Enforcing security policies on components

Ilaria Matteucci

Istituto di Informatica e Telematica
Consiglio Nazionale delle Ricerche
IIT-CNR, Pisa, Italy

Joint work with **Fabio Martinelli**

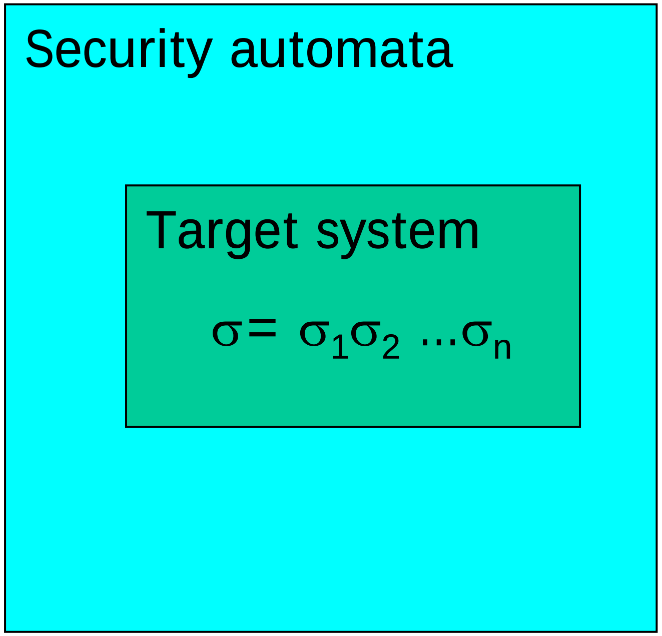
Outline

- Security automata
- Our *specification* and *verification* approach
  - Open systems analysis
  - Partial model checking
- Controller operators
  - Process algebra operators
  - Semantics of controllers
- *Synthesis* approach
- Conclusion and future work

How security automata works

A security automaton is a triple $(Q; q_0; \delta)$ where Q is a set of states, q_0 is the initial one and δ: $Act \times Q \rightarrow Q$, where Act is a set of actions and δ is the transition function.

It processes a sequence $\sigma = \sigma_1, \sigma_2 \ldots$ of input actions that has infinite or finite length.



It works by monitoring the target system and terminating any execution that is about to violate the security policy being enforced.

Security automata (Ligatti & al.)

The **truncation automaton** (similar to Schneider's ones) can recognize bad sequences of actions and halts program execution before security property is violated, but cannot otherwise modify program behavior.

The **suppression automaton** can halt program execution and suppress individual program actions without terminating the program outright.

The **insertion automaton** can insert a sequence of actions into the program actions stream as well as terminate the program.

The **edit automaton** combines the power of suppression and insertion automata. It can truncate actions sequences and insert or suppress security-relevant actions at will.

Our goals

- We introduce process algebra operators able to mimic the behavior of the security automata.
- We can automatically build programs that allow to enforce security properties for whatever unknown **X** .
- We can apply the huge set of security specification and verification techniques.

Process algebra (CCS)

Process algebra (CCS) is used in order to specify a lot of kind of system.

Syntax of expression:

$$\mathbf{P ::= 0 \mid \mathit{A} \mid \mathit{a}.P \mid P+P \mid P||P \mid P/L \mid P[f]}$$

Where **0** is deadlock, ***A*** is a set of name of processes (agents) and $\boldsymbol{a} \in \boldsymbol{Act} = \mathcal{L} \cup \overline{\mathcal{L}} \cup \tau$ where τ is an internal action.

Semantic of CCS

prefix $$\frac{}{a.P \xrightarrow{a} P}$$

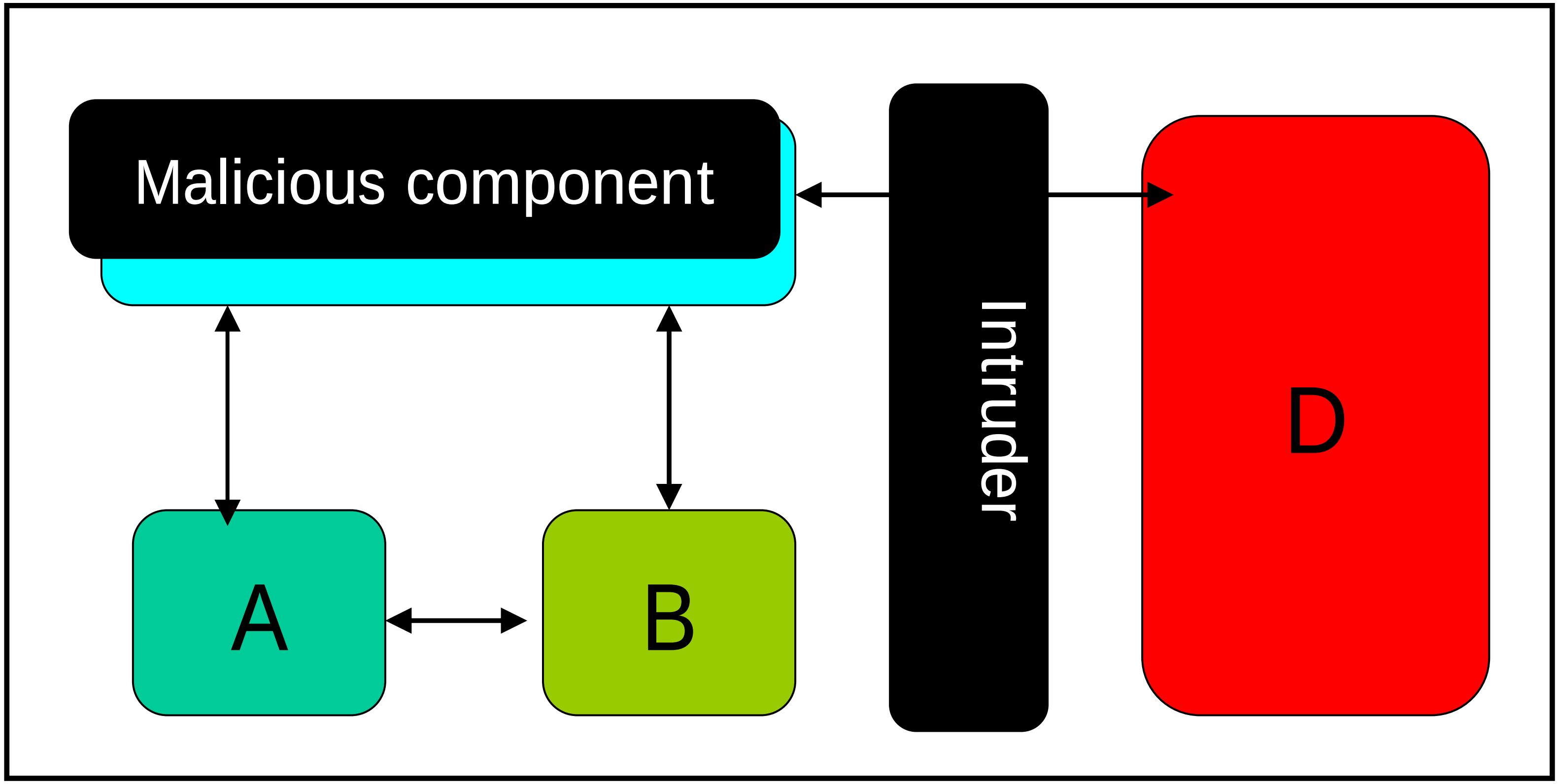
choice $$\frac{P \xrightarrow{a} P'}{P+Q \xrightarrow{a} P'+Q} \quad \frac{Q \xrightarrow{a} Q'}{P+Q \xrightarrow{a} P+Q'}$$

parallel $$\frac{P \xrightarrow{a} P'}{P|Q \xrightarrow{a} P'|Q} \quad \frac{Q \xrightarrow{a} Q'}{P|Q \xrightarrow{a} P|Q'} \quad \frac{Q \xrightarrow{a} Q' \quad P \xrightarrow{\bar{a}} P'}{P|Q \xrightarrow{\tau} P'|Q'}$$

restriction $$\frac{P \xrightarrow{a} P'}{P \backslash L \xrightarrow{a} P' \backslash L} a, \bar{a} \notin L$$

relabeling $$\frac{P \xrightarrow{a} P'}{P[f] \xrightarrow{f(a)} P'[f]}$$

Security specification and verification



Specification: A | B | [ ] | D | [ ]

Open system verification

An open system **S(_)** satisfy a property ϕ iff:

For all X we have $\quad$ **S|X** $\models \phi$

Where ϕ is a logic formula.

X is the unknown entity whose behavior cannot be predicted but whose presence must be considered.

Partial model checking (Andersen '95)

- Given a (finite) system **S**, and a formula ϕ, then we can compute a formula $\phi_{//S}$ s.t.:

$$\mathbf{S} \,|\, \mathbf{X} \models \phi$$

iff

$$\mathbf{X} \models \phi_{//S}$$

  - This is called **partial model checking (PMC)** since the behavior of the whole system, i.e. **S|X**, is only partially evaluated.

PMC for dealing with universal quantification

The presence of universal quantification makes it difficult to check open systems properties:

For all X we have $S | X \models \phi$

It would be easier to verify:

For all X we have $X \models \phi_{//S}$

Which is a validity checking problem of a logic formula.

Through PMC, we can perform a similar reduction.

Enforcing security properties: controller operators

In order to mimic the security automata, we define four process algebra operators, said $\mathbf{Y} \triangleright_{\mathbf{K}} \mathbf{X}$, where $\mathbf{K} \in$ **{T,S,I,E}** that have the same behavior of truncation, suppression, insertion and edit automata, respectively.

It can permit to control the behavior of the component **X**, given the behavior of a control program **Y**.

Semantics of co… s

The **truncation automaton** can recognize bad sequences of actions…
exec…
propert…
other…

The **suppression automaton** can halt program execution and supp…
progra…
termi…

The **edit automaton** combines the power of suppression and Insertion automata. It can truncate actions sequences and insert or suppress security-relevant actions at will

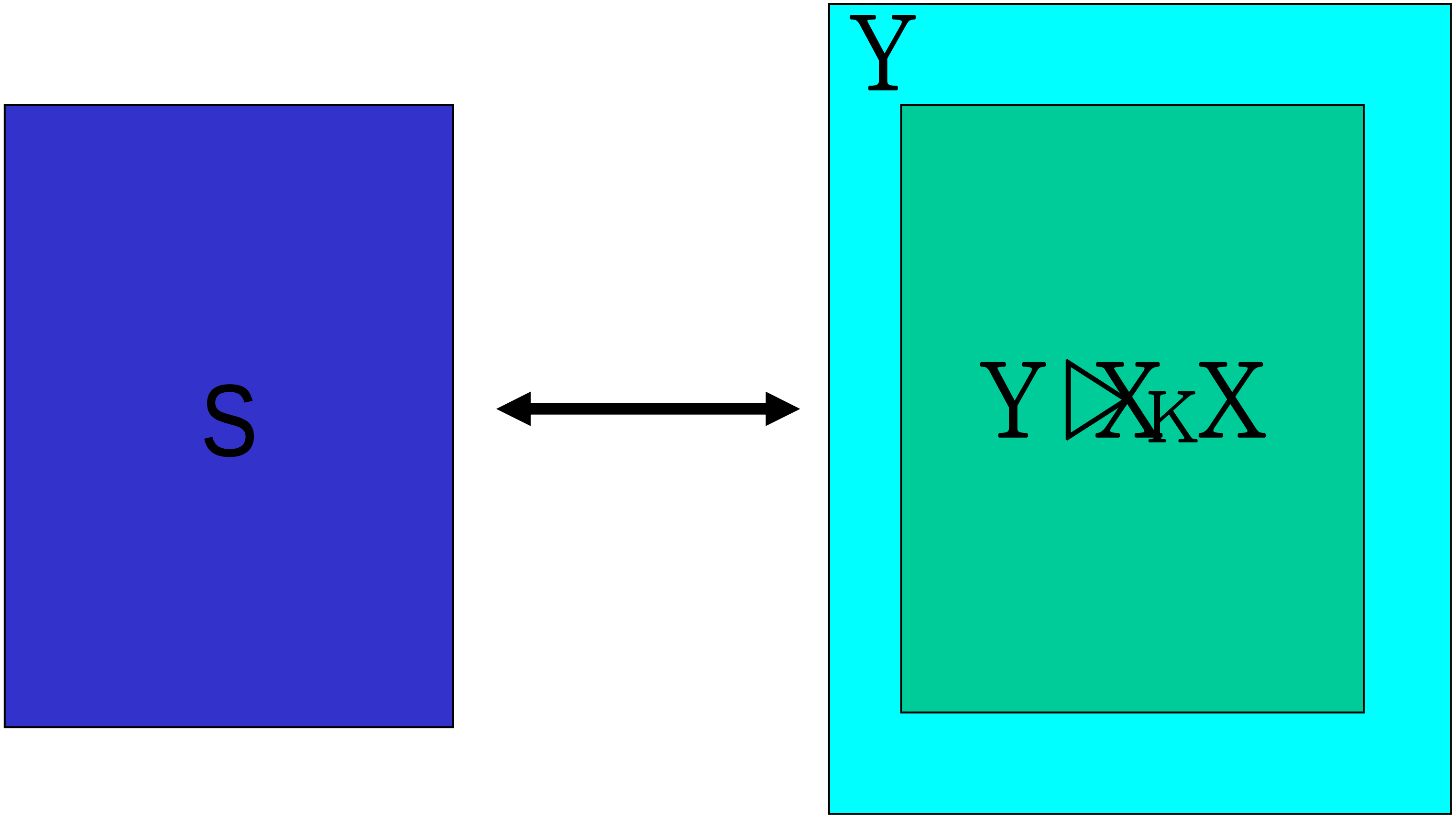
$\triangleright_T$
$$\frac{E \xrightarrow{a} E' \quad F \xrightarrow{a} F'}{E \triangleright_T F \xrightarrow{a} E' \triangleright_T F'}$$

$\triangleright_S$
$$\frac{E \xrightarrow{a} E' \quad F \xrightarrow{a} F'}{E \triangleright_S F \xrightarrow{a} E' \triangleright_S F'} \qquad \frac{E \xrightarrow{-a} E' \quad F \xrightarrow{a} F'}{E \triangleright_S F \xrightarrow{\tau} E' \triangleright_S F'}$$

$\triangleright_I$
$$\frac{E \xrightarrow{a} E' \qquad F \xrightarrow{a} F'}{E \triangleright_I F \xrightarrow{a} E' \triangleright_I F'} \qquad \frac{E \not\xrightarrow{a} E' \qquad E \xrightarrow{+a.b} E' \quad F \xrightarrow{a} F'}{E \triangleright_I F \xrightarrow{b} E' \triangleright_I F}$$

$\triangleright_E$
$$\frac{E \xrightarrow{a} E' \quad F \xrightarrow{a} F'}{E \triangleright_E F \xrightarrow{a} E' \triangleright_E F'} \qquad \frac{E \xrightarrow{-a} E' \quad F \xrightarrow{a} F'}{E \triangleright_E F \xrightarrow{\tau} E' \triangleright_E F'} \qquad \frac{E \not\xrightarrow{a} E' \quad E \xrightarrow{+a.b} E' \qquad F \xrightarrow{a} F'}{E \triangleright_E F \xrightarrow{b} E' \triangleright_E F}$$

Controller operator $\triangleright_K$



Specification: $S|(Y \triangleright_K X)$

Synthesis of the program controller (1)

A system $\mathbf{S} \,|\, (\mathbf{Y} \triangleright_{\mathbf{K}} \mathbf{X})$ always enjoys the desired security property even if **X** tries to break the security property. Thus, a control program **Y** is s.t.:

$$\text{For all } \mathbf{X} \text{ we have } (\mathbf{S} \,|\, (\mathbf{Y} \triangleright_{\mathbf{K}} \mathbf{X}) \models \phi$$

Equivalently, by **partial model checking** we get:

$$\exists \mathbf{Y} \,\forall \mathbf{X}\, (\mathbf{Y} \triangleright_{\mathbf{K}} \mathbf{X}) \models \phi' \qquad \mathbf{(2)}$$

where $\phi' = \phi_{//\mathbf{S}}$

Synthesis of the program controller (2)

For every $\mathbf{K} \in \{\mathbf{T}, \mathbf{S}, \mathbf{I}, \mathbf{E}\}$, $\mathbf{Y} \triangleright_{\mathbf{K}} \mathbf{X} \preceq \mathbf{Y}[\mathbf{f}_{\mathbf{K}}]$ holds, where $\mathbf{f}_{\mathbf{K}}$ is a relabeling function depending on **K**. In particular $\mathbf{f}_{\mathbf{T}}$ is the identity function on *Act* and

$$f_S(a) = \begin{cases} a & \text{if } a \in Act \\ \tau & \text{if } a = -a \end{cases} \quad f_I(a) = \begin{cases} a & \text{if } a \in Act \\ \tau & \text{if } a = +a \end{cases} \quad f_E(a) = \begin{cases} a & \text{if } a \in Act \\ \tau & \text{if } a = \{+a, -a\} \end{cases}$$

For safety properties formulae, if $\mathbf{F} \preceq \mathbf{E}$ then $\mathbf{E} \models \phi \Rightarrow \mathbf{F} \models \phi$, the equation **(2)** becomes

$$\exists\, \mathbf{Y}\ \mathbf{Y}[\mathbf{f}_{\mathbf{K}}] \models \phi'$$

Through PMC we obtain

$$\exists\, \mathbf{Y}\ \ \mathbf{Y} \models \phi''$$

where $\phi'' = \phi'_{//\mathbf{f_K}}$ for every **K**

Given a formula ϕ it is possible to decide in exponential time in length of ϕ if there exists a model of ϕ and it is also possible to give an example of it.

Benefits of pmc and enforcing

- We have a method to constrain only un-trusted components instead of the whole system.
  - Other approaches deal with the problem of monitoring un-trusted components of a systems to enjoy a given property, by treating it as the whole system interest.
- Building a reference monitor for a whole distributed architecture could not be possible, while it could be possible for some its components
- We can find minimum necessary and sufficient conditions that components of a systems must enjoy.

Conclusion and Future work

- We model the security automata of Schneider and Ligatti and al. by defining controller operators in process algebra.
- With respect to prior works in the area we also add the possibility to automatically build enforcing mechanisms.
- We can synthesize enforcing mechanisms also for ***parameterized systems*** and ***systems in timed settings***

Future work: The theory developed here can be extended to deal with more than one unknown component.

Thank you!!!