Lecture Notes:
Enforceable Security Policies

Michael Brown

January 26, 2016

1. Introduction

- Target - The system which the security policy is being enforced upon.
- Execution Monitoring (EM) - Class of policy enforcement mechanisms that works by monitoring execution of the target.
  - Example: firewalls
  - Only observes target's steps, not target's internals or alternate executions. Excludes any static analysis.
  - Does not modify target except to terminate execution when policy is violated.
- Execution steps - An event or change of state that may violate a security policy.
  - Example: method calls.

2. Characterizing EM Enforcement Mechanisms

The paper formulates 3 requirements for EMs: formulas (1), (2) and (3).

(1) requires that only one execution be observed. It defines a security policy as a predicate $\mathscr{P}$ that is satisfied if all executions satisfy the predicate $\hat{\mathscr{P}}$:

$$\mathscr{P}(\Pi) : (\forall \sigma \in \Pi : \hat{\mathscr{P}}(\sigma)) \tag{1}$$

(2) requires that $\mathscr{P}$ be prefix closed. If $\mathscr{P}$ is false for a finite sequence τ', anything with τ' as a prefix should be false as well. This is required because the EM must terminate the target as soon as the property is violated.

$$(\forall \tau' \in \Psi^{-} : \neg\hat{\mathscr{P}}(\tau') \Rightarrow (\forall \sigma \in \Psi : \neg\hat{\mathscr{P}}(\tau'\sigma))) \tag{2}$$

(3) requires that any rejection happens after a finite number of steps:

$$(\forall \sigma \in \Psi : \neg \hat{\mathscr{P}}(\sigma) \Rightarrow (\exists i : \neg \hat{\mathscr{P}}(\sigma[..i]))) \tag{3}$$

Any property satisfying (1), (2) and (3) is a safety property. Notice that (2) is basically the definition of safety. Non-safety properties cannot be enforced by EMs. The paper relates this to Lamport's formal definiton of a safety property:

$$\sigma \notin \Gamma \Rightarrow (\exists i : (\forall \tau \in \Psi : \sigma[..i]\tau \notin \Gamma)) \tag{4}$$

Example Properties:

- Access Control - This is a safety property.
- Information Flow - Not a safety property.
- Availability - Not a safety property, but Max Waiting Time (MWT) availability is a safety property.

3. Security Automata

They define Buchi automata, then define a shorthand notation for defining them in the form of guarded commands:

$$Guard \rightarrow Command$$

The guard is a condition and the command is a change of state that happens when that condition is true. See examples in figures 2 through 4. This notation can represent an automata with infinite states (e.g. Figure 3: Access Control).

4. Using Security Automata for Enforcement

The security automata allows the target to execute a step only if there is a valid transition in the security automata. This kind of enforcement makes a few assumptions:

- Bounded Memory - Using guarded command notation, we can define infinite automata, but in real life our memory is limited. In practice, security automata are not large enough for this to be a problem.
- Target Control - The EM must be able to stop the target when a property is violated.
  - Consider a Max Waiting Time requirement of 5 seconds. If the target stops producing input symbols, the EM will never terminate the target. The MWT requirement would have to be specified in execution steps.
- Enforcement Mechanism Integrity - All bets are off if the target can lie about events, or perform actions without notifying the EM.

Automaton Input Optimization

Choosing what events are sent to the EM can greatly determine how much time it takes to enforce the security policy. If the state of the automata will not change from an event, the event does not need to be sent to the EM.

Consider a memory access restriction: If every single memory access is sent to the EM for approval, it would be much slower than if only page faults were checked. The former provides a much finer grained ability of access control, but the latter is a reasonable concession.

Software-based Fault Isolation

Software-based Fault Isolation (SFI) modifies a program to make sure it will not violate some properties before it runs. Memory protection can be implemented using SFI by automatically inserting memory access checks so that no illegal accesses can be performed.

Static Analysis

If a program can be proven to uphold the desired properties, there is no need for runtime checks. Proof Carrying Code (PCC) is code that comes with its own proof that it doesn't violate some properties.

5. Discussion

Ease of Formulating Properties

Translating a policy from English to a formal property can be difficult. The difficulty of this can change with the formalism. Do you think finite automata or guarded commands are easier to use? A simple predicate like $FileRead$ can be hard to define because it may not translate directly to code.

Guidelines for Structuring Security Automata

It is best to split a complex policy into many smaller policies that can be conjoined together. This helps make the policy easier to understand and faster to execute.