Enforceable Security Policies Revisited

David Basin[1] **Vincent Jugé**[2]
Felix Klaedtke[1] Eugen Zălinescu[1]

[1]Institute of Information Security, ETH Zurich, Switzerland

[2]MINES ParisTech, France

POST 2012

Security Policies Come in all Shapes and Sizes

History-Based Access Control

Chinese Wall

Information Flow

Separation of Duty

Business Regulations

Data Usage

Privacy

Estonian Law

...

Security Policies Come in all Shapes and Sizes

History-Based Access Control

Chinese Wall

Information Flow

Separation of Duty

Business Regulations

Data Usage

Privacy

Estonian Law

...

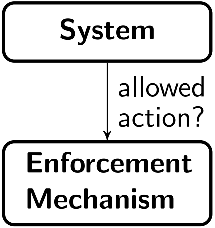
Which of these are enforceable?

Enforcement by Execution Monitoring

Enforceable Security Policies
F. Schneider, TISSEC 2000

Abstract Setting

- System iteratively executes actions
- Enforcement mechanism intercepts them (prior to their execution)
- Enforcement mechanism terminates system in case of violation



Main Concerns

- match with reality?
- enforceable $\Rightarrow$ / $\not\Leftarrow$ safety

Follow-Up Work

- *SASI Enforcement of Security Policies*
  Ú. Erlingsson and F. Schneider, NSPW 1999
- *IRM Enforcement of Java Stack Inspection*
  Ú. Erlingsson and F. Schneider, S&P 2000
- *Access Control by Tracking Shallow Execution History*
  P. Fong, S&P 2004
- *Edit Automata: Enforcement Mechanisms for Run-Time Security Properties*
  J. Ligatti, L. Bauer, and D. Walker, IJIS 2005
- *Computability classes for enforcement mechanisms*
  K. Hamlen, G. Morrisett, and F. Schneider, TISSEC 2006
- *Run-Time Enforcement of Nonsafety Policies*
  J. Ligatti, L. Bauer, and D. Walker, TISSEC 2009
- *A Theory of Runtime Enforcement, with Results*
  J. Ligatti and S. Reddy, ESORICS 2010
- *Do you really mean what you actually enforced?*
  N. Bielova and F. Massacci, IJIS 2011
- *Runtime Enforcement Monitors: Composition, Synthesis and Enforcement Abilities*
  Y. Falcone, L. Mounier, J.-C. Fernandez, and J.-L. Richier, FMSD 2011
- *Service Automata*
  R. Gay, H. Mantel, and B. Sprick, FAST 2011
- *Enforceable Policies Revisited*
  D. Basin, V. Jugé, F. Klaedtke, and E. Zălinescu, POST 2012
- ...

Enforcement by Execution Monitoring (Fundamental Open Question)

Match with Reality

- Can we refine Schneider's abstraction?

Limited Understanding

- Schneider: enforceable $\Rightarrow$ safety
- Necessary and sufficient condition?

Our Solution

Refined abstract setting by distinguishing between **observable** and **controllable** actions:

- **clock tick**
- **administrative actions**
- **user actions**

Contributions

1. **Formalization and Characterization of Enforceability**
2. Realizability of Enforcement Mechanisms

Refined Abstract Setting

Actions

Set of actions $\Sigma = \mathbf{O} \cup \mathbf{C}$:

- $\mathbf{O} = \{\text{observable actions}\}$
- $\mathbf{C} = \{\text{controllable actions}\}$

Traces

Trace universe $\mathbf{U} \subseteq \Sigma^{\infty}$:

- $\mathbf{U} \neq \emptyset$
- $\mathbf{U}$ prefix-closed

Example: $\textbf{request} \cdot \textbf{tick} \cdot \textbf{deliver} \cdot \textbf{tick} \cdot \textbf{tick} \cdot \textbf{request} \cdot \textbf{deliver} \cdot \textbf{tick} \ldots \in \mathbf{U}$

Refined Abstract Setting

Actions

Set of actions $\Sigma = \mathbf{O} \cup \mathbf{C}$:

- $\mathbf{O}$ = {observable actions}
- $\mathbf{C}$ = {controllable actions}

Traces

Trace universe $\mathbf{U} \subseteq \Sigma^\infty$:

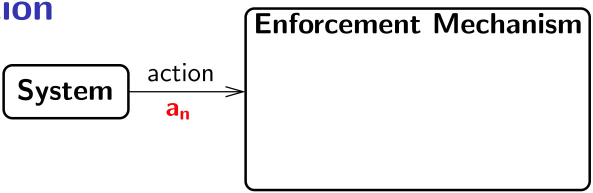
- $\mathbf{U} \neq \emptyset$
- $\mathbf{U}$ prefix-closed

Example: **request** · **tick** · **deliver** · **tick** · **tick** · **request** · **deliver** · **tick** . . . $\in \mathbf{U}$

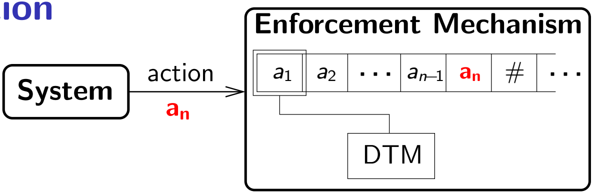
Requirements (on the Enforcement Mechanism)

- **Computability**: Make decisions
- **Soundness**: Prevent policy-violating traces
- **Transparency**: Allow policy-compliant traces

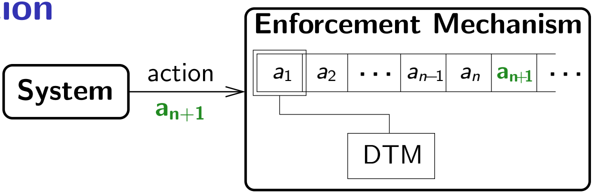
Formalization



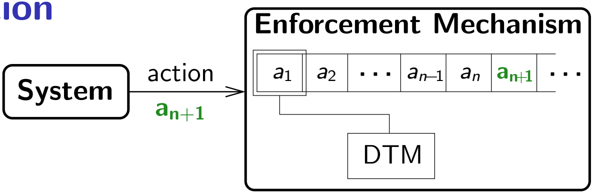
Formalization



Formalization



Formalization



Definition

$P \subseteq (\mathbf{O} \cup \mathbf{C})^\infty$ is **enforceable** in $\mathbf{U}$ $\quad\overset{\text{def}}{\Longleftrightarrow}\quad$ exists DTM $\mathcal{M}$ with

1. $\varepsilon \in L(\mathcal{M})$
   "$\mathcal{M}$ accepts the empty trace"
2. $\mathcal{M}$ halts on inputs in $\big(trunc(L(\mathcal{M})) \cdot (\mathbf{O} \cup \mathbf{C})\big) \cap \mathbf{U}$
   "$\mathcal{M}$ either permits or denies intercepted action"
3. $\mathcal{M}$ accepts inputs in $\big(trunc(L(\mathcal{M})) \cdot \mathbf{O}\big) \cap \mathbf{U}$
   "$\mathcal{M}$ permits intercepted observable action"
4. $limitclosure\big(trunc(L(\mathcal{M}))\big) \cap \mathbf{U} = P \cap \mathbf{U}$
   "soundness ($\subseteq$) and transparency ($\supseteq$)"

Examples

Setting

- Controllable actions: $\mathbf{C} = \{\mathbf{login}, \mathbf{request}, \mathbf{deliver}\}$
- Observable actions: $\mathbf{O} = \{\mathbf{tick}, \mathbf{fail}\}$
- Set of actions: $\Sigma = \mathbf{C} \cup \mathbf{O}$
- Trace universe: $\mathbf{U} = \Sigma^* \cup (\Sigma^* \cdot \{\mathbf{tick}\})^\omega$

Policies

1. "**login** must not happen within 3 time units after a **fail**."
2. "each **request** must be followed by a **deliver** within 3 time units."

Examples

Setting

- Controllable actions: $\mathbf{C} = \{\mathbf{login}, \mathbf{request}, \mathbf{deliver}\}$
- Observable actions: $\mathbf{O} = \{\mathbf{tick}, \mathbf{fail}\}$
- Set of actions: $\Sigma = \mathbf{C} \cup \mathbf{O}$
- Trace universe: $\mathbf{U} = \Sigma^* \cup (\Sigma^* \cdot \{\mathbf{tick}\})^\omega$

Policies

1. "**login** must not happen within 3 time units after a **fail**."
   $\Rightarrow$ **enforceable**
2. "each **request** must be followed by a **deliver** within 3 time units."
   $\Rightarrow$ **not enforceable**

Evolution of Safety

Early Definitions

- L. Lamport, 1977: "A **safety property** is one which states that something bad will *not* happen."
- B. Alpern and F. Schneider, 1986: A property $P \subseteq \Sigma^\omega$ is ω-**safety** if $\forall \sigma \in \Sigma^\omega .\, \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N}.\, \forall \tau \in \Sigma^\omega .\, \sigma^{<i} \cdot \tau \notin P\big)$
- Folklore: A property $P \subseteq \Sigma^\infty$ is ∞-**safety** if $\forall \sigma \in \Sigma^\infty .\, \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N}.\, \forall \tau \in \Sigma^\infty .\, \sigma^{<i} \cdot \tau \notin P\big)$
- T. Henzinger, 1992: A property $P \subseteq \Sigma^\omega$ is **safety in** $U \subseteq \Sigma^\omega$ if $\forall \sigma \in U .\, \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N}.\, \forall \tau \in \Sigma^\omega .\, \sigma^{<i} \cdot \tau \notin P \cap U\big)$

Evolution of Safety

Early Definitions

- L. Lamport, 1977: "A **safety property** is one which states that something bad will *not* happen."
- B. Alpern and F. Schneider, 1986: A property $P \subseteq \Sigma^\omega$ is ω-**safety** if
  $\forall \sigma \in \Sigma^\omega.\, \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N}.\, \forall \tau \in \Sigma^\omega.\, \sigma^{<i} \cdot \tau \notin P\big)$
- Folklore: A property $P \subseteq \Sigma^\infty$ is ∞-**safety** if
  $\forall \sigma \in \Sigma^\infty.\, \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N}.\, \forall \tau \in \Sigma^\infty.\, \sigma^{<i} \cdot \tau \notin P\big)$
- T. Henzinger, 1992: A property $P \subseteq \Sigma^\omega$ is **safety in** $U \subseteq \Sigma^\omega$ if
  $\forall \sigma \in U.\, \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N}.\, \forall \tau \in \Sigma^\omega.\, \sigma^{<i} \cdot \tau \notin P \cap U\big)$

Refined Definition

A property $P \subseteq \Sigma^\infty$ is ∞-**safety** if
$\forall \sigma \in \Sigma^\infty.\, \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N}.\, \forall \tau \in \Sigma^\infty.\, \sigma^{<i} \cdot \tau \notin P\big)$

Evolution of Safety

Early Definitions

- L. Lamport, 1977: "A **safety property** is one which states that something bad will *not* happen."
- B. Alpern and F. Schneider, 1986: A property $P \subseteq \Sigma^\omega$ is ω-**safety** if $\forall \sigma \in \Sigma^\omega .\, \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N}. \forall \tau \in \Sigma^\omega .\, \sigma^{<i} \cdot \tau \notin P\big)$
- Folklore: A property $P \subseteq \Sigma^\infty$ is ∞-**safety** if $\forall \sigma \in \Sigma^\infty .\, \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N}. \forall \tau \in \Sigma^\infty .\, \sigma^{<i} \cdot \tau \notin P\big)$
- T. Henzinger, 1992: A property $P \subseteq \Sigma^\omega$ is **safety in** $U \subseteq \Sigma^\omega$ if $\forall \sigma \in U .\, \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N}. \forall \tau \in \Sigma^\omega .\, \sigma^{<i} \cdot \tau \notin P \cap U\big)$

Refined Definition

A property $P \subseteq \Sigma^\infty$ is **U-safety** if
$\forall \sigma \in \mathbf{U} .\, \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N}. \forall \tau \in \Sigma^\infty .\, \sigma^{<i} \cdot \tau \notin P \cap \mathbf{U}\big)$

Evolution of Safety

Early Definitions

- L. Lamport, 1977: "A **safety property** is one which states that something bad will *not* happen."
- B. Alpern and F. Schneider, 1986: A property $P \subseteq \Sigma^\omega$ is ω-**safety** if $\forall \sigma \in \Sigma^\omega . \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N} . \forall \tau \in \Sigma^\omega . \sigma^{<i} \cdot \tau \notin P\big)$
- Folklore: A property $P \subseteq \Sigma^\infty$ is ∞-**safety** if $\forall \sigma \in \Sigma^\infty . \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N} . \forall \tau \in \Sigma^\infty . \sigma^{<i} \cdot \tau \notin P\big)$
- T. Henzinger, 1992: A property $P \subseteq \Sigma^\omega$ is **safety in** $U \subseteq \Sigma^\omega$ if $\forall \sigma \in U . \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N} . \forall \tau \in \Sigma^\omega . \sigma^{<i} \cdot \tau \notin P \cap U\big)$

Refined Definition

A property $P \subseteq \Sigma^\infty$ is **(U, O)-safety** if
$\forall \sigma \in \mathbf{U} . \sigma \notin P \rightarrow \big(\exists i \in \mathbb{N} . \sigma^{<i} \notin \Sigma^* \cdot \mathbf{O} \wedge \forall \tau \in \Sigma^\infty . \sigma^{<i} \cdot \tau \notin P \cap \mathbf{U}\big)$

Intuition: "P is safety in **U** *and* bad things are not caused by an **O**"

Safety and Enforceability

Theorem

Let P be a property and $\mathbf{U}$ a trace universe with $\mathbf{U} \cap \Sigma^*$ decidable.

P is $(\mathbf{U}, \mathbf{O})$-enforceable $\iff$

1. P is $(\mathbf{U}, \mathbf{O})$-safety,
2. $\mathrm{pre}_*(P \cap \mathbf{U})$ is a decidable set, and
3. $\varepsilon \in P$.

Schneider's "characterization": only $\Rightarrow$ for (1), where $\mathbf{U} = \Sigma^\infty$ and $\mathbf{O} = \emptyset$

Contributions

1. Formalization and Characterization of Enforceability
2. **Realizability of Enforcement Mechanisms**

Realizability of Enforcement Mechanisms

Fundamental Algorithmic Problems

Given a specification of a policy.

- Is this policy enforceable?
- If yes, can we synthesize an enforcement mechanism for it?
- With what complexity can we do so?

Some Results

Deciding if P is (**U**, **O**)-enforceable when both **U** and P are given as

- PDAs is **undecidable**.
- FSAs is **PSPACE-complete**.
- LTL formulæ is **PSPACE-complete**.
- MLTL formulæ is **EXPSPACE-complete**.

Checking Enforceability and Safety (PDA and FSA)

Checking Enforceability

Let **U** and P be given as PDAs or FSAs $\mathcal{A}_{\mathsf{U}}$ and $\mathcal{A}_P$.

1. $\mathrm{pre}_*(L(\mathcal{A}_P) \cap L(\mathcal{A}_{\mathsf{U}}))$ is known to be decidable
2. check whether $\varepsilon \in L(\mathcal{A}_P)$
3. check whether $L(\mathcal{A}_P)$ is $(L(\mathcal{A}_{\mathsf{U}}), \mathsf{O})$-safety

Checking Safety

Let **U** and P be given as PDAs or FSAs $\mathcal{A}_{\mathsf{U}}$ and $\mathcal{A}_P$.

- PDAs: undecidable in general
- FSAs: generalization of standard techniques

Checking Enforceability and Safety (LTL and MLTL)

Checking Enforceability

Let **U** and P be given as LTL or MLTL formulæ $\varphi_{\mathbf{U}}$ and φ_P.

1. $\mathrm{pre}_*(L(\varphi_P) \cap L(\varphi_{\mathbf{U}}))$ is known to be decidable
2. check whether $\varepsilon \in L(\varphi_P)$
3. check whether $L(\varphi_P)$ is $(L(\varphi_{\mathbf{U}}), \mathbf{O})$-safety

Checking Safety

Let **U** and P be given as LTL or MLTL formulæ $\varphi_{\mathbf{U}}$ and φ_P.

1. translate $\varphi_{\mathbf{U}}$ and φ_P into FSAs $\mathcal{A}_{\mathbf{U}}$ and $\mathcal{A}_P$
2. use the results of the previous slide on $\mathcal{A}_{\mathbf{U}}$ and $\mathcal{A}_P$
3. perform all these calculations on-the-fly

Conclusion

Summary

- Formalization of enforceability in a refined abstract setting
- Characterization of enforceability
- Realizability problem for enforcement

Future Work

- Investigate more powerful enforcement mechanisms
- Investigate more expressive specification languages
- Provide tool support