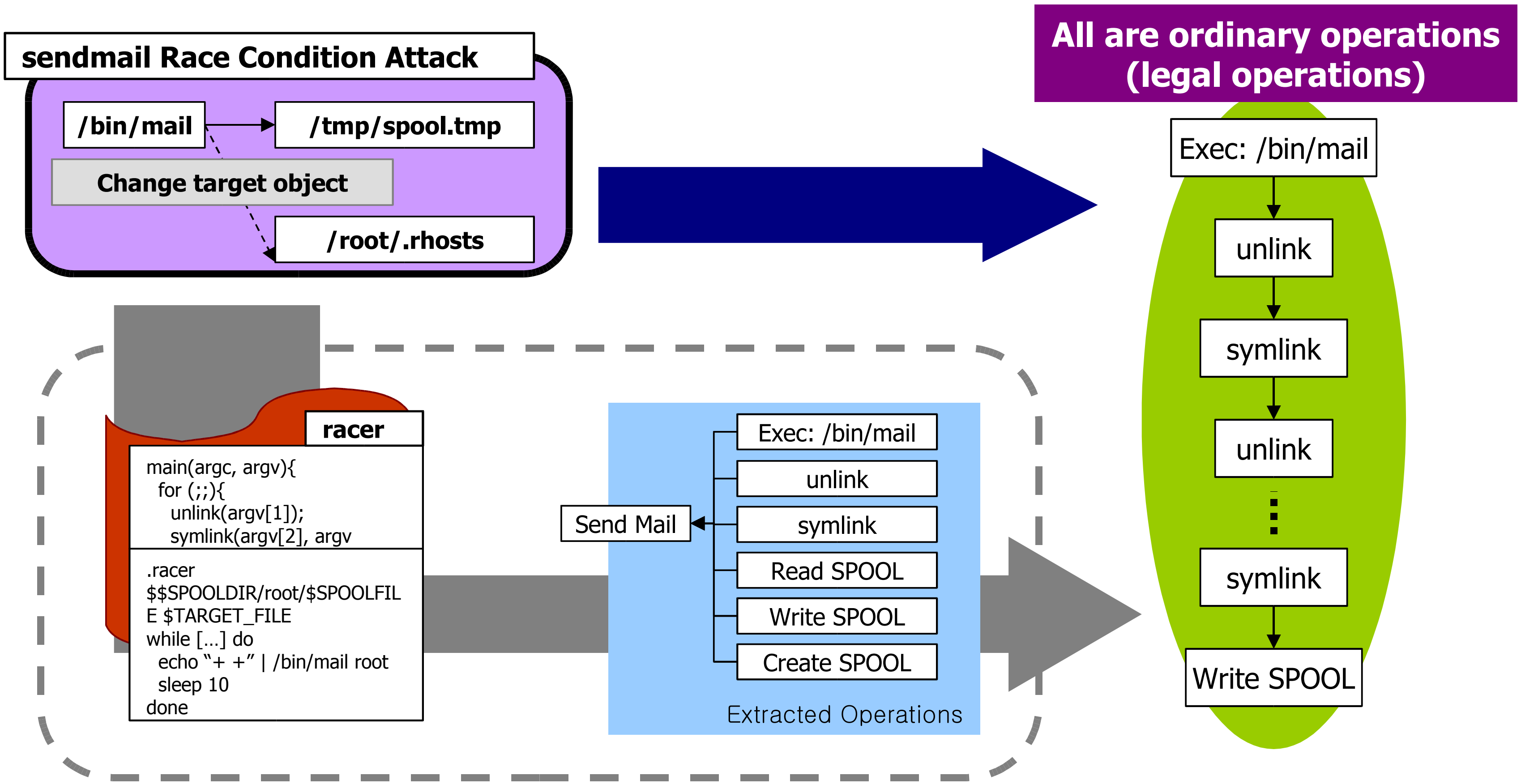


- The process of access control
  - Gather Access Control Information(ACI) at the moment of each access
  - Make a decision based on the ACI
  - Discard the ACI



Insufficiency of Current Access Controls

- Current access controls cannot block some kinds of attacks



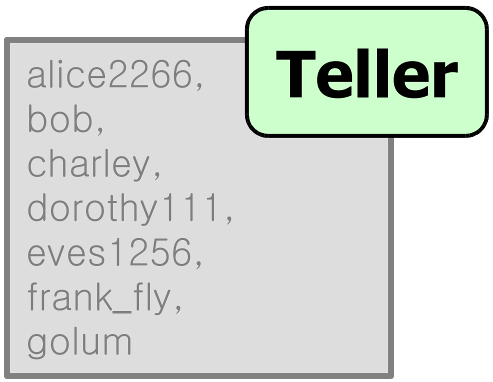
Summary of the Motivation

- 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

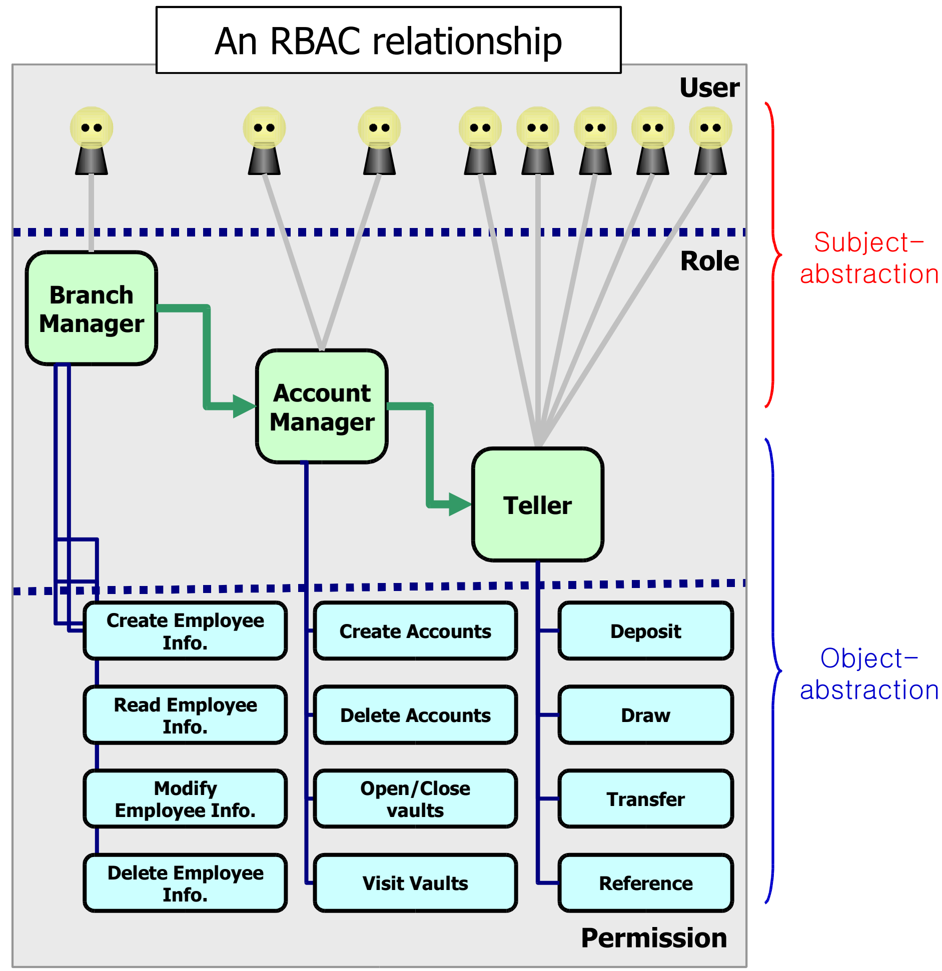
Role Based Access Control

- 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

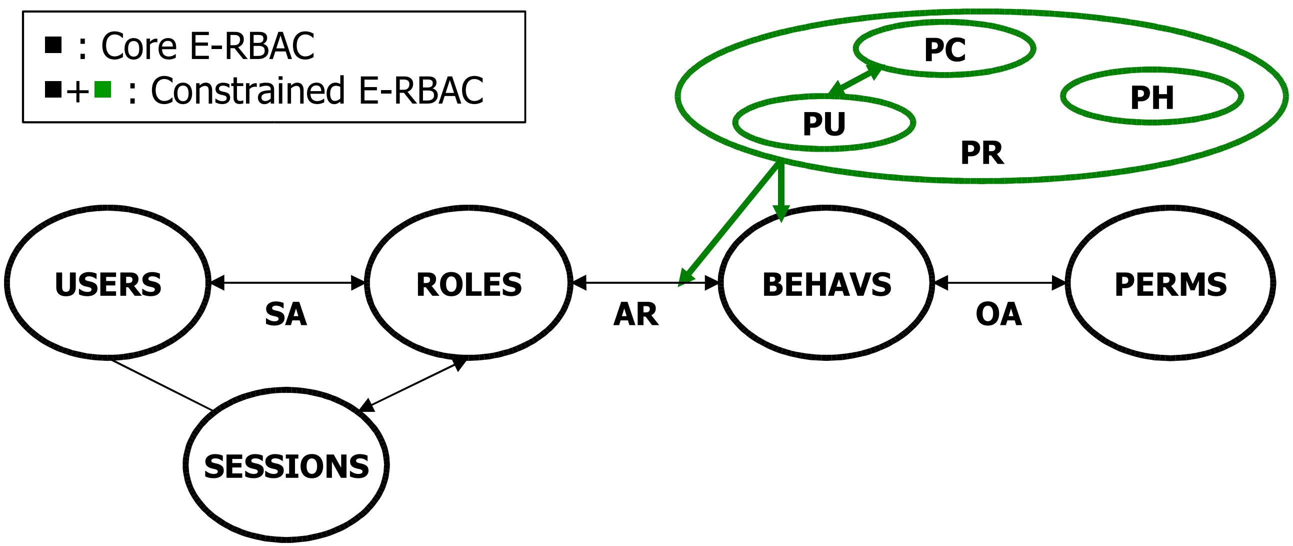
- Subject Abstraction and Object Abstraction
  - Roles: a set of users (subject-abstraction)
    Ex) Secretaries := {John, Michael, Tom}
  - Behaviors: a set of permissions (object-abstraction)
    Ex) FileOpSet := { f_open, f_close, f_read, f_write}

- 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



Core E-RBAC

Access Control Models
- describe the access control **concept**,
- specify access control **policy**,
- and specify **requirements of a system** without ambiguity

- Core E-RBAC Model
  - USERS, ROLES, BEHAVS, and PERMS
  - SESSIONS
  - SA $\subseteq$ USERS $\times$ ROLES
  - OA $\subseteq$ BEHAVS $\times$ PERMS
  - AR $\subseteq$ ROLES $\times$ BEHAVS
  - assigned_users: (r: ROLES) $\rightarrow$ 2^{USERS}, the mapping from a role r onto a set of users
    - Formally: assigned_agents(r) = { a $\in$ USERS | (a, r) $\in$ SA}
  - assigned_permissions: (b: BEHAVS) $\rightarrow$ 2^{PERMS}, the mapping of behavior b onto a set of permissions
    - Formally: assigned_permissions(b) = { p $\in$ PERMS | (p, b) $\in$ OA}

Core E-RBAC

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

- assigned_behaviors: (r: ROLES) $\rightarrow$ 2^{BEHAVS}, the mapping of role r onto a set of behaviors
  - Formally: assigned_behaviors(r) = { b $\in$ BEHAVS | (r, b) $\in$ AR}

- avail_session_permissions: (s: SESSIONS) $\rightarrow$ 2^{PERMS}, the mapping from a session s onto a set of permissions
  - Formally: avail_session_permissions(s) =

$$\bigcup_{b \in \text{assigned\_behaviors}(r)} \text{assigned\_permissions}(b)$$

(,where r$\in$session_roles(s))

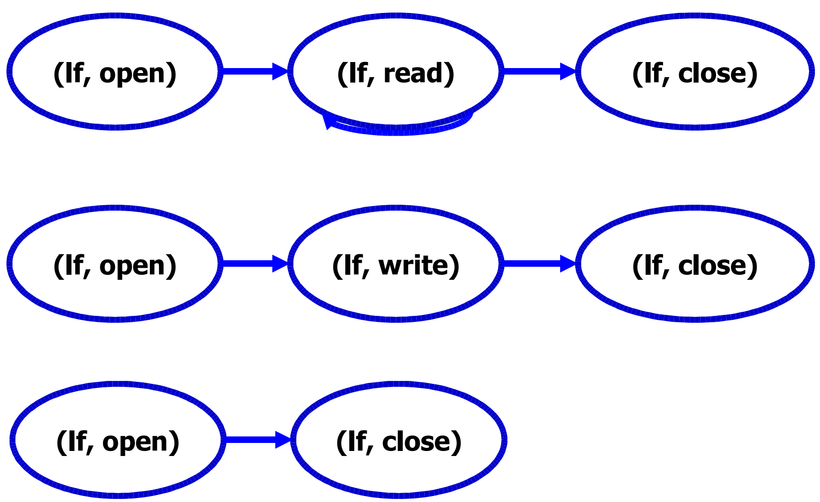
Constrained E-RBAC

- 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

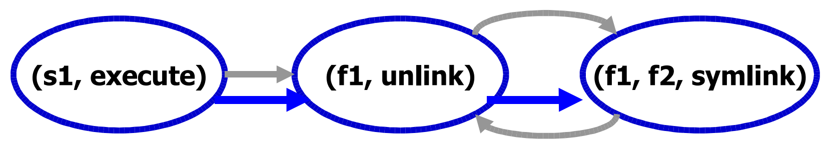
Modeling Behaviors

- Normal and Attack behaviors can be described

Positive	The permitted behavioral patterns of log file management



Negative	The prohibited behavioral pattern of the race condition attack



Modeling Behaviors with PR

- Normal and Attack behaviors can be expressed with PR elements

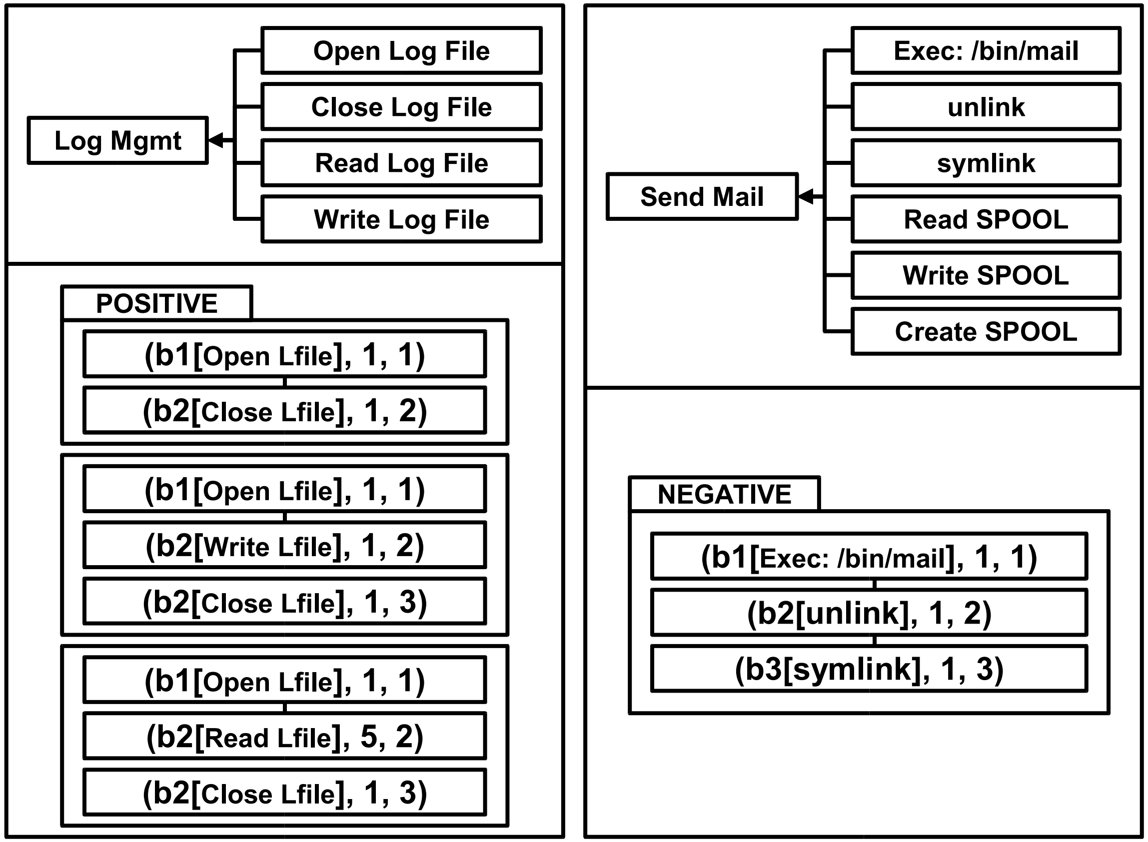


Table of Contents

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

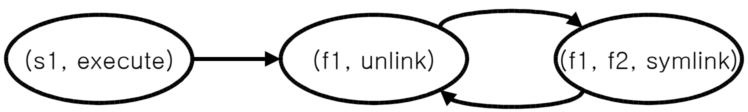
- Formal methods in TOS developments
  - for correct design and implementation
    - Specifying requirements and systems
    - Specifying security policies, models, and implemented systems
    - Determining how well a specification meets the requirements
  - **for security administration**
    - **check 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
    - $R_{u1} = R_{sub1} \cup R_{sub2}$
    - avail_session_permissions(s) = $\bigcup_{b \in assigned\_behaviors(r)}$ assigned_permissions(b)

      (where $r \in$ 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

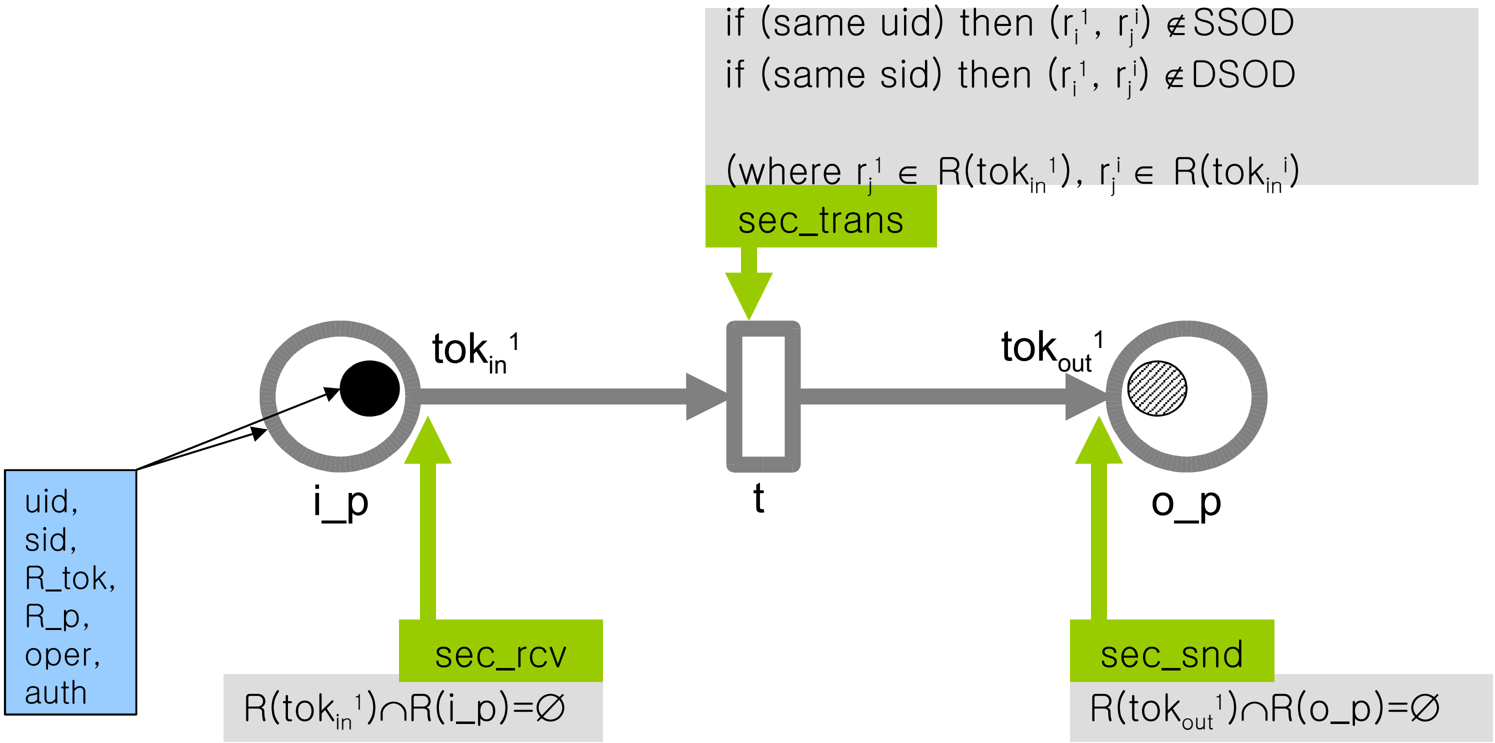
- Coloured Petri Nets (CPN)
  - Modeling concurrent systems
  - The state machine based formalism, but also supports
    - Type definitions: Token, Place have data type(color)
    - Type Manipulations: Transition, Arc have expression
  - Other advantages [7]
    - CPN support hierarchical structures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

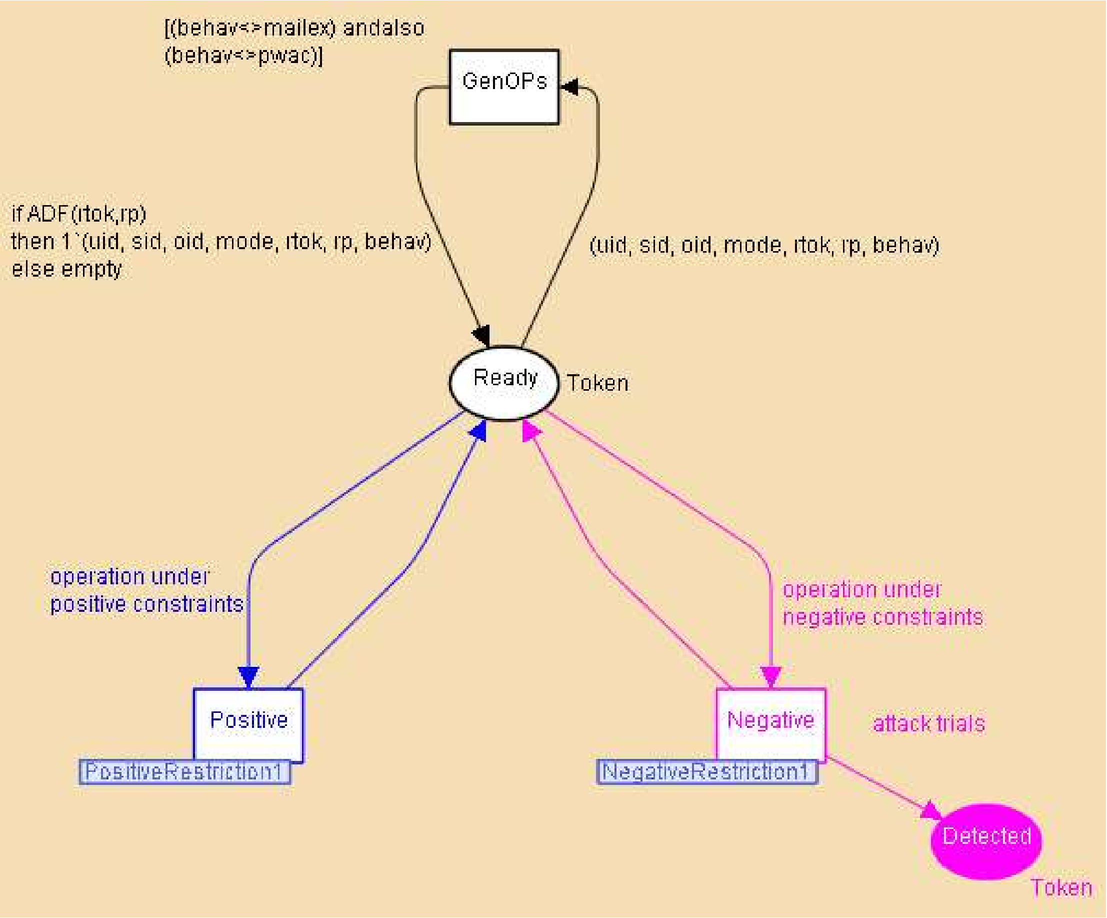
- Constrained CPN (CCPN)



  – Enable condition: sec_rcv $\wedge$ sec_trans $\wedge$ sec_snd

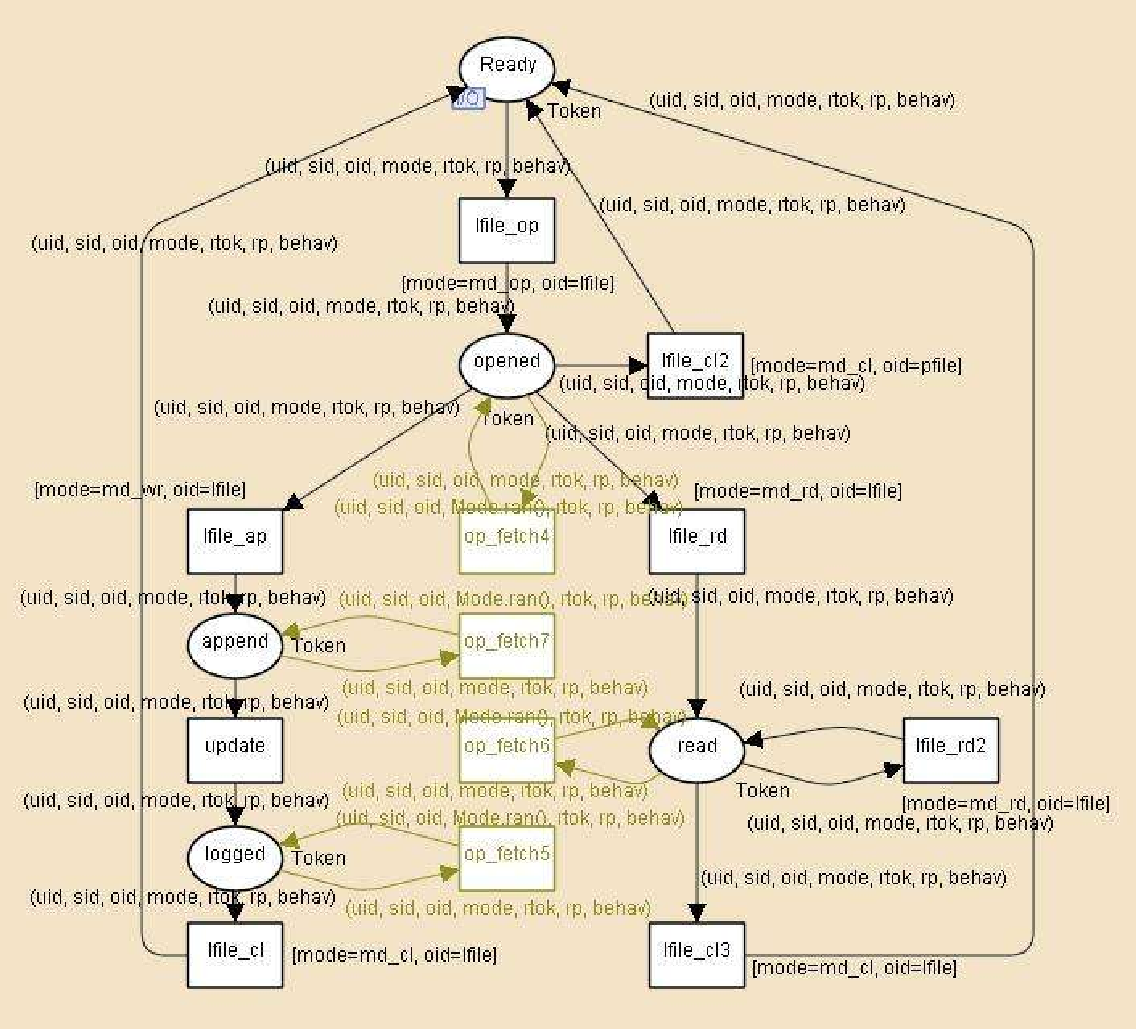
CCPN Example

- Overall Diagram



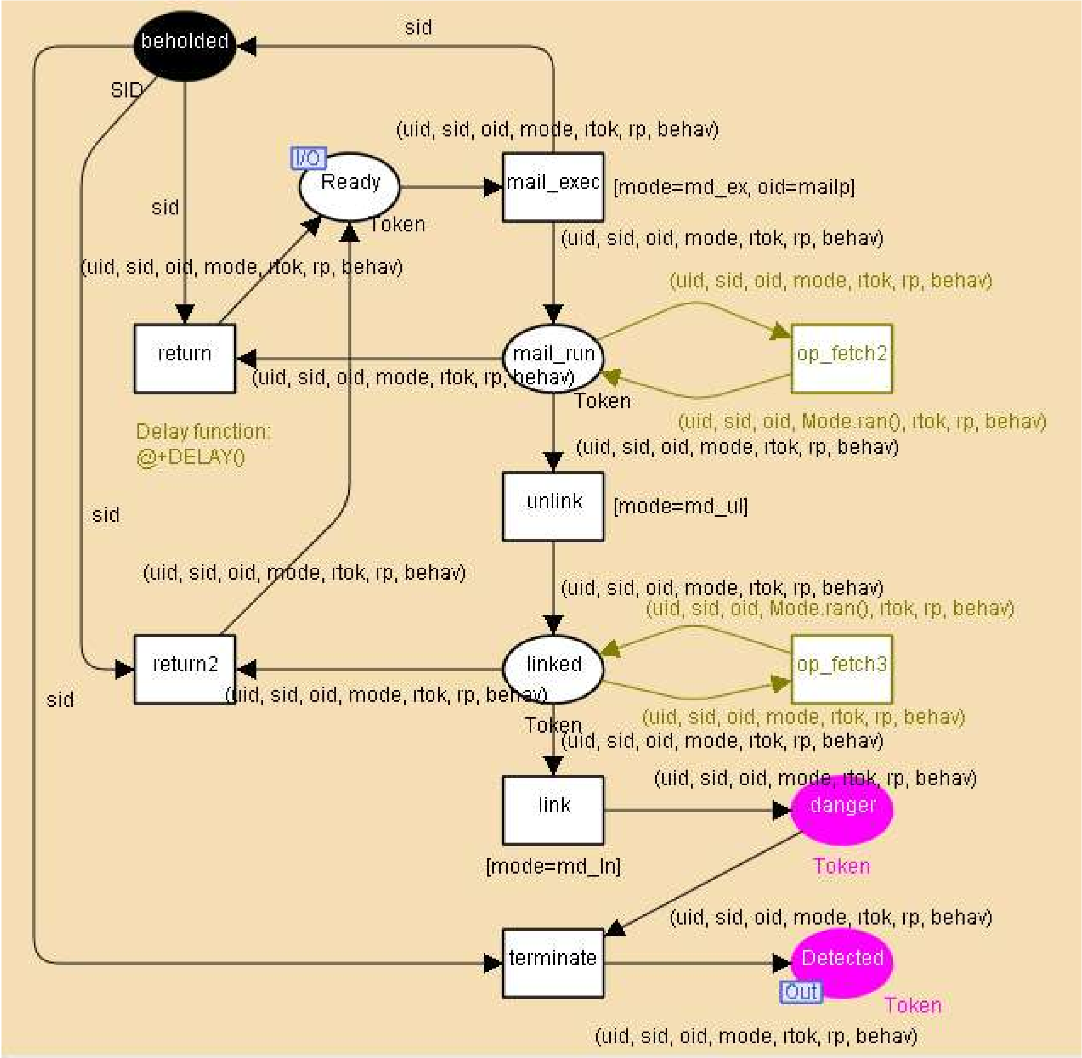
CCPN Example

- Positive PC



CCPN Example

- Negative PC



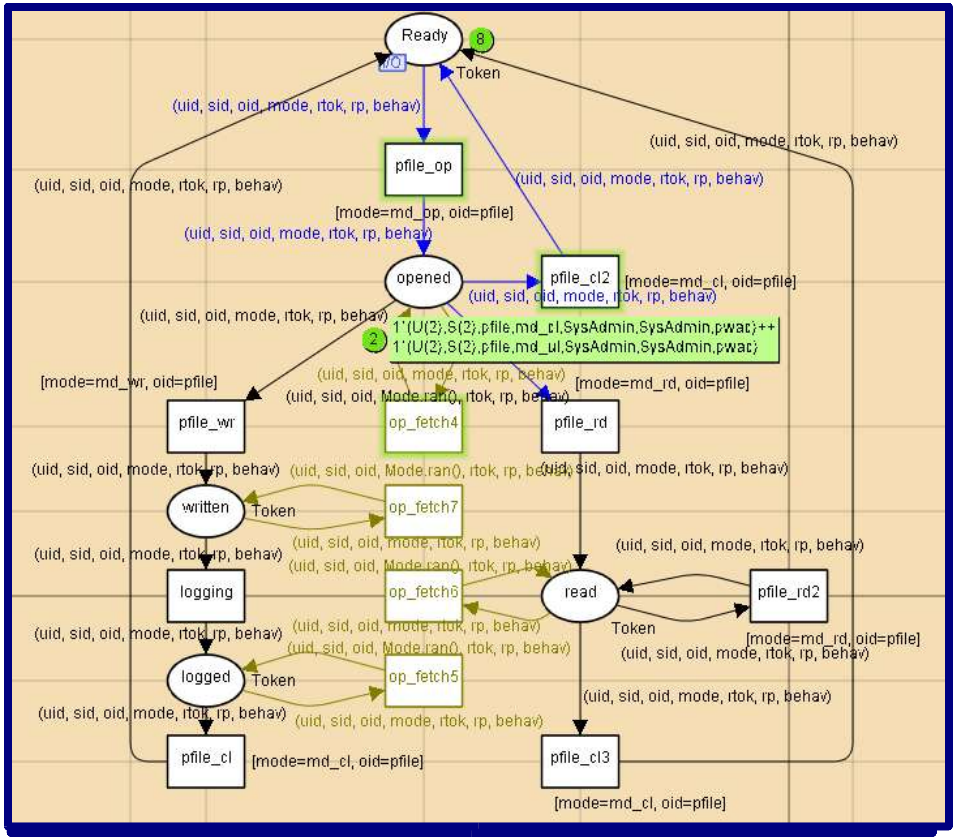
Testing a configuration with CPN

- Analysis
  - Simulation
  - Formal Analysis

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

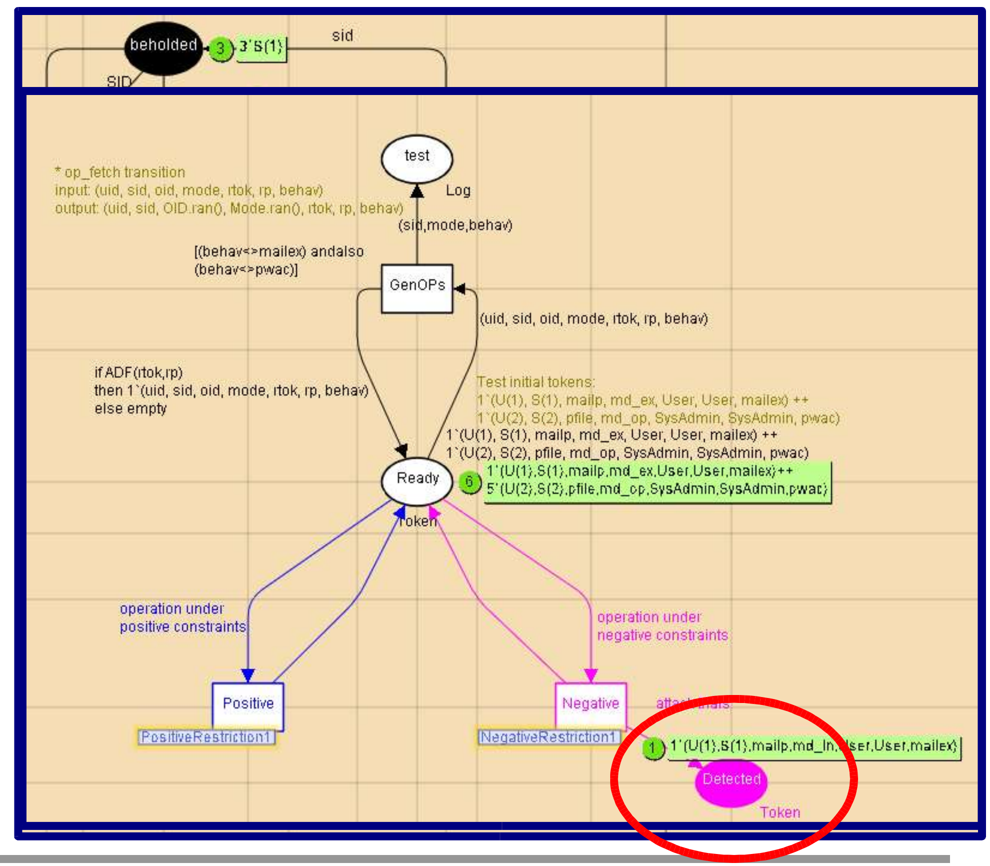
Simulation Example

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



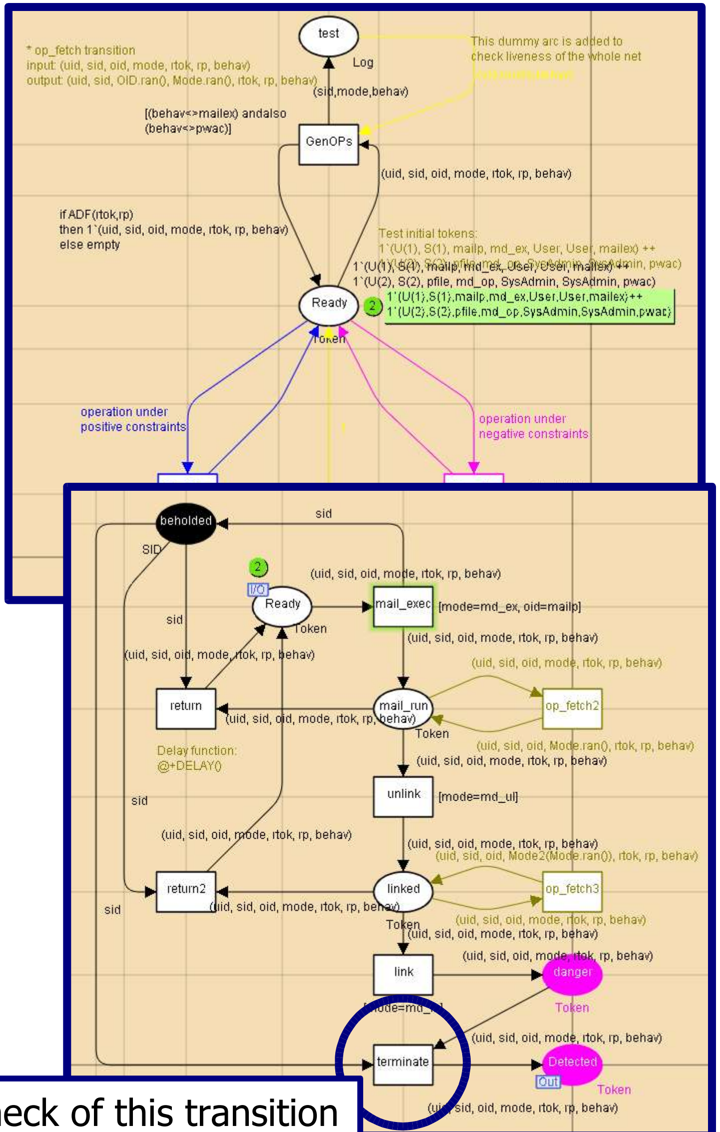
Simulation Example

- 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



Liveness check of this transition

Formal analysis results

```
Liveness Properties
---------------------------------------------------
 Dead Markings:  None
 Dead Transitions Instances:

 NegativeRestriction1'link 1
 NegativeRestriction1'mail_exec 1
 NegativeRestriction1'op_fetch2 1
 NegativeRestriction1'op_fetch3 1
 NegativeRestriction1'return 1
 NegativeRestriction1'return2 1
 NegativeRestriction1'terminate 1
 NegativeRestriction1'unlink 1
 Overview'GenOPs 1
 Overview'dummy 1
 PositiveRestriction1'logging 1
 PositiveRestriction1'op_fetch5 1
 PositiveRestriction1'op_fetch6 1
 PositiveRestriction1'op_fetch7 1
 PositiveRestriction1'pfile_cl 1
 PositiveRestriction1'pfile_cl2 1
 PositiveRestriction1'pfile_cl3 1
 PositiveRestriction1'pfile_rd 1
 PositiveRestriction1'pfile_rd2 1
 PositiveRestriction1'pfile_wr 1
 Live Transitions Instances:

 PositiveRestriction1'op_fetch4 1
```

Original configuration

```
Liveness Properties
---------------------------------------------------
 Dead Markings:  6 [359,221,169,144,109,...]
 Dead Transition Instances:

 NegativeRestriction1'link 1
 NegativeRestriction1'terminate 1
 Overview'GenOPs 1
 Overview'dummy 1
 Live Transitions Instances: None
```

Modified configuration
(Prohibit the unlink operation)

Table of Contents

Implementation Environments

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

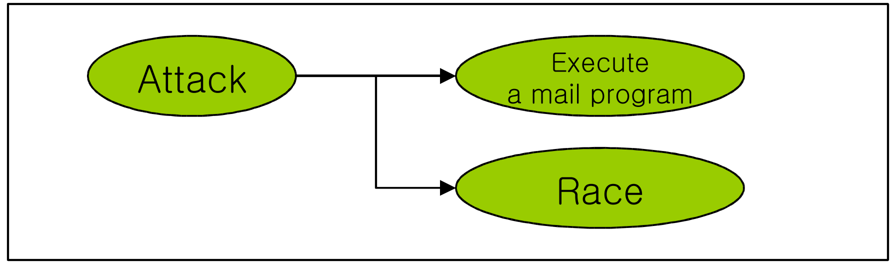
Implementation Structure

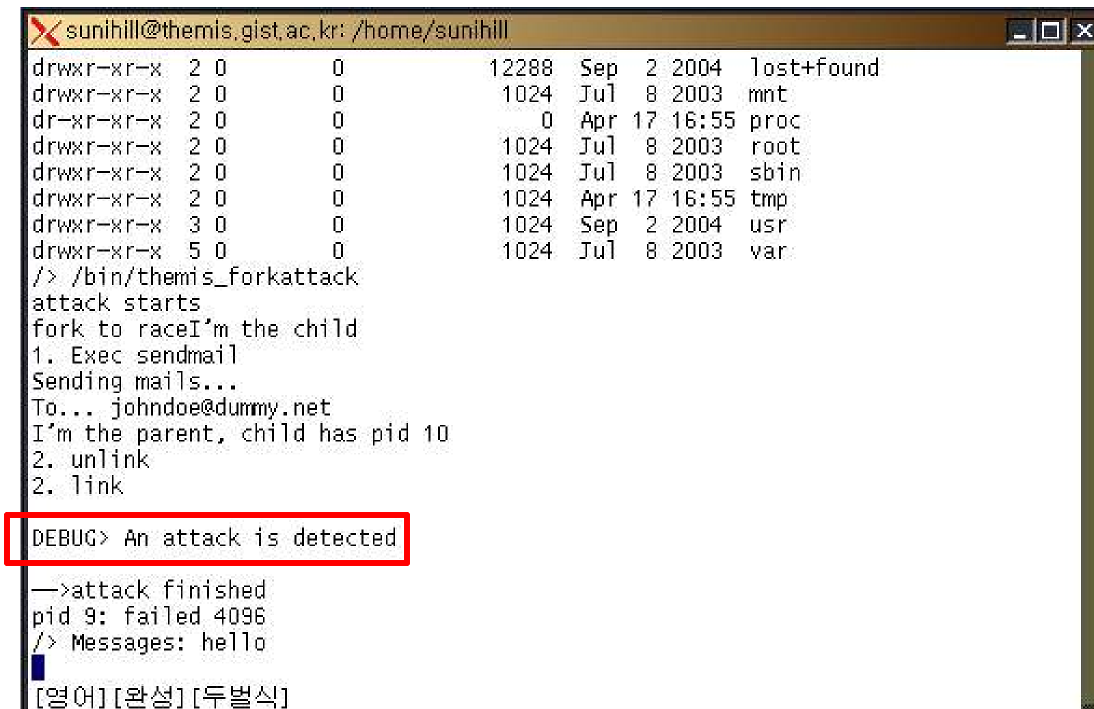
- For traditional Access Control (Core E-RBAC)
  - Process
    - Permission vector (Information of roles)
  - File
    - Permission vector (Information of roles)
  - ADF
    - Comparing the set of roles

- 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





Performance Test

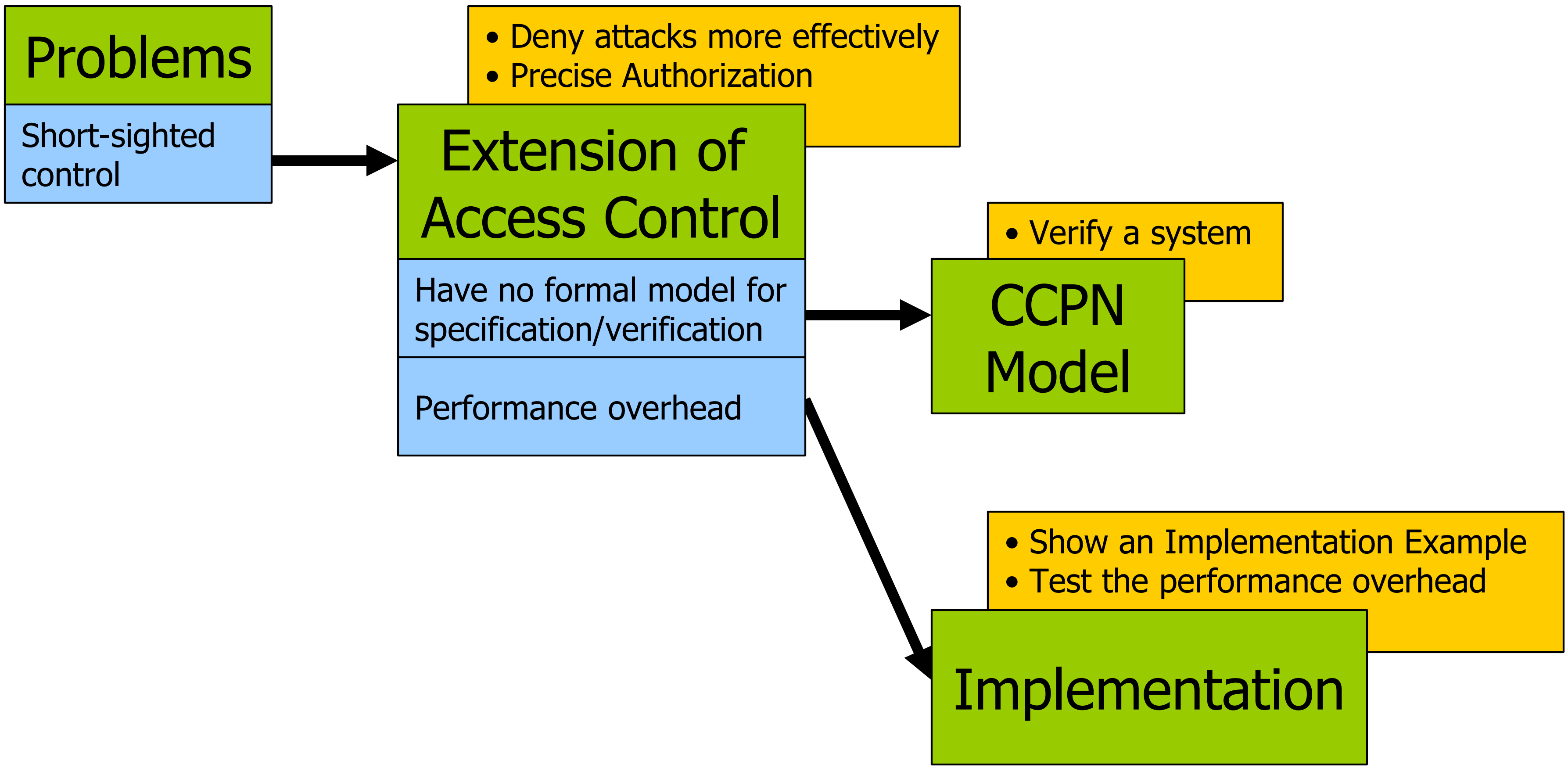
- Performance Measurement
  - Time costs of the execution of a simple program
  - Time costs of the execution of a file copy (512bytes)
  - Time costs of the execution of a simple program that have procedural constraints
- Results

Our system: 10 % overhead

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

Table of Contents

Works for the Extension

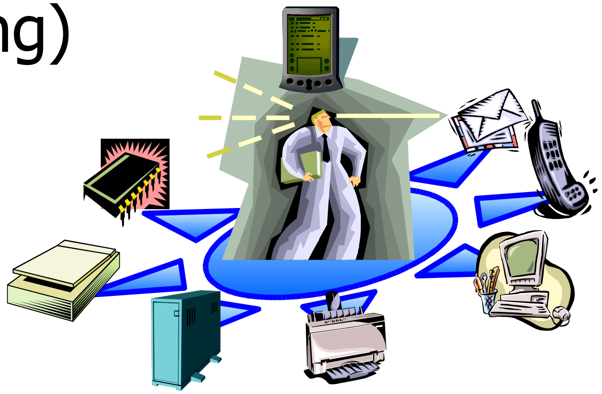


Conclusion

- 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

Bibliography

[1] T. Ptacek and T. Newsham, "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", 1998.

[2] D. Baker, "Fortresses built upon sand", In Proceedings of the New Security Paradigms Workshop, 1996.

[3] D. Gollmann, "Computer Security", John Wiley & SONS, 1999.

[4] CPN Tools: http://wiki.daimi.au.dk/cpntools/

[5] M.Gasser, "Building a Secure Computer System", van Nostrand Reinhold, 1988.

[6] M. Bishop, “Computer Security: Art and Science”, Addison Wesley Professional, 2003

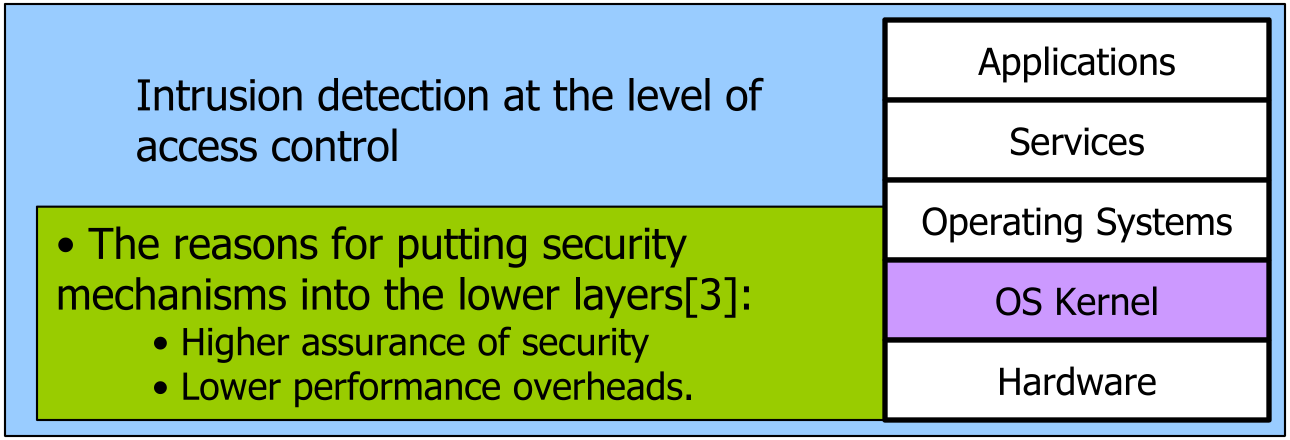
[7] K. Jansen, “Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use”, Vol. 1-2, Springer Verlag, 1992

[8] Department of Defense (U.S.), "Department of Defense Trusted Computer System Evaluation Criteria", Department of Defense Standard(Dod 5200.28-STD), Library Number S225, 711, December 1985.

[9] D. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli, "Proposed standard for Role Based Access Control,“ ACM Transactions on Information and System Security, vol. 4, no. 3 (August, 2001) - draft of a consensus standard for RBAC.

Appendix: IT Layers

- IDS (x) => TOS (O)



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

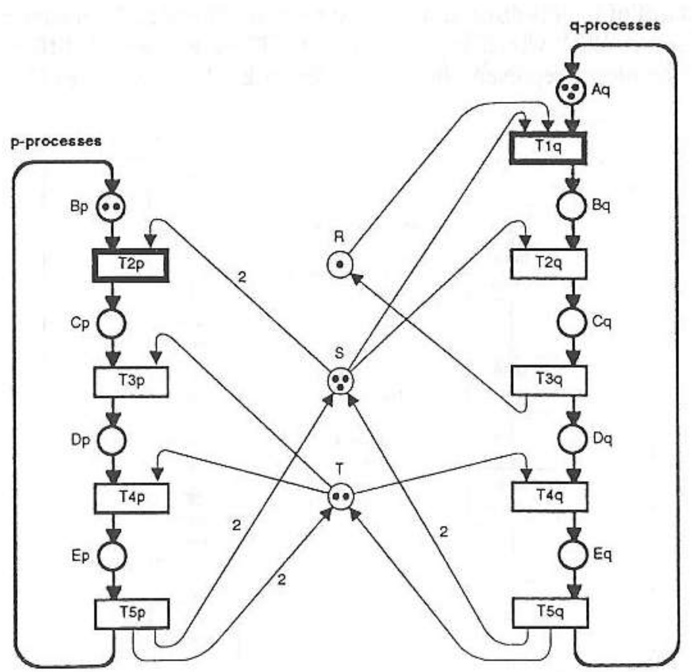
- Orders and identification properties are not needed for the subject abstractions such as ‘Branch Manager’, ‘Account Manager’, and ‘Teller’
- The fields and manipulation functions for the properties in implementation
  - storage overhead
  - implementation burdens
  - Semantically awkward

- Mixed abstractions in a single abstraction layer brings administrative confusions

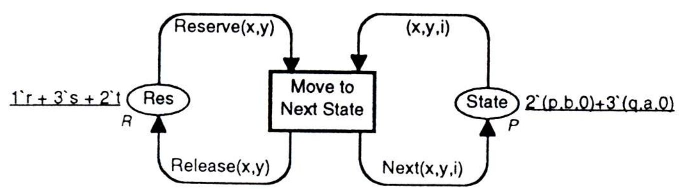
Appendix: Petri Nets

- PN and CPN

- Petri Net

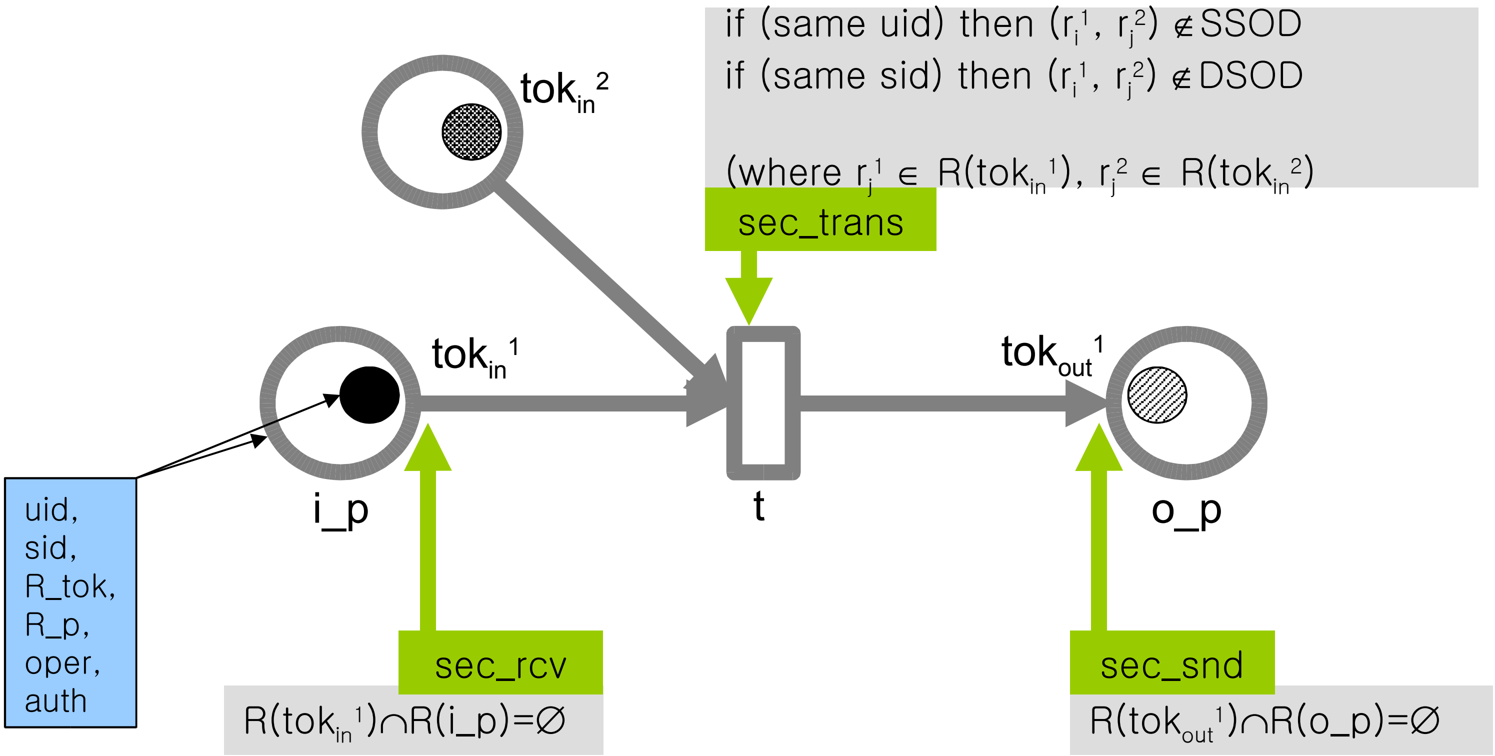


- Coloured Petri Net



Appendix: Constrained CPN

- Constrained CPN (CCPN)



– Enable condition: sec_rcv $\wedge$ sec_trans $\wedge$ sec_snd

Appendix: Overheads more detail

- More Detail

Execution						
	Original Kernel			Modified Kernel		
	user CPU tim	system CPU tim	Sum	user CPU tim	System CPU Tim	Sum
100	0.19	6.29	6.48	0.19	6.98	7.17
200	0.37	13.9	14.27	0.37	13.95	14.32
300	0.69	20.78	21.47	0.57	20.94	21.51
400	0.85	27.8	28.65	0.75	27.91	28.66
500	0.95	34.84	35.79	1.04	34.79	35.83
600	1.12	41.82	42.94	1.05	41.96	43.01
700	1.23	48.84	50.07	1.56	48.61	50.17
800	1.55	55.67	57.22	1.45	55.88	57.33
900	1.85	62.58	64.43	1.74	62.69	64.43
1000	1.94	69.54	71.48	1.9	69.73	71.63
[illegible]						
	Original Kernel			Modified Kernel		
	user CPU tim	System CPU Tin	Sum	user CPU tim	System CPU Tim	Sum
100	0.01	0.32	0.33	0.05	0.32	0.37
200	0.15	0.56	0.71	0.1	0.61	0.71
300	0.23	0.82	1.05	0.19	0.87	1.06
400	0.27	1.12	1.39	0.23	1.17	1.4
500	0.28	1.46	1.74	0.32	1.44	1.76
600	0.35	1.73	2.08	0.39	1.71	2.1
700	0.42	2	2.42	0.41	2.03	2.44
800	0.37	2.39	2.76	0.48	2.3	2.78
900	0.51	2.55	3.06	0.61	2.52	3.13
1000	0.61	2.87	3.48	0.53	2.97	3.5
fork attack						
	Original Kernel			Modified Kernel		
	user CPU tim	System CPU Tin	Sum	user CPU tim	System CPU Tim	Sum
100	0.25	7.26	7.51	0.25	7.9	8.15
200	0.31	14.49	14.8	0.46	15.79	16.25
300	0.74	21.53	22.27	0.74	23.57	24.31
400	1.1	28.51	29.61	1.04	31.24	32.28
500	1.24	35.93	37.17	1.11	39.4	40.51
600	1.5	42.97	44.47	1.42	47.13	48.55
700	1.69	50.25	51.94	1.63	54.98	56.61
800	2.02	57.58	59.6	1.83	62.85	64.68
900	2.42	64.65	67.07	2.29	70.96	73.25
1000	2.48	71.94	74.42	2.56	78.69	81.25

About 0.01 sec (10msec) overhead for 1 execution

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

Appendix: Access Matrix

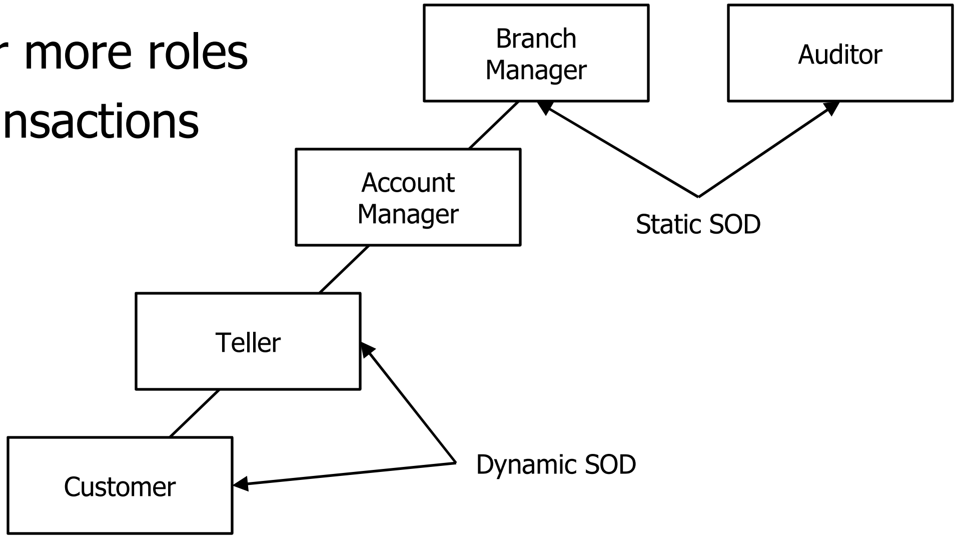
- Harrison-Ruzzo-Ullman Model
- Every subject is an object, too

	S_1	S_2	S_3	O_1	O_2	O_3
S_1	Control	Own Suspend Resume		Own	Own	
S_2		Control			Extend	Own
S_3			Control	Read Write	Write	Read
…						

Access Matrix in HRU model

Appendix: Previous RBAC Extensions

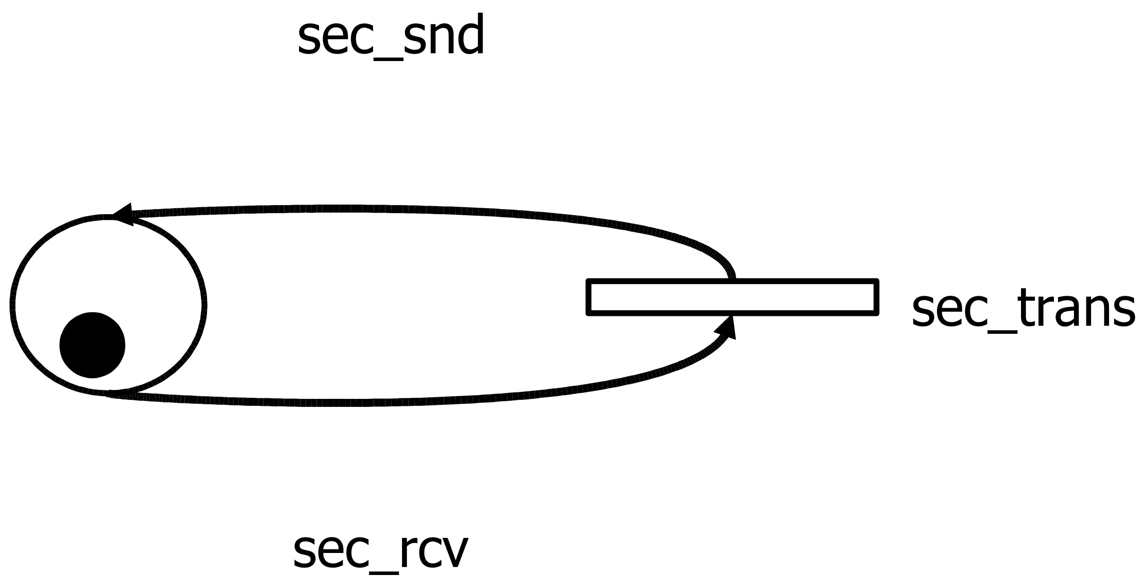
- TBAC: Extending RBAC to enforce Separation of Duty
  - Separation of Duty
    - Defined between two or more roles
    - Roles are defined as transactions



  - 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, $|B| < |P|/2$