Tradeoff between Performance and Security

Alessandro Aldini

University of Urbino "Carlo Bo" – Italy

Foundations of Security Analysis and Design (FOSAD)
3 September 2011, Bertinoro

Outline

Where is the focus...

- Modern software systems: security (dependability) and performance requirements
- Trading security and performance: what does it mean?
- Different perspectives:
  - Performability-like principle: do the costs of security cause a tolerable degradation of performance?
  - Noninterference-like principle: do the performance optimizations cause security leaks?
  - Trading the two aspects: is it possible to balance the costs of the security mechanisms with the performance profile of the system?

Where is the focus...

- Modern software systems: security (dependability) and performance requirements
- Trading security and performance: what does it mean?
- Different perspectives:
  - Performability-like principle: do the costs of security cause a tolerable degradation of performance?
  - Noninterference-like principle: do the performance optimizations cause security leaks?
  - Trading the two aspects: is it possible to balance the costs of the security mechanisms with the performance profile of the system?

Where is the focus...

- Modern software systems: security (dependability) and performance requirements
- Trading security and performance: what does it mean?
- Different perspectives:
  - Performability-like principle: do the costs of security cause a tolerable degradation of performance?
  - Noninterference-like principle: do the performance optimizations cause security leaks?
  - Trading the two aspects: is it possible to balance the costs of the security mechanisms with the performance profile of the system?

Performability-like principle

Determine security metrics, estimate security costs, and evaluate the relation with performance measures

security metrics

Some analogy with dependability metrics: time between security incidents, time to security incident detection/recovery, time between detection and recovery, reward of leaked information (empirical data are important)

references

Littlewood 1993, 2004
Trivedi 2004
Verendel 2009

Performability-like principle

Determine security metrics, estimate security costs, and evaluate the relation with performance measures

security costs

Example: encryption

- encryption time (symmetric vs. asymmetric), key length, key generation
- impact on throughput and response time
- empirical analysis of both costs and impact

references

Lamprecht 2006

Performability-like principle

Determine security metrics, estimate security costs, and evaluate the relation with performance measures

security costs

Example: cryptographic protocols

- key distribution mechanisms, additional message exchange
- impact on throughput, response time, scalability, utilization
- many (semi) formal models, no formulation of the impact

references

Zhao 2009 (Kerberos)

Performability-like principle

Determine security metrics, estimate security costs, and evaluate the relation with performance measures

security costs

Example: access control

- authentication mechanisms, intrusion detection systems
- impact on response time, utilization, availability
- many (semi) formal models, no analysis of tradeoff

references

Madan 2004 (IDS)
Wang 2010 (email system)

Performability-like principle

Determine security metrics, estimate security costs, and evaluate the relation with performance measures

security costs

Example: lightweight security

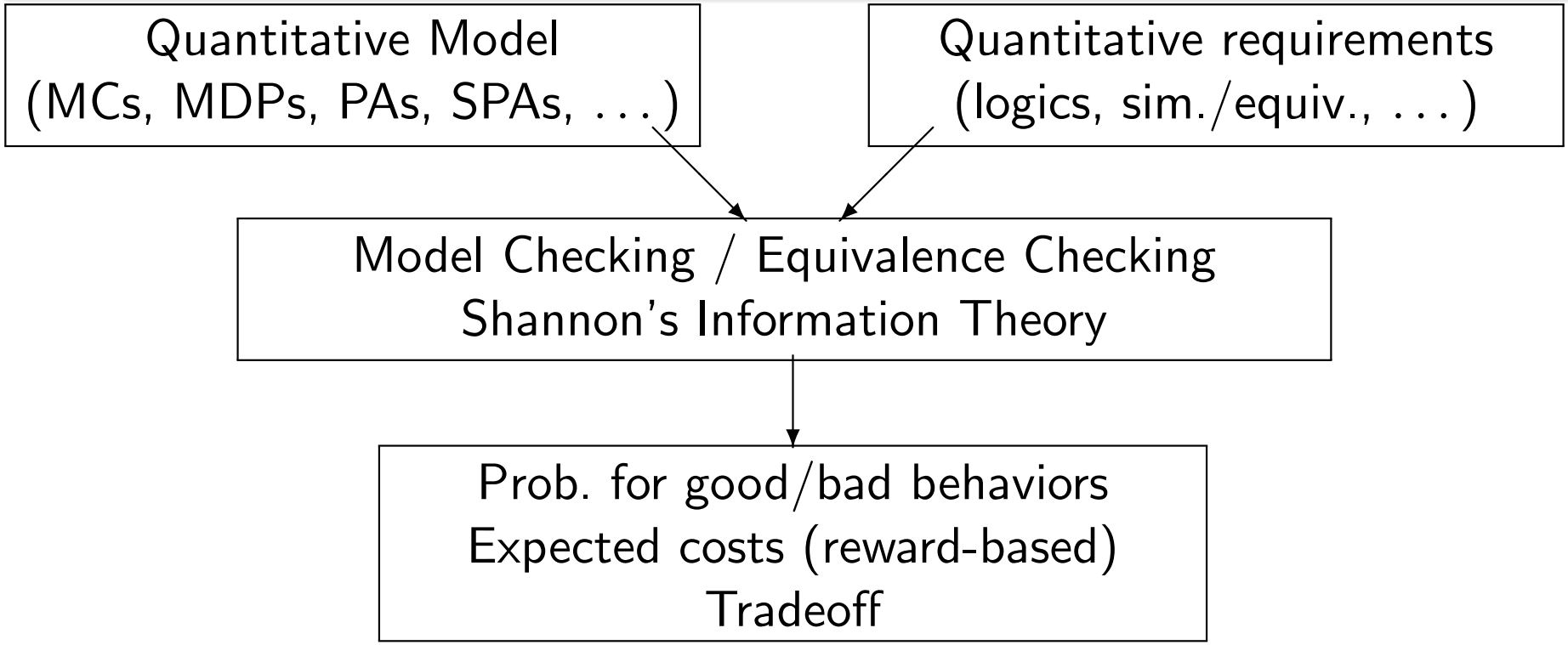
- trust/security infrastructures in WLANs and MANETs
- impact on response time, utilization
- many (semi) formal models, difficult analysis of tradeoff

references

Cho 2008 (MANETs)

Noninterference-like principle

Employ quantitative information to estimate security leaks



Examples

PIN cracking schemes, contract signing, fair exchange, network virus infection, anonymity, DoS, non-repudiation, crypto-protocols, . . .

Noninterference-like principle

Employ quantitative information to estimate security leaks

Pros and Cons

- quantitative information typically considered: (conditional) probability distributions of events, discrete time
- security metric typically considered: amount of information leakage
- tradeoff: sometimes it is clear the cost to pay for a reduction of the information leakage

references

Baier et al.
Di Pierro et al.
Malacaria
Segala et al.

General Requirements

What we need

- Need for performance/security models that can be mutually validated
- Need for specification of performance/security measures
- Need for trading guidelines/mechanisms

General Requirements

What we need

- Need for performance/security models that can be mutually validated
- Need for specification of performance/security measures
- Need for trading guidelines/mechanisms

General Requirements

What we need

- Need for performance/security models that can be mutually validated
- Need for specification of performance/security measures
- Need for trading guidelines/mechanisms

A General Methodology

Problem

- Analyzing both *quantitative* aspects (e.g. performance) and *dependability* aspects (such as security, reliability, safety, and availability)
- in a component-oriented fashion.

Goal

- Guiding the system design towards a balanced trade-off among all these aspects.

Solution

- Integrated view
- Equivalence-based integrated analysis

A General Methodology

Problem

- Analyzing both *quantitative* aspects (e.g. performance) and *dependability* aspects (such as security, reliability, safety, and availability)
- in a component-oriented fashion.

Goal

- Guiding the system design towards a balanced trade-off among all these aspects.

Solution

- Integrated view
- Equivalence-based integrated analysis

A General Methodology

Problem

- Analyzing both *quantitative* aspects (e.g. performance) and *dependability* aspects (such as security, reliability, safety, and availability)
- in a component-oriented fashion.

Goal

- Guiding the system design towards a balanced trade-off among all these aspects.

Solution

- Integrated view
- Equivalence-based integrated analysis

A General Methodology

Problem

- Analyzing both *quantitative* aspects (e.g. performance) and *dependability* aspects (such as security, reliability, safety, and availability)
- in a component-oriented fashion.

Goal

- Guiding the system design towards a balanced trade-off among all these aspects.

Solution

- Integrated view
- Equivalence-based integrated analysis

A General Methodology

Problem

- Analyzing both *quantitative* aspects (e.g. performance) and *dependability* aspects (such as security, reliability, safety, and availability)
- in a component-oriented fashion.

Goal

- Guiding the system design towards a balanced trade-off among all these aspects.

Solution

- Integrated view
- Equivalence-based integrated analysis

A General Methodology

Scenario

- A single component may cope with a sole specific aspect in a one-to-one fashion, or else crosscutting aspects may be handled by several components.
- In any case, the components may interfere each other when pursuing the goal of satisfying the requirements of different aspects.

Examples

- Mechanisms for controlling power-consumption / resource access / resource usage may interfere with security aspects.
- Viceversa, mechanisms dedicated to security aspects may interfere with QoS parameters.

A General Methodology

Scenario

- A single component may cope with a sole specific aspect in a one-to-one fashion, or else crosscutting aspects may be handled by several components.
- In any case, the components may interfere each other when pursuing the goal of satisfying the requirements of different aspects.

Examples

- Mechanisms for controlling power-consumption / resource access / resource usage may interfere with security aspects.
- Viceversa, mechanisms dedicated to security aspects may interfere with QoS parameters.

A General Methodology

Scenario

- A single component may cope with a sole specific aspect in a one-to-one fashion, or else crosscutting aspects may be handled by several components.
- In any case, the components may interfere each other when pursuing the goal of satisfying the requirements of different aspects.

Examples

- Mechanisms for controlling power-consumption / resource access / resource usage may interfere with security aspects.
- Viceversa, mechanisms dedicated to security aspects may interfere with QoS parameters.

A General Methodology

Goal

- Evaluating the capability of a component of interfering with the behaviors (of other components) aiming at satisfying the requirements of specific aspects.
- The ultimate goal is to reach a balanced trade-off among all the functional and nonfunctional aspects.

Approach

- Performing a noninterference check in order to assess the impact of every component on the security requirements.
- Applying quantitative analysis techniques in order to estimate the revealed interference and the impact of the mitigating strategies on the performability aspects.

A General Methodology

Goal

- Evaluating the capability of a component of interfering with the behaviors (of other components) aiming at satisfying the requirements of specific aspects.
- The ultimate goal is to reach a balanced trade-off among all the functional and nonfunctional aspects.

Approach

- Performing a noninterference check in order to assess the impact of every component on the security requirements.
- Applying quantitative analysis techniques in order to estimate the revealed interference and the impact of the mitigating strategies on the performability aspects.

A General Methodology

Goal

- Evaluating the capability of a component of interfering with the behaviors (of other components) aiming at satisfying the requirements of specific aspects.
- The ultimate goal is to reach a balanced trade-off among all the functional and nonfunctional aspects.

Approach

- Performing a noninterference check in order to assess the impact of every component on the security requirements.
- Applying quantitative analysis techniques in order to estimate the revealed interference and the impact of the mitigating strategies on the performability aspects.

A General Methodology

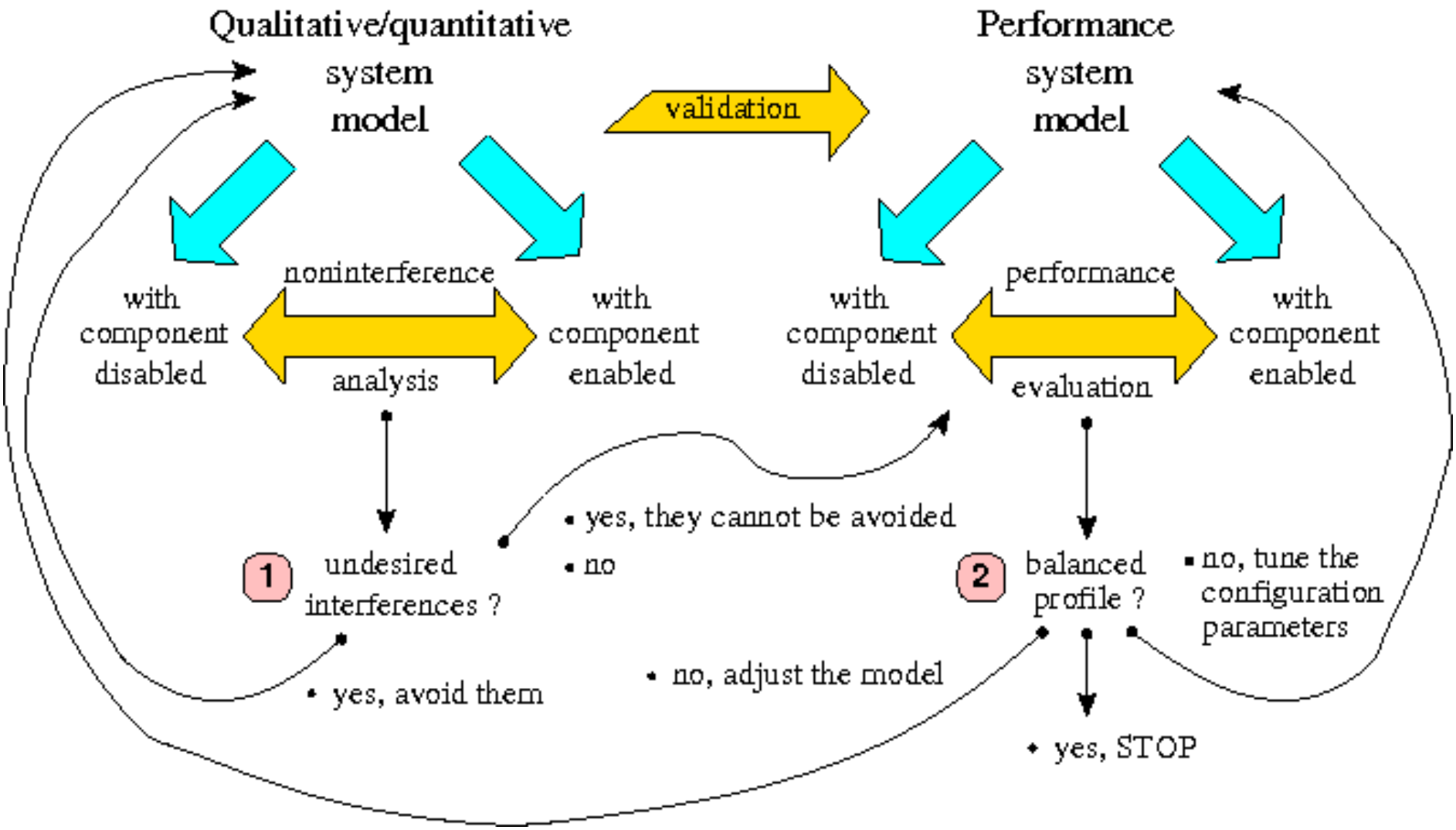
Goal

- Evaluating the capability of a component of interfering with the behaviors (of other components) aiming at satisfying the requirements of specific aspects.
- The ultimate goal is to reach a balanced trade-off among all the functional and nonfunctional aspects.

Approach

- Performing a noninterference check in order to assess the impact of every component on the security requirements.
- Applying quantitative analysis techniques in order to estimate the revealed interference and the impact of the mitigating strategies on the performability aspects.

A General Methodology



A General Methodology

Steps

- Provide a (architectural) description of the system
- Choose the security property of interest and single out the components dedicated to this aspect
- Determine the components of which the interference has to be evaluated
- Perform the noninterference check
- Decide whether the revealed interference (or the related mitigating strategy) is tolerable or not

A General Methodology

Steps

- Provide a (architectural) description of the system
- Choose the security property of interest and single out the components dedicated to this aspect
- Determine the components of which the interference has to be evaluated
- Perform the noninterference check
- Decide whether the revealed interference (or the related mitigating strategy) is tolerable or not

A General Methodology

Steps

- Provide a (architectural) description of the system
- Choose the security property of interest and single out the components dedicated to this aspect
- Determine the components of which the interference has to be evaluated
- Perform the noninterference check
- Decide whether the revealed interference (or the related mitigating strategy) is tolerable or not

A General Methodology

Steps

- Provide a (architectural) description of the system
- Choose the security property of interest and single out the components dedicated to this aspect
- Determine the components of which the interference has to be evaluated
- Perform the noninterference check
- Decide whether the revealed interference (or the related mitigating strategy) is tolerable or not

A General Methodology

Steps

- Provide a (architectural) description of the system
- Choose the security property of interest and single out the components dedicated to this aspect
- Determine the components of which the interference has to be evaluated
- Perform the noninterference check
- Decide whether the revealed interference (or the related mitigating strategy) is tolerable or not

A General Methodology

Examples

- Determine the influence of faults triggered by a component upon the behavior of system components performing security-critical applications.
- Determine the influence of events triggered by non-trusted components upon the behavior of system components performing security-critical applications.
- Determine the impact of mechanisms for controlling power consumption / performance optimizations on security aspects.

A General Methodology

Examples

- Determine the influence of faults triggered by a component upon the behavior of system components performing security-critical applications.
- Determine the influence of events triggered by non-trusted components upon the behavior of system components performing security-critical applications.
- Determine the impact of mechanisms for controlling power consumption / performance optimizations on security aspects.

A General Methodology

Examples

- Determine the influence of faults triggered by a component upon the behavior of system components performing security-critical applications.
- Determine the influence of events triggered by non-trusted components upon the behavior of system components performing security-critical applications.
- Determine the impact of mechanisms for controlling power consumption / performance optimizations on security aspects.

Noninterference Theory

Original Idea

- A group of high-security level users, employing confidential operations only, is not interfering with a group of low-security level users, observing public operations only, if what the first group of users can do with the confidential operations has no effect on what the second group of users can see.
- In the security setting, noninterference analysis can reveal direct and indirect information flows, called *covert channels*, that violate the access policies based on the different access clearances assigned to different user groups.

Noninterference Theory

Original Idea

- A group of high-security level users, employing confidential operations only, is not interfering with a group of low-security level users, observing public operations only, if what the first group of users can do with the confidential operations has no effect on what the second group of users can see.
- In the security setting, noninterference analysis can reveal direct and indirect information flows, called *covert channels*, that violate the access policies based on the different access clearances assigned to different user groups.

Noninterference Theory

Implementation

- Action names (events) are divided into two disjoint sets:
  - *High*, representing system activities at high-security level
  - *Low*, representing system activities at low-security level
- A system model Q has no covert channels if the system view where all the high-level activities are **hidden** to low-level observers, is indistinguishable with respect to the system view where these activities are **prevented** from execution:

$$Q/High \approx_B Q\backslash High$$

Noninterference Theory

General Idea

- A system execution can be viewed as an information flow.
- A group of system components (high components), described by a certain set of behaviors, is not interfering with another group of system components (low components) if the behaviors of the first group of components have no effect on what the second group of components can see.
- In this more general setting, noninterference analysis can reveal covert channels indicating the existence of undesired information flows among component behaviors that are responsible for compromising several different (security) aspects.

Noninterference Theory

General Idea

- A system execution can be viewed as an information flow.
- A group of system components (high components), described by a certain set of behaviors, is not interfering with another group of system components (low components) if the behaviors of the first group of components have no effect on what the second group of components can see.
- In this more general setting, noninterference analysis can reveal covert channels indicating the existence of undesired information flows among component behaviors that are responsible for compromising several different (security) aspects.

Noninterference Theory

General Idea

- A system execution can be viewed as an information flow.
- A group of system components (high components), described by a certain set of behaviors, is not interfering with another group of system components (low components) if the behaviors of the first group of components have no effect on what the second group of components can see.
- In this more general setting, noninterference analysis can reveal covert channels indicating the existence of undesired information flows among component behaviors that are responsible for compromising several different (security) aspects.

A Process-algebraic Approach to Model Design

The Approach in a Nutshell

- The architectural description of a component-based system comprises the description of the **individual system component types** and the description of the **overall system topology**.

A Process-algebraic Approach to Model Design

The Approach in a Nutshell

- The architectural description of a component-based system comprises the description of the **individual system component types** and the description of the **overall system topology**.
- The description of a single component type (Architectural Element Type, AET) includes its name, its parameters, its **behavior**, and its **interactions** with other component types.
- The component behavior expresses all the alternative sequences of activities that the AET can carry out – which is formalized by means of process algebra – while the component interactions are those activities used by the component type to communicate with the rest of the system.

A Process-algebraic Approach to Model Design

The Approach in a Nutshell

- The architectural description of a component-based system comprises the description of the **individual system component types** and the description of the **overall system topology**.
- The description of the system topology includes the **instances** of the component types that form the system (Architectural Element Instance, AEI), together with the specification of the way in which their interactions are attached to each other in order to make the components communicate.

A Process-algebraic Approach to Model Design

The Semantics in a Nutshell

- Given an AET $\mathcal{C}$ and an AEI C of type $\mathcal{C}$, from the semantics of C, $[\![C]\!]$, which is defined to be the process algebraic behavior associated with C, it is possible to extract a stochastic model in the form of an action-labeled Continuous Time Markov Chain (CTMC).
- The semantics of an architectural description $\mathcal{A}$ is derived by **composing in parallel** the semantics of its AEIs according to the declared attachments:

$$[\![C_1, \ldots, C_n]\!]_{\mathcal{A}}$$

A Process-algebraic Approach to Model Design

The Semantics in a Nutshell

- Given an AET $\mathcal{C}$ and an AEI C of type $\mathcal{C}$, from the semantics of C, $[\![C]\!]$, which is defined to be the process algebraic behavior associated with C, it is possible to extract a stochastic model in the form of an action-labeled Continuous Time Markov Chain (CTMC).
- The semantics of an architectural description $\mathcal{A}$ is derived by **composing in parallel** the semantics of its AEIs according to the declared attachments:

$$[\![C_1, \ldots, C_n]\!]_{\mathcal{A}}$$

First Step: Noninterference Analysis

- Let $\mathcal{A}$ be an architectural description with AEIs $K, C_1, \ldots, C_n, B_1, \ldots, B_m$.
- Suppose to be interested in evaluating the impact of K on the behavior of $C_1, \ldots, C_n$ that is related to a specific security aspect.
- The set *Name* of action names is divided into two disjoint sets:
  - $High \subseteq Name$, representing the system activities performed by K of which we intend to evaluate the impact.
  - $Low \subseteq Name$, representing the system activities performed by $C_1, \ldots, C_n$ and related to the behavior we intend to monitor.
- All the remaining activities carried out by the system are simply disregarded and can be hidden.
- At the architectural level, the different system views can be defined as behavioral modifications by employing static operators for hiding and restriction.

First Step: Noninterference Analysis

- Let $\mathcal{A}$ be an architectural description with AEIs $K, C_1, \ldots, C_n, B_1, \ldots, B_m$.
- Suppose to be interested in evaluating the impact of K on the behavior of $C_1, \ldots, C_n$ that is related to a specific security aspect.
- The set *Name* of action names is divided into two disjoint sets:
  - $High \subseteq Name$, representing the system activities performed by K of which we intend to evaluate the impact.
  - $Low \subseteq Name$, representing the system activities performed by $C_1, \ldots, C_n$ and related to the behavior we intend to monitor.
- All the remaining activities carried out by the system are simply disregarded and can be hidden.
- At the architectural level, the different system views can be defined as behavioral modifications by employing static operators for hiding and restriction.

First Step: Noninterference Analysis

- Let $\mathcal{A}$ be an architectural description with AEIs $K, C_1, \ldots, C_n, B_1, \ldots, B_m$.
- Suppose to be interested in evaluating the impact of K on the behavior of $C_1, \ldots, C_n$ that is related to a specific security aspect.
- The set *Name* of action names is divided into two disjoint sets:
  - $High \subseteq Name$, representing the system activities performed by K of which we intend to evaluate the impact.
  - $Low \subseteq Name$, representing the system activities performed by $C_1, \ldots, C_n$ and related to the behavior we intend to monitor.
- All the remaining activities carried out by the system are simply disregarded and can be hidden.
- At the architectural level, the different system views can be defined as behavioral modifications by employing static operators for hiding and restriction.

First Step: Noninterference Analysis

- Let $\mathcal{A}$ be an architectural description with AEIs $K, C_1, \ldots, C_n, B_1, \ldots, B_m$.
- Suppose to be interested in evaluating the impact of K on the behavior of $C_1, \ldots, C_n$ that is related to a specific security aspect.
- The set *Name* of action names is divided into two disjoint sets:
  - $High \subseteq Name$, representing the system activities performed by K of which we intend to evaluate the impact.
  - $Low \subseteq Name$, representing the system activities performed by $C_1, \ldots, C_n$ and related to the behavior we intend to monitor.
- All the remaining activities carried out by the system are simply disregarded and can be hidden.
- At the architectural level, the different system views can be defined as behavioral modifications by employing static operators for hiding and restriction.

First Step: Noninterference Analysis

- Let $\mathcal{A}$ be an architectural description with AEIs $K, C_1, \ldots, C_n, B_1, \ldots, B_m$.
- Suppose to be interested in evaluating the impact of K on the behavior of $C_1, \ldots, C_n$ that is related to a specific security aspect.
- The set *Name* of action names is divided into two disjoint sets:
  - $High \subseteq Name$, representing the system activities performed by K of which we intend to evaluate the impact.
  - $Low \subseteq Name$, representing the system activities performed by $C_1, \ldots, C_n$ and related to the behavior we intend to monitor.
- All the remaining activities carried out by the system are simply disregarded and can be hidden.
- At the architectural level, the different system views can be defined as behavioral modifications by employing static operators for hiding and restriction.

First Step: Noninterference Analysis

- K does not interfere with $C_1, \ldots, C_n$ if the *low* behavior of $C_1, \ldots, C_n$ with the *high* activities of K being made unobservable is equivalent to the *low* behavior of $C_1, \ldots, C_n$ with the same activities being prevented from taking place:

$$\boxed{\begin{array}{c} [\![C_1, \ldots, C_n, K, B_1, \ldots, B_m]\!]_{\mathcal{A}} / (Name - Low) \\ \approx \\ [\![C_1, \ldots, C_n, K, B_1, \ldots, B_m]\!]_{\mathcal{A}} \backslash High / (Name - Low) \end{array}}$$

  where $\approx$ is the chosen notion of behavioral equivalence, which can be nondeterministic, probabilistic, timed, or a combination of these.

- The more information is added to the system model, the higher the number of potential vulnerabilities revealed through fine-grain notions of noninterference.

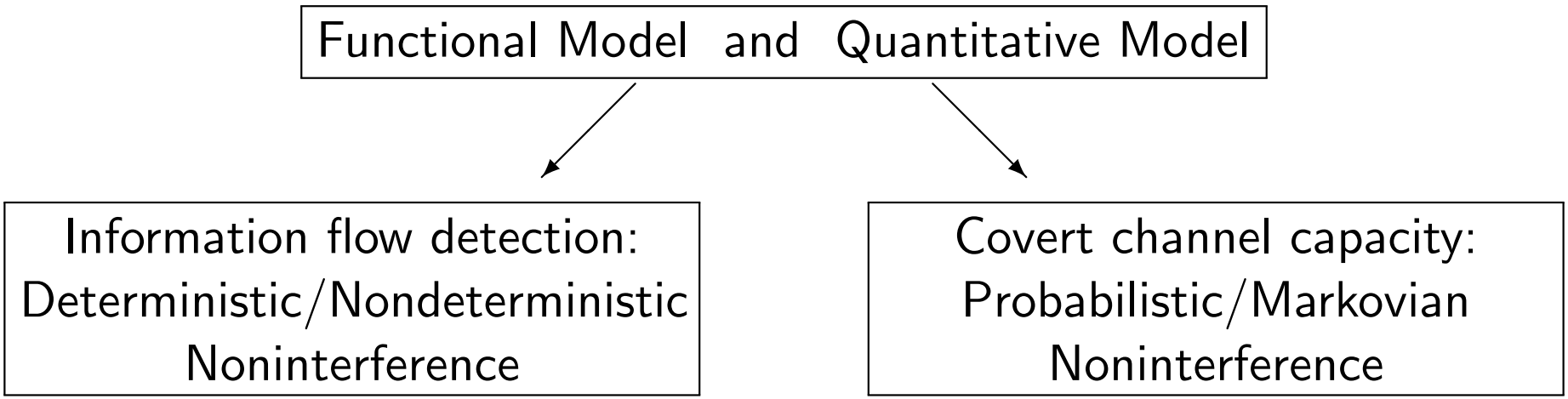
First Step: Noninterference Analysis

- The interference of K on the behavior of $C_1, \ldots, C_n$ can be inferred compositionally and in a more efficient way whenever:
  1. K is the central AEI of a star topology that includes $C_1, \ldots, C_n$.
  2. K is an AEI of a cycle that includes $C_1, \ldots, C_n$.

First Step: Result of the Analysis

- **No information flow is revealed**: we have the guarantee that K is transparent with respect to the monitored activities of $C_1, \ldots, C_n$ and with respect to the chosen noninterference check.
- **An undesired, direct or indirect, information flow is revealed**: diagnostic information provided by the noninterference check (typically in the form of a logic formula) can be employed to determine the causes of the interference.
- If the interference can be eliminated with a minor impact on the functionalities of the system behavior, the diagnostic information can be employed to suitably modify the model.
- In contrast, for all information flows that are either unavoidable or tolerated as they would require a significant revision of the model, it is necessary to estimate the interference.
- In any case, a performance-based validation is needed.

Trading the Noninterference Check with the Model: An Example



- exact analysis
- 0/1 result

- approximate analysis
  Probabilistic/Markovian noninterference (Gray, Dipierro et al., Aldini et al., Smith, Desharnais et al., ...)
- $[0; 1]$ result

$\uparrow$

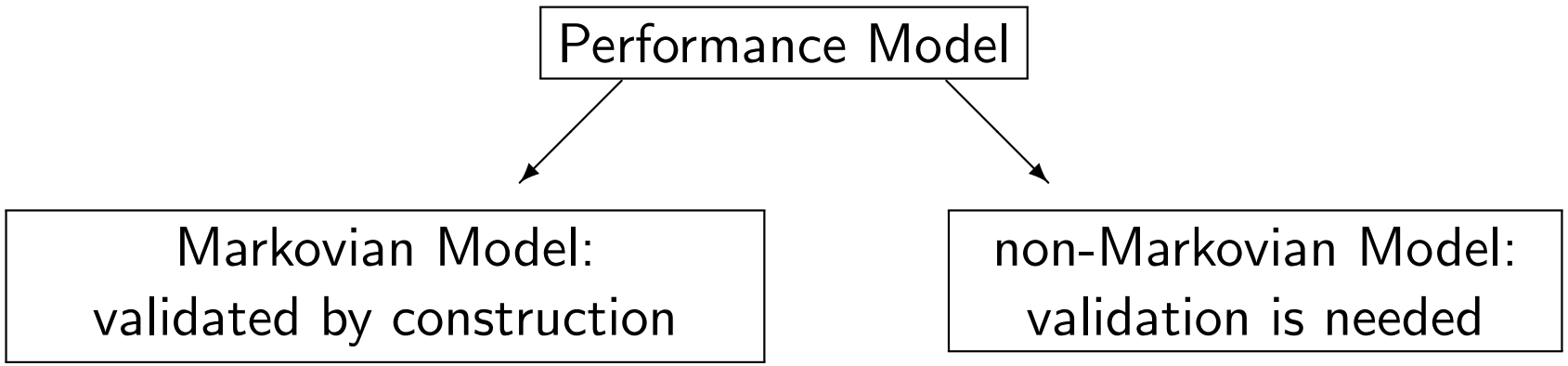
best-case/average/worst-case

Trading the Noninterference Check with the Model: An Example

Nonfunctional Noninterference

- **Model**: generative/reactive probabilistic systems, CTMC, DTMC, MDP, PA
- **Equivalence**: weak probabilistic bisimulation, Markovian bisimulation, Markovian testing bisimulation
- **Noninterference Check**: if passed, see the nondeterministic case, otherwise an interference is revealed. *Can we measure the covert channel capacity?*
- **Approximation**: the notion of equivalence is relaxed to tolerate negligible fluctuations in the nonfunctional behaviors. The measure that is estimated by the approximation represents the covert channel capacity.

Second Step: Performance Analysis



- Markovian Noninterference
- How to measure the covert channel capacity?
- Approximations
- Performance comparison through reward-based numerical analysis or simulation

Second Step: Performance Analysis

Performance Comparison

- Define the performance metrics that are directly related to the bandwidth of the revealed covert channel (follow the guidelines provided by the diagnostic information).
- Alternatively, in the case no information flow has been revealed, define the QoS-related metrics of interest.
- In any case, the resulting performance figures reveal whether a balanced tradeoff between security - in terms of bandwidth of each covert channel - and QoS - in terms of performance measures like system throughput and response time - is met or not.

Second Step: Performance Analysis

Performance Comparison

- Depending on the obtained results, the performance figures are then used as a feedback to decide whether the system is to be tuned by changing the configuration parameters that affect the metrics of interest, or else it is necessary to adjust the architectural model and restart the integrated analysis.
- The choice is mainly guided by the design requirements, e.g. strict vs. relaxed security needs and loose vs. tight QoS.

The Methodology in Practice

Examples

- Determine the bandwidth of covert channels revealing security violations in order to estimate, e.g., the amount of sensitive information leaked per unit of time, or the overhead due to strategies minimizing these covert channels, or else the tradeoff between security requirements and quality of service parameters.
- Estimate the impact of mechanisms for minimizing the security risks on the service availability by measuring the service response time.

Some Open Problems

- Choice of the most adequate notion of equivalence (bisimulation may distinguish too much).
- Approximation of Markovian equivalences.
- Automatization of the passage from first to second step.

References

- A. Aldini, M. Bernardo, F. Corradini. A Process Algebraic Approach to Software Architecture Design. Springer, 2010.
- A. Aldini, M. Bernardo. A Formal Approach to the Integrated Analysis of Security and QoS. Journal of Reliability Engineering and System Safety 92(11):1503-1520, Elsevier, 2007.
- A. Aldini, R. Gorrieri. A Study About Trade-off Between Performance and Security in an Internet Audio Mechanism. Global Computing: Programming Environments, Languages, Security and Analysis of Systems, C. Priami, ed., LNCS 2874:203-228, Springer, 2003.

References

- A. Aldini, M. Bernardo. Component-Oriented Verification of Noninterference. Journal of Systems Architecture 57:282-293, Elsevier, 2011.
- A. Aldini, M. Bernardo. A General Framework for Nondeterministic, Probabilistic, and Stochastic Noninterference. Joint Workshops on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (ARSPA-WITS'09), LNCS 5511:18-33, Springer, 2009.
- A. Aldini, A. Di Pierro. Estimating the Maximum Information Leakage. International Journal of Information Security 7(3):219-242, Springer, 2008.