به نام خدا

Probabilistic Noninterference

Seyed Mohammad Mehdi Ahmadpanah

smahmadpanah@aut.ac.ir

Supervisor

Dr. Mehran S. Fallah

Formal Security Lab.

Amirkabir University of Technology

August 13, 2016

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی امیر کبیر (پلی تکنیک تهران)

Outline

- Introduction
- Preliminaries
- Literature Review
- Probabilistic Noninterference for Multithreaded Programs
- Conclusion

Introduction

- Noninterference
  - Confidentiality
  - Extensional
  - Nondeterminism
    - Multi-Threading and Thread Scheduling
  - Possibilistic vs. Probabilistic
- Motivation

Preliminaries

- Bisimulation
  - Weak Bisimulation
- Noninterference Notions
- Attacker Observation (internal, external)
- Classification of Flows
- Type System

Bisimulation

- Bisimulation
  - Given a labelled state transition system $(S, \Lambda, \rightarrow)$
  - A bisimulation relation is a binary relation R over S ($R \subseteq S \times S$)
  - R is a bisimulation if for every pair of elements p,q in S with (p,q) in R, for all α in Λ:
  - For all p' in S : $p \xrightarrow{\alpha} p'$ implies that there is a q' in S s.t. $q \xrightarrow{\alpha} q'$ and $(p', q') \in R$,
  - And symmetrically vice versa.
  - Written $p \sim q$

Bisimulation

- Weak Bisimulation
  - β is a, $\bar{a}$, ε for action name a, then

    If $p \xrightarrow{\beta} p'$ then $q \xrightarrow{\beta} q'$ for some q' s.t. $(p', q') \in R$

    and

    If $q \xrightarrow{\beta} q'$ then $p \xrightarrow{\beta} p'$ for some p' s.t. $(p', q') \in R$
  - Written $p \approx q$

Noninterference Notions

- Noninterference Notions
  - Self Isomorphism
  - Discreetness (highness)
  - Self Strong Bisimilarity (Strong Security)
  - 01-Bisimilarity
  - Weak Bisimilarity
- Attacker
  - External (stop-watch) vs. Internal
  - Program Counter

Classification of Flows

- Direct
  - `l := h`
- Indirect
  - `if h=1 then l:=1 else l:=0`
- Termination Behavior
  - `if h=1 then loop`
- Probabilistic
  - `h:=h%2; (l:=h` $\square_{\frac{1}{2}}$ (l:=0 $\square_{\frac{1}{2}}$ l:=1))
- Externally Observable Timing
  - `if h=1 then sleep 100`
- Internally Observable Timing
  - `((if h=1 then sleep 100);l:=1)|l:=0`

Desired Properties

- Desired properties of an approach to information flow for multithreaded programs [10]
  - Permissiveness
  - Scheduler-Independence
  - Standard Semantics
  - Language Expressiveness
  - Practical Enforcement

Literature Review

- Probabilistic Noninterference in a Concurrent Language – Volpano and Smith – 1999
  - First notion of Probabilistic NI
  - Internal program behavior
  - A type system
  - Uniform scheduler
  - Every conditional with a guard containing high variables must be executed atomically
    - *protect*
  - The guard of a while-loop must be L

Literature Review (Cont.)

- Transforming out Timing Leaks – Agat - 2000
  - Timing leakages to external observer
    - covert timing channel
  - A type system + A type-directed transformation
  - By padding with dummy instructions where needed

Literature Review (Cont.)

- Probabilistic Noninterference for Multi-threaded Programs–Sabelfed and Sands – 2000
  - Internal timing behavior
  - Strong Security
    - Compositional
  - Scheduler-Independent
  - Agat's Approach
    - Combination of Typing and Code Transformation
  - Elimination of *protect*
- Will be discussed later...

Literature Review (Cont.)

- A New Type System for Secure Information Flow – Smith – 2001
  - Severe restrictions in the previous work
    - High variables were disallowed from the guards of while-loops.
  - Uniform scheduler
  - New type of the form $\tau_1 cmd \tau_2$
  - No assignment to an L variable may sequentially follow a command whose running time depends on H variables

Literature Review (Cont.)

- Probabilistic Noninterference through Weak Probabilistic Bisimulation – Smith – 2003
  - Weak probabilistic bisimulation for Markov chains
    - More relax with respect to timing
  - A type system
  - Supporting *fork* command that allows new threads to be spawned (Dynamic Thread Creation)

Literature Review (Cont.)

- Observational Determinism for Concurrent Program Security– Zdancewic and Myers – 2003
  - Immune to internal timing attacks and to attacks that exploit information about the thread scheduler
  - A type system and A race-freedom analysis
  - Two arbitrary initial configuration that the low observer cannot distinguish must produce indistinguishable traces

Literature Review (Cont.)

- Improved Typings for Probabilistic Noninterference in a Multi-Threaded Language- Smith – 2006
  - A more permissive type system
  - Internal leaks
  - Integrates the two previous work of Smith!

Literature Review (Cont.)

- Noninterference for Concurrent Programs and Thread Systems – Boudol and Castellani - 2006
  - A type system for controlled thread systems
  - Handling scheduling entirely within a possibilistic setting
  - Dynamic creation is not supported
  - if the termination of certain threads would be signaled to the scheduler, then controlled thread systems writing public variables cannot be typed.
    - Non-standard restriction
  - based on an operational semantics for the scheduler

Literature Review (Cont.)

- Securing Interaction between Threads and the Scheduler– Russo and Sabelfeld – 2006
  - Revisit the goal set (mentioned before)
  - A permissive NI-like security specification
  - A compositional type system
    - guarantees security for a wide class of schedulers
  - Language with primitives for raising and lowering the security level of threads
  - Termination-Insensitive security with respect to a class of deterministic schedulers
  - *hide* and *unhide* a thread
  - Scheduler with a non-standard interface must be used

Literature Review (Cont.)

- Security for Multithreaded Programs under Cooperative Scheduling – Russo and Sabelfeld – 2006
  - A transformation that eliminates the need for *protect* under cooperative scheduling
    - A wide class of schedulers
  - Both Termination-Sensitive and Termination-Insensitive security
    - yield command
  - Supporting dynamic thread creation

Literature Review (Cont.)

- Flexible Scheduler-Independent Security– Mantel and Sudbrock- 2010
  - Permits nondeterminism in a program's observable behavior without requiring a restrictive lock-step indistinguishability
  - No require non-standard modifications
  - Identification of a suitable class of schedulers that covers the most relevant schedulers
    - Robust schedulers e.g. uniform and RoundRobin
  - A type system

Literature Review (Cont.)

- Noninterfering Schedulers – Popescu, Holzl and Nipkow – 2013
  - A framework for expressing and analyzing the behavior of probabilistic schedulers
  - if a multi-threaded program is possibilistically noninterfering, then it is also probabilistically noninterfering when run under the given scheduler
  - covers dynamic thread creation
  - allows timing of thread execution to depend on high data
  - free from the termination assumption of the program

Literature Review (Cont.)

- Formal Verification of Language-Based Concurrent Noninterference – Popescu, Holzl and Nipkow – 2013
  - compositionality techniques for proving possibilistic NI for a while language with parallel composition
  - systemizing and comparing existing type-based NI notions
  - formalism and theorems have been verified in Isabelle/HOL

Literature Review (Cont.)

- Formalizing Probabilistic Noninterference – Popescu, Holzl and Nipkow – 2013
  - an Isabelle formalization of probabilistic NI for a multi-threaded language with uniform scheduling
  - resumption-based (bisimulation-based) and trace-based notions of NI
  - no support for dynamic thread creation

$$\text{syntactic criteria} \overset{\text{compositionality}}{\Longrightarrow} \text{resumption noninterference} \Longrightarrow \text{trace noninterference}$$

Literature Review (Cont.)

- Hybrid Monitors for Concurrent Noninterference – Askarov, Chong and Mantel – 2015
  - A monitoring framework to enforce strong security
  - hybrid = dynamic + static program analysis
  - each thread is guarded by its own local monitor, and a single global monitor
  - flow-sensitive progress-sensitive information flow control

Literature Review (Cont.)

- Specification and Static Enforcement of Scheduler-Independent Noninterference in a Middleweight Java– Iranmanesh and Fallah – 2016
  - introduce a new timing covert channel arises from the interplay between multithreading and object-orientation
  - multithreading + Middleweight Java
  - Scheduler-independent
  - A more permissive NI
  - A type system

Recall

- Probabilistic Noninterference for Multi-threaded Programs–Sabelfed and Sands – 2000
  - Internal timing behavior
  - Strong Security
    - Compositional
  - Scheduler-Independent
  - Agat's Approach
    - Combination of Typing and Code Transformation
  - Elimination of *protect*

Probabilistic NI for Multi-threaded Programs

- Language
  - a very simple shared-variable language
    - C, D, E,… range over commands Com
    - $\vec{C}$ denote a vector of commands $\langle C_0 \ldots C_{n-1} \rangle$
    - $\vec{C}, \vec{D}, \vec{E}$ range over $\vec{Com} = \cup_{n \in \mathbb{N}} Com^n$ the set of thread pools or programs

$$C ::= \text{skip} \mid Id := Exp \mid C_1; C_2$$
$$\mid \text{if } B \text{ then } C_1 \text{ else } C_2 \mid \text{while } B \text{ do } C \mid \text{fork}(C\vec{D})$$

Language (Cont.)

- small-step semantics
- configuration c is a pair of a command and a state (memory).
  - state $s \in St$ is a finite mapping from variables to values
- Low-equivalence

$s_1 =_L s_2$ iff the low components of s_1 and s_2 are the same. Let $Config$ denote $\vec{Com} \times \mathbf{St}$. The small-step semantics is given by transitions between configurations.

Small-step Semantics

$$\langle \text{skip}, s\rangle \rightarrow \langle\langle\rangle, s\rangle$$

$$\frac{\langle Exp, s\rangle \downarrow n}{\langle Id := Exp, s\rangle \rightarrow \langle\langle\rangle, [Id = n]s\rangle}$$

$$\frac{\langle C_1, s\rangle \rightarrow \langle\langle\rangle, s'\rangle}{\langle C_1; C_2, s\rangle \rightarrow \langle C_2, s'\rangle} \qquad \frac{\langle C_1, s\rangle \rightarrow \langle C_1'\vec{D}, s'\rangle}{\langle C_1; C_2, s\rangle \rightarrow \langle (C_1'; C_2)\vec{D}, s'\rangle}$$

$$\frac{\langle B, s\rangle \downarrow \text{True}}{\langle \text{if } B \text{ then } C_1 \text{ else } C_2, s\rangle \rightarrow \langle C_1, s\rangle}$$

$$\frac{\langle B, s\rangle \downarrow \text{False}}{\langle \text{if } B \text{ then } C_1 \text{ else } C_2, s\rangle \rightarrow \langle C_2, s\rangle}$$

$$\frac{\langle B, s\rangle \downarrow \text{True}}{\langle \text{while } B \text{ do } C, s\rangle \rightarrow \langle C; \text{while } B \text{ do } C, s\rangle}$$

$$\frac{\langle B, s\rangle \downarrow \text{False}}{\langle \text{while } B \text{ do } C, s\rangle \rightarrow \langle\langle\rangle, s\rangle}$$

$$\langle \text{fork}(C\vec{D}), s\rangle \rightarrow \langle C\vec{D}, s\rangle$$

Uniform Probabilistic Semantics of thread pools

$$\frac{\langle C_i, s\rangle \to \langle \vec{C}, s'\rangle}{\langle\langle C_0 \ldots C_{n-1}\rangle, s\rangle \to^{i}_{\frac{1}{n}} \langle\langle C_0 \ldots C_{i-1}\vec{C}C_{i+1} \ldots C_{n-1}\rangle, s'\rangle}$$

$$(\Sigma)\frac{p = \sum\{\!|q|\langle \vec{C}, s\rangle \to^{i}_{q} \langle \vec{D}, s'\rangle|\!\}}{\langle \vec{C}, s\rangle \to_p \langle \vec{D}, s'\rangle}$$

The probabilistic semantics satisfies the following "soundness" properties for all nonterminal programs $\vec{C}$

$$(i)\ \forall s.\ \sum\{\!|p|\langle \vec{C}, s\rangle \to_p \langle \vec{D}, s'\rangle|\!\} = 1,$$

$$(ii)\ \langle \vec{C}, s\rangle \to_p \langle \vec{D}, s'\rangle, \langle \vec{C}, s\rangle \to_q \langle \vec{D}, s'\rangle \Longrightarrow p = q.$$

Probabilistic NI

- A distribution μ of a set X (write $\mu \in \mathcal{D}(X)$) is a function $\mu \in X \rightarrow [0,1]$ s.t. $\sum_{x \in X} \mu x = 1$
- A probabilistic state μ is a distribution of configurations $\mu \in \mathcal{D}(\vec{C} \times \mathbf{St})$
- Modelled by a Markov chain of probabilistic states
- Every following probabilistic state μ' is computed by multiplying the row vector of probabilities from μ by the stochastic transition matrix T

Probabilistic NI (Cont.)

- The rows and columns of T correspond to possible configurations reachable from the initial one
- The element T(i,j) are the probability of a transition from configuration i to configuration j

$$\mu_1 \sim_l \mu_2 \text{ implies } \mu_1 T \sim_l \mu_2 T$$

Relational Definitions and Notation

- partial equivalence relation (PER) on A
  - a binary relation on A which is *symmetric* and *transitive*

$$dom(P) = \{x \in A \mid x\ P\ x\}$$

    - Domain and Range of a PER P are both equal to *dom(P)*

Probabilistic Bisimulation

- An *unlabeled probabilistic transition system* is a set of states S and a set of transitions of the form $s \to \mu$ where $s \in S$ and $\mu \in \mathcal{D}(S)$.

$$\langle \vec{C}, s \rangle \to \mu \text{ where } \mu \in \mathcal{D}(Config)$$

$$\mu(\langle \vec{D}, s' \rangle) = p \text{ iff } \langle \vec{C}, s \rangle \to_p \langle \vec{D}, s' \rangle$$

$$c \to_p S \iff p = \sum \{\!| q | c \to_q d, d \in S |\!\}$$

Probabilistic Bisimulation (Cont.)

- Definition1 – An equivalence relation R∈*Rel*(*Config*) is a probabilistic bisimulation if whenever c R d then

$$\forall S \in Config/R.\ c \rightarrow_p S \Longrightarrow d \rightarrow_p S.$$

- Definition2- A PER R∈*Rel*(*Config*) is a partial probabilistic bisimulation if whenever c R d then

$$(i)\ \forall S \in Config/R.\ c \rightarrow_p S \Longrightarrow d \rightarrow_p S,$$
$$(ii)\ c \rightarrow_0 Config \setminus dom(R).$$

- Proposition1- R is a partial probabilistic bisimulation implies R* a probabilistic bisimulation.

Probabilistic Bisimulation (Cont.)

- Definition3 - A PER $R \in Rel(\overrightarrow{Com})$ is a partial probabilistic bisimulation on thread pools iff $R \times Id_{st}$ is a partial probabilistic bisimulation on configurations.

- We write c~d (resp. C~D) iff there exists a partial probabilistic bisimulation on congurations (commands) relating c and d (C and D).

NI based on Partial Probabilistic Bisimulation

- Definition4- A PER R$\in$*Rel*($\overrightarrow{Com}$) is a partial probabilistic low-bisimulation iff R$\times$($=_L$) is a partial probabilistic bisimulation on configurations.

- We write C$\sim_L$D iff there exists a a partial probabilistic low-bisimulation relating programs C and D.

NI based on Partial Probabilistic Bisimulation (Cont.)

- Proposition2 - A PER R is a partial probabilistic low-bisimulation on commands iff whenever $\vec{C} R \vec{D}$ then

$$\forall s_1 =_L s_2 . \langle \vec{C}, s_1 \rangle \to \langle \vec{C}', s_1' \rangle \Longrightarrow$$
$$\exists \vec{D}', s_2' . \langle \vec{D}, s_2 \rangle \to \langle \vec{D}', s_2' \rangle$$
$$(i) \sum \{\!| p | \langle \vec{C}, s_1 \rangle \to_p \langle \vec{S}, s \rangle, \vec{S} \in [\vec{C}']_R, s =_L s_1' |\!\} =$$
$$\sum \{\!| p | \langle \vec{D}, s_2 \rangle \to_p \langle \vec{S}, s \rangle, \vec{S} \in [\vec{D}']_R, s =_L s_2' |\!\}$$
$$(ii)\ \vec{C}'\ R\ \vec{D}', s_1' =_L s_2',$$

- The security specification for a thread pool:

$$\vec{C} \text{ is } \textit{secure} \iff \vec{C} \sim_L \vec{C}$$

Semantics with Schedulers

- Inductive definition: the set of possible histories *Hist*.
- A *history* H is a sequence of pairs $H \in (\mathbb{N} \times \mathbb{N})^*$
  - In each pair of the sequence, the first component is the index of the thread last chosen for computation, and the second component is the total number of threads that remained in the configuration after that thread's computation step

$$\epsilon \in Hist \qquad (0, n) \in Hist$$

$$\frac{H(i,m) \in Hist}{H(i,m)(j,n) \in Hist}(j \leq m - 1 \leq n)$$

Define the number of live processes for a given history by

$$live(\epsilon) = 1, \qquad live(H(i,n)) = n.$$

Semantics with Schedulers(Cont.)

- A scheduler σ is any function that given a history H and the low part *l* of a state returns a probability distribution on live processes.

$$\sigma(H, l) \in \mathcal{D}(\{0 \ldots live(H) - 1\})$$

- Given a scheduler σ not all histories are feasible under σ. Define the set of σ-feasible histories $Hist_\sigma \subseteq Hist$ by

$$\epsilon \in Hist_\sigma$$

$$\frac{H \in Hist_\sigma \quad H(i,m) \in Hist \quad \exists l.\, \sigma(H,l)i > 0}{H(i,m) \in Hist_\sigma}$$

Semantics with Schedulers(Cont.)

$$\frac{\langle C_i, s\rangle \twoheadrightarrow \langle \vec{C}, s'\rangle}{\langle H, \langle C_0 \ldots C_i \ldots C_{n-1}\rangle, s\rangle \rightarrow_p \langle H', \langle C_0 \ldots \vec{C} \ldots C_{n-1}\rangle, s'\rangle}$$

where $H' = H(i, n + |\vec{C}| - 1), p = \sigma(H, l)\, i$ and $s = (h, l)$ for some h.

Probabilistic NI with Schedulers

- Let us capture the degree of how the history can vary by defining a relation $=_\sigma$ on σ-feasible histories. It relates σ-feasible histories which are indistinguishable for σ.
- Define $=_\sigma \in \mathrm{Rel}(\mathrm{Hist}_\sigma)$ to be the largest relation satisfying

$$\frac{H =_\sigma H'}{\forall l.\, \sigma(H,l) = \sigma(H',l)} \qquad \frac{H =_\sigma H'}{H(i,m) =_\sigma H'(i,m)}$$

Probabilistic NI with Schedulers (Cont.)

- Definition5- Given a scheduler σ, a PER $R \in Rel(\overrightarrow{Com})$ is a σ-specific partial probabilistic low-bisimulation iff $(=_\sigma) \times R \times (=_L)$ is a σ-specific partial probabilistic bisimulation on configurations.
- Write $\vec{C} \sim_L^\sigma \vec{D}$ iff there exists a partial probabilistic low-bisimulation relating programs $\vec{C}$ and $\vec{D}$.

$$\vec{C} \text{ is } \sigma\text{-}secure \iff \vec{C} \sim_L^\sigma \vec{C}$$

Scheduler-independent NI

- The scheduler-independent low-bisimulation $\approx_L$ and the scheduler-independent security (SI-security).

$$\vec{C} \approx_L \vec{D} \iff \forall \sigma. \vec{C} \sim_L^{\sigma} \vec{D}, \text{ and}$$

$$\vec{C} \text{ is } \textit{SI-secure} \iff \vec{C} \approx_L \vec{C}.$$

- $\approx_L \subseteq \sim_L \Rightarrow \ \vec{C}\ is\ SI-secure \rightarrow \vec{C}\ is\ secure.$
- To achieve a compositional bisimulation
  - restrict the strong low-bisimulation so that any two strongly low-bisimilar thread pools must be of equal size and must create/kill exactly the same number of processes at each step under any schedule.

Scheduler-independent NI (Cont.)

- Definition6- Define the strong low-bisimulation $\approxeq_L$ to be the union of all symmetric relations R on thread pools of equal size s.t. whenever $\langle C_0 \ldots C_{n-1} \rangle \ R \ \langle D_0 \ldots D_{n-1} \rangle$ then

$$\forall s_1 =_L s_2 \forall i. \langle C_i, s_1 \rangle \rightarrow \langle \vec{C'}, s_1' \rangle \Longrightarrow$$
$$\exists \vec{D'}, s_2'. \langle D_i, s_2 \rangle \rightarrow \langle \vec{D'}, s_2' \rangle, \vec{C'} \ R \ \vec{D'}, s_1' =_L s_2'$$

$$\vec{C} \text{ is } \textit{strongly secure} \iff \vec{C} \approxeq_L \vec{C}.$$

Proposition 3 $\vec{C}$ *is strongly secure* $\Longrightarrow$ $\vec{C}$ *is SI-secure.*

Conclusion

- Definition of Probabilistic NI
- A survey on related works
  - Pros and Cons
- Probabilistic Noninterference for Multi-threaded Programs
  - Strong Security
- Future work:
  - full-featured languages
  - dynamic mechanisms
  - widening attacker observations
  - more permissive NI notion

References

[1] Zdancewic, Steve, and Andrew C. Myers. "Observational determinism for concurrent program security." In *Computer Security Foundations Workshop, 2003. Proceedings. 16th IEEE*, pp. 29-43. IEEE, 2003.

[2] Volpano, Dennis, and Geoffrey Smith. "Probabilistic noninterference in a concurrent language1." *Journal of Computer Security* 7, no. 2-3 (1999): 231-253.

[3] Sabelfeld, Andrei, and David Sands. "Probabilistic noninterference for multi-threaded programs." In *Computer Security Foundations Workshop, 2000. CSFW-13. Proceedings. 13th IEEE*, pp. 200-214. IEEE, 2000.

[4] Popescu, Andrei, Johannes Hölzl, and Tobias Nipkow. "Formalizing probabilistic noninterference." In *International Conference on Certified Programs and Proofs*, pp. 259-275. Springer International Publishing, 2013.

[5] Smith, Geoffrey. "Improved typings for probabilistic noninterference in a multi-threaded language." *Journal of Computer Security* 14, no. 6 (2006): 591-623.

[6] Mantel, Heiko, and Henning Sudbrock. "Flexible scheduler-independent security." In *European Symposium on Research in Computer Security*, pp. 116-133. Springer Berlin Heidelberg, 2010.

[7] Popescu, Andrei, Johannes Hölzl, and Tobias Nipkow. "Formal verification of language-based concurrent noninterference." *Journal of Formalized Reasoning* 6, no. 1 (2013): 1-30.

[8] Askarov, Aslan, Stephen Chong, and Heiko Mantel. "Hybrid monitors for concurrent noninterference." In *2015 IEEE 28th Computer Security Foundations Symposium*, pp. 137-151. IEEE, 2015.

[9] Boudol, Gérard, and Ilaria Castellani. "Noninterference for concurrent programs and thread systems." *Theoretical Computer Science* 281, no. 1 (2002): 109-130.

References

[10] Russo, Alejandro, and Andrei Sabelfeld. "Securing interaction between threads and the scheduler." In *19th IEEE Computer Security Foundations Workshop (CSFW'06)*, pp. 13-pp. IEEE, 2006.

[11] Smith, Geoffrey. "A New Type System for Secure Information Flow." In *csfw*, vol. 1, p. 4. 2001.

[12] Popescu, Andrei, Johannes Hölzl, and Tobias Nipkow. "Noninterfering schedulers." In *International Conference on Algebra and Coalgebra in Computer Science*, pp. 236-252. Springer Berlin Heidelberg, 2013.

[13] Smith, Geoffrey. "Probabilistic noninterference through weak probabilistic bisimulation." In *Computer Security Foundations Workshop, 2003. Proceedings. 16th IEEE*, pp. 3-13. IEEE, 2003.

[14] Popescu, Andrei, Johannes Hölzl, and Tobias Nipkow. "Proving concurrent noninterference." In *International Conference on Certified Programs and Proofs*, pp. 109-125. Springer Berlin Heidelberg, 2012.

[15] Russo, Alejandro, and Andrei Sabelfeld. "Security for multithreaded programs under cooperative scheduling." In *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*, pp. 474-480. Springer Berlin Heidelberg, 2006.

[16] Agat, Johan. "Transforming out timing leaks." In *Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 40-53. ACM, 2000.

[17] Iranmanesh, Zeinab, and Mehran S. Fallah. "Specification and static enforcement of scheduler-independent noninterference in a middleweight Java." *Computer Languages, Systems & Structures* 46 (2016): 20-43.

[18] Focardi, Riccardo, and Roberto Gorrieri. "A Classification of Security Properties for Process Algebras1." *Journal of Computer security* 3, no. 1 (1995): 5-33.

Any Questions?