Which Security properties are Enforceable by Monitors?

by
Raphaël Khoury

Laval University

19 April 2013

Outline

Outline

General Framework

- A system is modeled by a (possibly infinite) set of actions Σ.
- An execution is a finite or infinite sequence of actions from Σ.
- A security policies of interest are subsets of valid sequences $\hat{\mathcal{P}}$ termed properties.
- The monitor is an automaton that receives a sequence as input, and outputs another sequence.
- We let σ and τ range over possible executions, and $\mathcal{A}(\sigma)$ denote the output of monitor $\mathcal{A}$ when its input is σ. We write $\hat{\mathcal{P}}(\sigma)$ to indicate that the sequence σ respects the security policy $\hat{\mathcal{P}}$.

Framework and Review of the Literature

When can we consider that a monitor enforces a security Policy? An effective enforcement paradigm must be based on the following 2 principles (from Ligatti et al.):

Correction The output sequence is valid.

Transparency The semantics of a valid input sequence is preserved. An equivalence relation between executions limits the monitor's ability to transform sequences.

Outline

Model by Schneider

- According to Schneider, a monitor can be modeled by an automaton which observes the execution of the target program and aborts the execution to prevent a violation of the security property from occurring.
  - The monitor can only consider the execution in progress.
  - It possesses no information about the possible future behavior of the program
  - It can only react to a potential violation of the security policy by aborting the execution
- In this context, only safety properties are enforceable.

Automata representation of Monitors

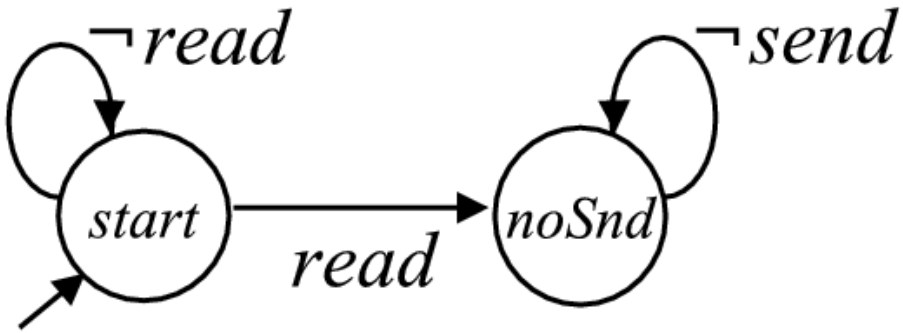
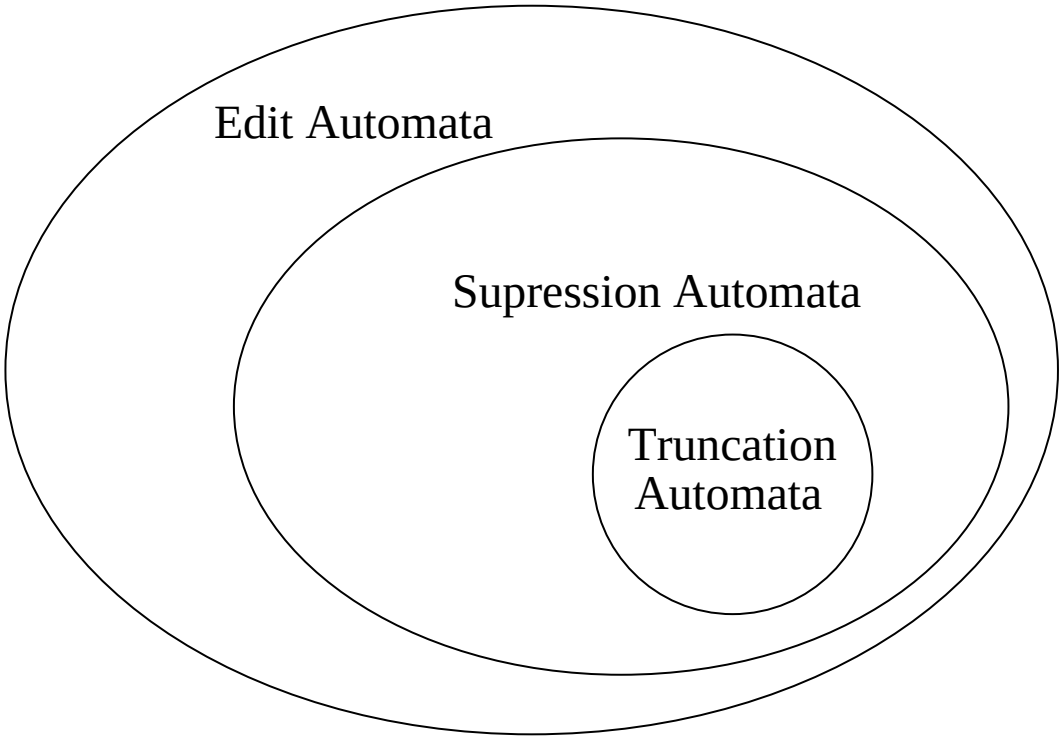


Figure 1. "No messages sent after reading a file".

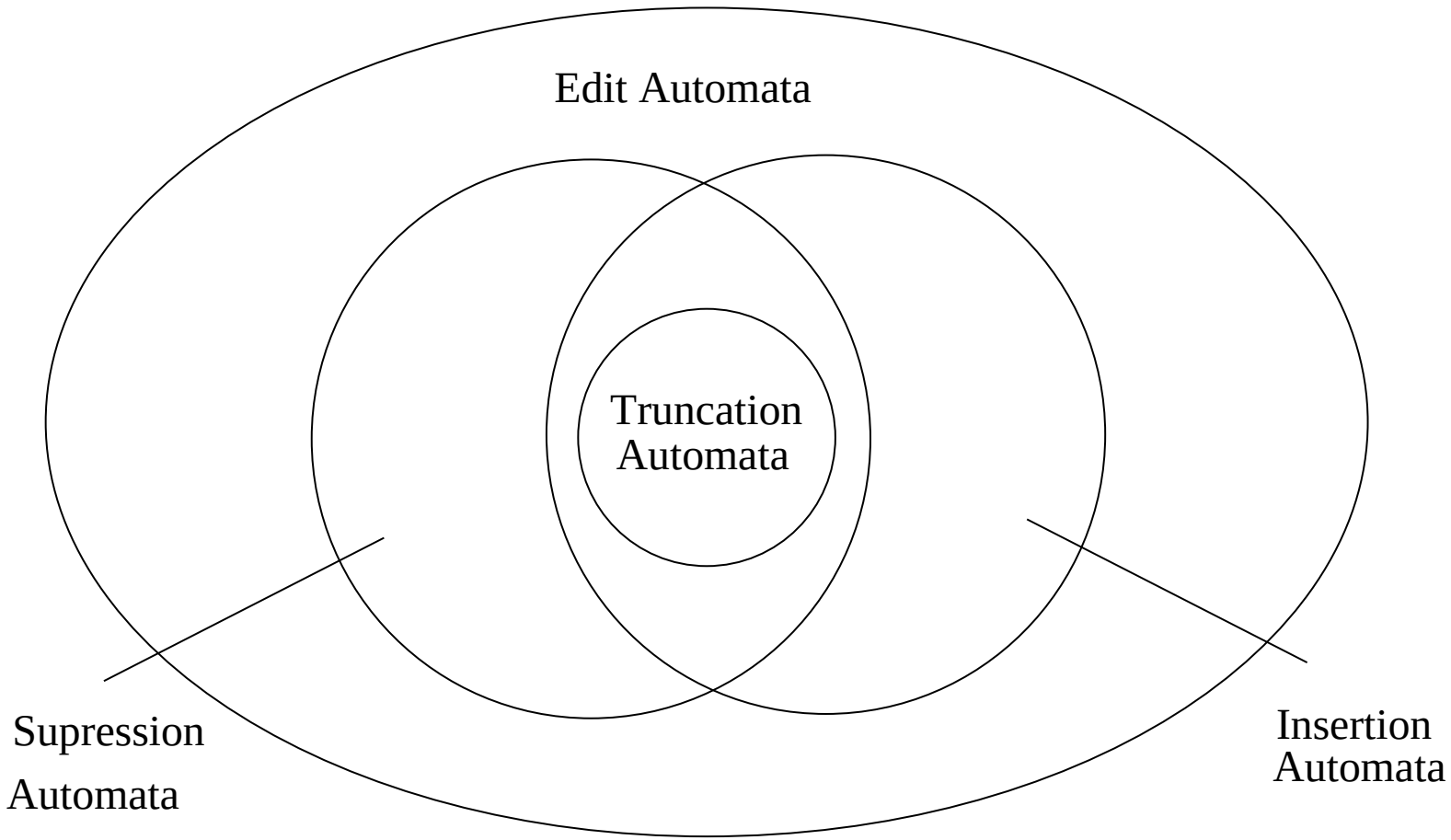
Model by Ligatti et al.

- Ligatti et al. extend Schneider's model along three axes:
  - According to the means available to the monitor to react to a possible violation;
  - According to the information at the disposal of the monitor about the program's possible behavior(non-uniform context);
  - According to its capacity to transform valid sequences into equivalent sequences.
- The set of policies enforceable by these mechanisms increases in the non-uniform context.
- More diverse reaction mechanisms can extend the range of enforceable policies, but only if the monitor is equipped with a sufficiently flexible equivalence relation.

Results-Precise nonuniform enforcement



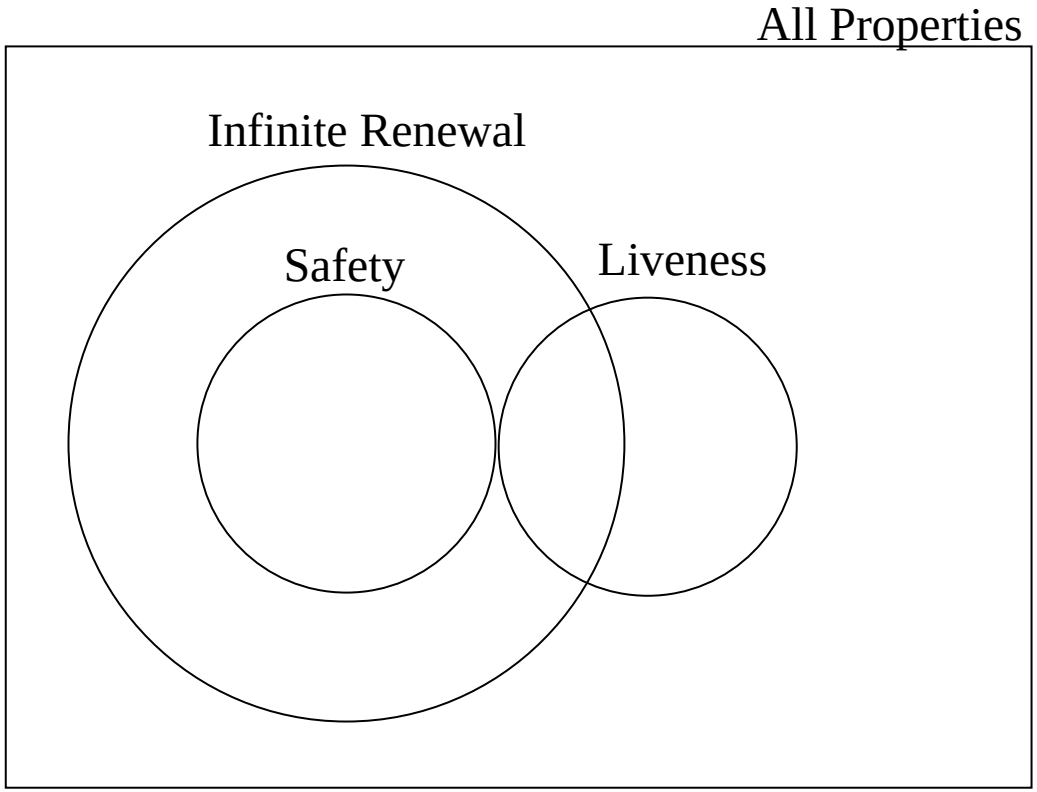
Results-Effective Enforcement enforcement



The Edit Automaton and infinite sequences

- In a subsequent study, Ligatti et al. define the set of properties enforceable by an edit automaton, using syntactic equality as its equivalence relation.
- This corresponds to the set of infinite renewal property. A property is in this set if every valid sequence has infinitely many valid prefixes, while every infinite invalid sequences has only finitely many such prefixes. All properties over finite sequences are in this set.

Results-Infinite Renewal



Outline

Kim et al.

- Kim et al. observe that only a subset of safety properties are truncation enforceable since there exists properties for which the monitor cannot detect the violation.
- A property $\hat{\mathcal{P}}$ is enforceable by a security automaton iff $\hat{\mathcal{P}}$ is a safety property and the set $\Sigma^* \backslash \{pref(Property)\}$ is recursively enumerable.
- This is the class of co-recursively enumerable languages (coRE).

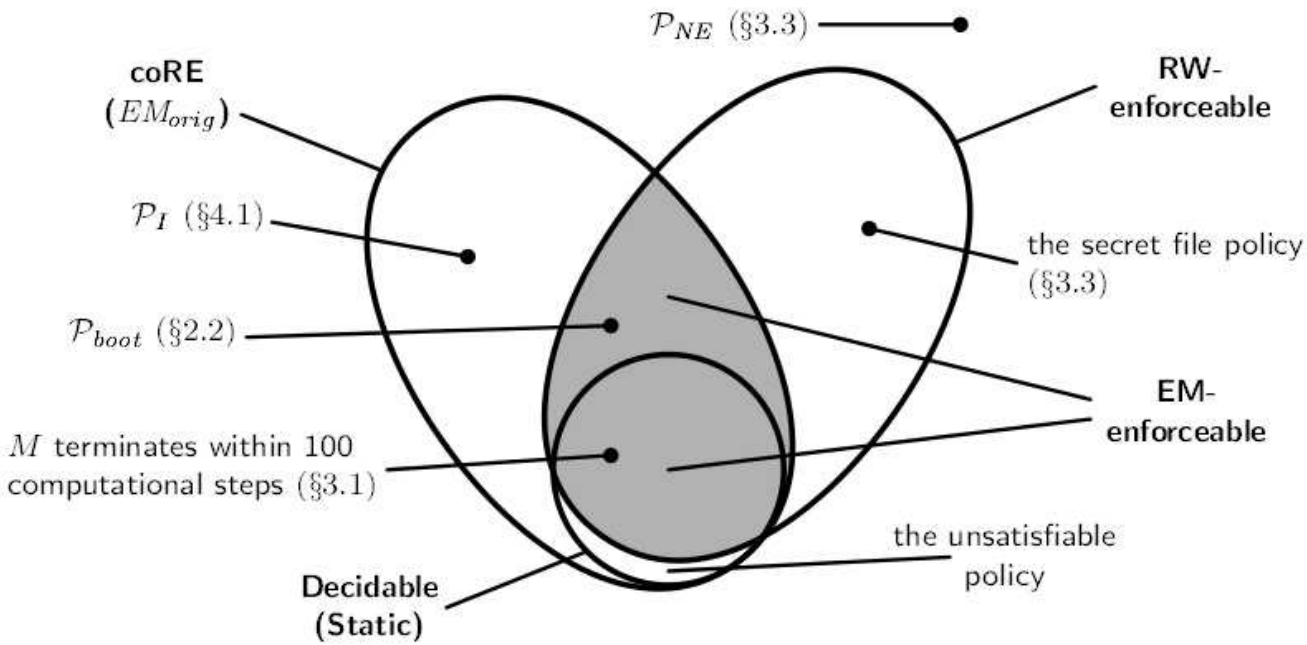
Hamlen et al.

- Hemlen et al. proceed to compare the enforcement power of monitors, static analysis and code rewriting and link them to computability classes.
- Statically enforceable properties are those for which there exists a total computable procedure that decides them. This coincides with the the class of recursively decidable properties.
- Code rewriting is equivalent to effective$_{\simeq}$ enforcement, it includes all recursively decidable properties except the empty property does not correspond to any computability class.
- The class of co-recursively enumerable languages is actually a superset of the set of properties enforceable by monitors, since in some cases, the monitor might be able to detect the violation but unable to react.

Hamlen et al.

- A better characterization of the properties enforceable by monitors is the intersection of coRE and of the set of properties enforceable by rewriters.
- Properties in this intersection exhibit a particular behavior termed benevolence.
- A property is benevolent if there exists a decision procedure that rejects any invalid prefix of an invalid execution, but accepts any valid prefix of a valid execution.

Hamlen et al. -Results



Fong- The Shallow History Automaton

- Fong considers monitors that only records the shallow history (i.e. the unordered set of security relevant events performed by the target program).
- He proposes a monitor-based model, the shallow history automaton (SHA), to study their enforcement power.
- The set of SHA-enforceable properties is necessarily a strict subset of safety properties, but it does include many interesting real-life properties:
  - the Chinese Wall Policy
  - the Low-Water-Mark policy
  - the One-out-of-k authorization
  - the Assured Pipelines policy

Talhi et al.- The Bounded History Automaton

- Talhi et al. study the enforcement power of a monitor operating with a memory of bounded size.
- The propose two new models, the Bounded Security Automata (BSA) and the by Bounded Edit Automata (BEA), to study such monitors.
- The set of properties enforceable by BSA and BEA are naturally subsets to the set of properties enforceable by their unrestricted counterparts.
- The set of properties enforceable by bounded automata increases monotonously as a larger memory is made available to the monitor.
- There exists a close connection between the set of properties enforceable by BHA and a class of properties termed locally testable properties.

Beauquier et al.- The finite Automaton

- Beauquier et al. examine the set of properties enforceable by monitors whose memory is finite but unbounded.
- They focus on effective enforcement and on uniform systems containing both finite and infinite sequences with $=$ as the equivalence relation.
- They find that the set of enforceable properties to be intermediate to those enforceable by the BEA and by the edit automaton.

Outline

Review of the Literature

First idea: precise enforcement (from Schneider) Every action of a valid sequence must be output in lockstep. $\forall \sigma \in \Sigma^{\infty}$

- $\hat{\mathcal{P}}(\mathcal{A}(\sigma))$
- $\hat{\mathcal{P}}(\sigma) \Rightarrow \forall i : \mathcal{A}(\sigma_i)$

Review of the Literature

Second Idea: effective$_{\cong}$ enforcement (from Ligatti et al.) The output must be equivalent if the input is valid. $\forall \sigma \in \Sigma^{\infty}$

1. $\hat{\mathcal{P}}(\mathcal{A}(\sigma))$
2. $\hat{\mathcal{P}}(\sigma) \Rightarrow \mathcal{A}(\sigma) \cong \sigma$

Solution 1: Iterative Enforcement

One possible solution is to tie the notion of enforcement to a specific policy. Bielova et al. suggest the notion of iterative enforcement for a class of policies called iterative policies:
$\forall \sigma \in \Sigma^{\infty}$

1. $\hat{\mathcal{P}}(\sigma) \Rightarrow \mathcal{A}(\sigma) = \sigma$
2. $\forall \sigma \in \Sigma^{*} : \neg\hat{\mathcal{P}}(\sigma) : \exists \sigma' \succeq \sigma : \hat{\mathcal{P}}(\sigma') \Rightarrow \mathcal{A}(\sigma) = \sigma_o$ where σ_o is the longest valid prefix of σ
3. $\forall \sigma \in \Sigma^{*} : \neg\hat{\mathcal{P}}(\sigma) \wedge \forall \sigma' \succeq \sigma : \neg\hat{\mathcal{P}}(\sigma') : \exists \sigma_b \in \Sigma : \sigma = \sigma_o; \sigma_b \sigma_r \Rightarrow \mathcal{A}(\sigma) = \sigma_o; \mathcal{A}(\sigma_r)$ where σ_o is the longest valid prefix of σ, and σ_b is the smallest sequence s.t. after deleting it from σ, the resulting sequence is valid.

Informally, a monitor iteratively enforces by suppression an iterative property $\hat{\mathcal{P}}$ iff every valid transaction is output and every invalid transaction is suppressed.

Solution 2: Syntactic Constraints

We can limit the number of insertions and suppressions performed by the monitor when correcting an invalid sequence. Formally, this is done by adding a *predictability* requirement to the definition of enforcement.

Predictability: A enforcement mechanism is predictable if every trace that is close to some valid trace is mapped into a trace close to the same valid trace.

This distance between traces can be quantified using established metrics such as the Levenshtein distance.

Solution 3: Corrective$_{\simeq}$ Enforcement

1. The monitor should be both :
   - required to output a valid sequence, and
   - forbidden from altering the semantics of the input.
2. Valid behaviors present in an invalid input sequence should be preserved, while minimal alterations are made to correct the input sequence.

Corrective$_{\cong}$ Enforcement

1. Corrective$_{\cong}$ Enforcement
2. An abstraction captures essential properties of the input sequence, which must be preserved, despite the monitor's transformations.
3. These abstractions are grouped into equivalence classes.
4. The output must always be kept equivalent to the input.
5. $\forall \sigma \in \Sigma^{\infty}$
   - $\hat{\mathcal{P}}(\mathcal{A}(\sigma))$
   - $\sigma \cong \mathcal{A}(\sigma)$

Examples

Examples

1. Transactional properties: The equivalence relation is the multiset of valid transactions present in a sequence.
2. Renewal Properties: Two sequences are equivalent if they share the same longest valid prefix, w.r.t. the property of interest.

Solution 4: Corrective$_{\sqsubseteq}$ Enforcement

It is often easier to organise execution sequences into a preorder. The output must be higher or equal to the input w.r.t the preorder. $\forall \sigma \in \Sigma^{\infty}$

1. $\hat{\mathcal{P}}(\mathcal{A}(\sigma))$
2. $\sigma \sqsubseteq \mathcal{A}(\sigma)$

Corrective$_{\sqsubseteq}$ Enforcement

1. We use an abstraction function $\mathcal{F}$ to capture the property of the input sequence which must be preserved throughout the manipulations performed by the monitor.
2. We let $\leq$ stand for a preorder over the codomain of $\mathcal{F}$, and $\sqsubseteq$ stand for a corresponding preorder over the possible execution sequences, s.t. for any two sequences σ, σ'
$\sigma \sqsubseteq \sigma' \Leftrightarrow \mathcal{F}(\sigma) \leq \mathcal{F}(\sigma')$.
3. The monitor is only allowed to replace an invalid sequence σ with a valid sequence σ' if $\sigma \sqsubseteq \sigma'$.

Example - Transactional Properties

Corrective enforcement allows several different enforcement possibilities.

1. We let the abstraction function $\mathcal{F}$ be the function $valid(\sigma)$ which abstracts the multiset of valid transactions from a sequence.
2. Two sequences have the same abstraction if they share the same multiset of valid transactions, regardless of their ordering or of the presence of invalid transactions.
3. $\sigma \sqsubseteq \sigma'$ indicates that $valid(\sigma) \subseteq valid(\sigma')$.
4. This allows the monitor to either suppress invalid transactions or replace them with alternative valid transactions.
5. To be correctively$_{\sqsubseteq}$ enforceable, the set of valid transactions must meet a restriction termed unambiguity.

Other Examples

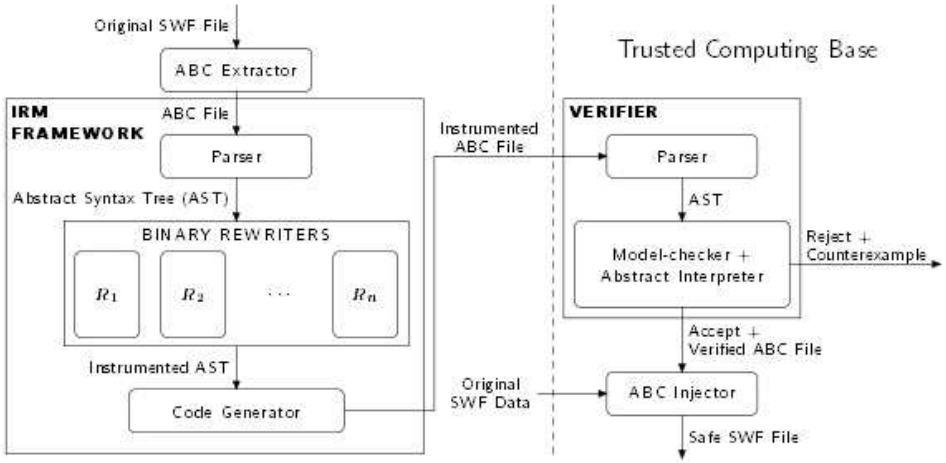
I investigated other possible security policies :

1. The Assured Pipeline Policy
2. The Chinese Wall Policy
3. General Availability

Outline

There still exists several interesting avenues for future research:

1-Can we certify the transparency of a monitor's enforcement?



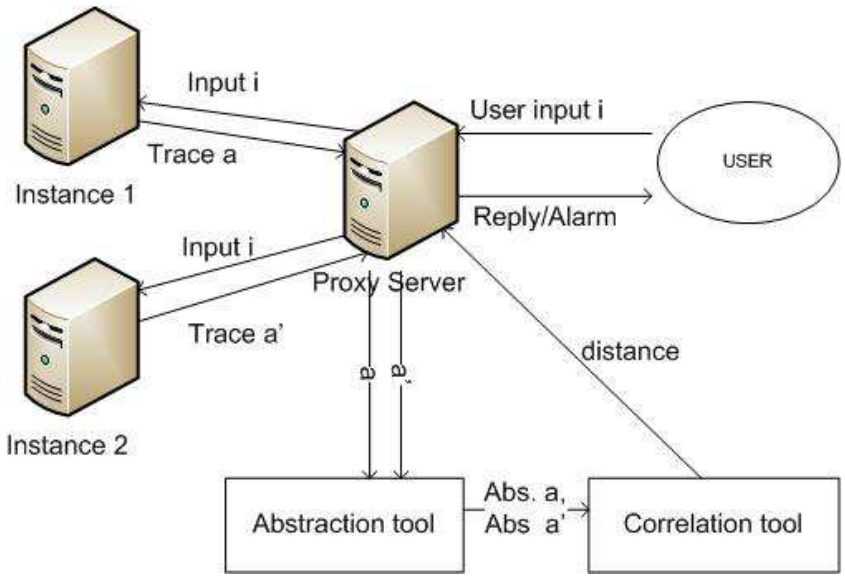
There still exists several interesting avenues for future research:

2-Which properties are enforceable in the presence of trace abstraction?

```
kernel.syscall_entry: 442192.435342606 (/tmp/trace10/kernel_1), 22438, 22438, ./Files, , 29184, 0x0,
SYSCALL { ip = 0xb7fac430, syscall_id = 5 [sys_open+0x0/0x40] }
fs.open: 442192.435348299 (/tmp/trace10/fs_1), 22438, 22438, ./Files, , 29184, 0x0, SYSCALL { fd = 3,
filename = "output.txt" }
kernel.syscall_exit: 442192.435348407 (/tmp/trace10/kernel_1), 22438, 22438, ./Files, , 29184, 0x0,
USER_MODE { ret = 3 }
kernel.syscall_entry: 442192.435350985 (/tmp/trace10/kernel_1), 22438, 22438, ./Files, , 29184, 0x0,
SYSCALL { ip = 0xb7fac430, syscall_id = 4 [sys_write+0x0/0xc0] }
fs.write: 442192.435351307 (/tmp/trace10/fs_1), 22438, 22438, ./Files, , 29184, 0x0, SYSCALL { count =
72, fd = 3 }
kernel.syscall_exit: 442192.435351415 (/tmp/trace10/kernel_1), 22438, 22438, ./Files, , 29184, 0x0,
USER_MODE { ret = 72 }
kernel.syscall_entry: 442192.435351522 (/tmp/trace10/kernel_1), 22438, 22438, ./Files, , 29184, 0x0,
SYSCALL { ip = 0xb7fac430, syscall_id = 6 [sys_close+0x0/0x100] }
fs.close: 442192.435351629 (/tmp/trace10/fs_1), 22438, 22438, ./Files, , 29184, 0x0, SYSCALL { fd = 3 }
```

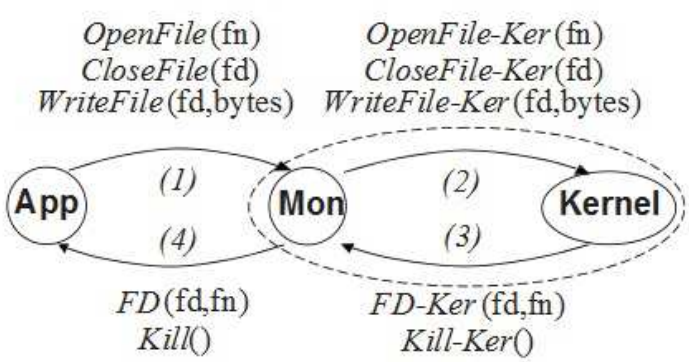
There still exists several interesting avenues for future research:

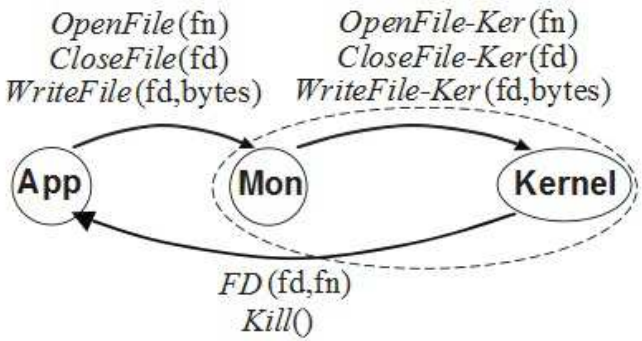
3-How can we evaluate the use of diversity and redundancy for security?



There still exists several interesting avenues for future research:

4-Can the set of enforceable properties be refined in the context of the reactive automaton model?





There still exists several interesting avenues for future research:

5-Can security properties be stated in such a way that they include a transparency requirement?

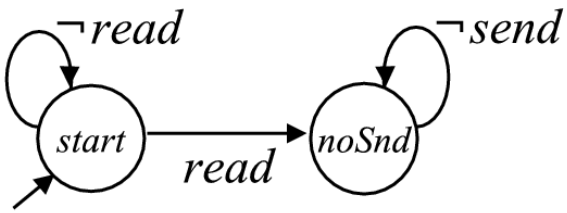


Figure 1. "No messages sent after reading a file".

Outline

1. H. Chabot R. Khoury and N. Tawbi, "Generating In-Line Monitors For Rabin Automata", 14th Nordic Conference on Secure IT Systems, Oslo, Norvège (octobre 2009), Springer's Lecture Notes In Computer Science (LNCS).
2. H. Chabot, R. Khoury and N. Tawbi, "Extending the Enforcement Power of Truncation Monitors Using Static Analysis", Computers & Security.
3. R. Khoury and N. Tawbi, "Using Equivalence Relations for Corrective Enforcement of Security Policies", 5th International Conference Mathematical Methods, Models, and Architectures for Computer Networks Security (MMM-ACNS-2010), St-Petersburg, Russia, Sept. 2010, Springer's Lecture Notes In Computer Science (LNCS).
4. R. Khoury and N. Tawbi, "A Corrective Enforcement Framework for Runtime Monitors", 7th International Workshop on Formal Aspects of Security & Trust (FAST2010) Pisa, Italy, Sept. 2010, Springer's Lecture Notes In Computer Science (LNCS).
5. R. Khoury and N. Tawbi, "Corrective Enforcement of Security Policies", ACM Transactions on Information and System Security 15(2)(2012).

1. R. Khoury and N. Tawbi, "Corrective Enforcement: A new Paradigm of Security Policy enforcement by Monitors" Submitted for publication.
2. R. Khoury and N. Tawbi, "Which Security Policies are Enforceable by Runtime Monitors? A Survey", Computer Science Review, 6(1), pp.27-45 (2012).
3. F. Lemay and R. Khoury and N.Tawbi "Optimized Inlining of Runtime Monitors", In proceedings of the 16th Nordic Conference on Secure IT Systems (NORDSEC11), Tallinn, (Estonia), October 2011, Springer's Lecture Notes In Computer Science (LNCS) series.

Thank You

Questions?