Enforcement of Opacity Security Properties Using Insertion Functions

Yi-Chin Wu and Stéphane Lafortune

EECS Department, University of Michigan
CDC 12’
Dec. 13th, 2012
Motivation

• Security and privacy concerns in online services

• “Opacity”: whether the secret information of the system can be inferred by outside observers

• Example: Location-based services
Related Work

• Notions of opacity
  – [Mazaré et al. 04], [Bryans et al. 05], [Saboori et al. 07], [Bryans et al. 08]

• Verification of opacity
  – [Saboori et al. 08, 09], [Cassez et al. 09], [Lin 11]

• Enforcement of opacity
  – [Dubreil et al. 08], [Saboori et al. 08], [Cassez et al. 09]
Enforcement of Opacity Security Properties Using Insertion Functions

Yi-Chin Wu, Stéphane Lafortune

Contribution

- Enforce opacity using insertion functions

- “i-enforcing” property
- “All Insertion Structure” (AIS)
- Synthesis of i-enforcing insertion functions
Automaton Model

- $X_0$
- $L(G,X_0) := \{s \in E^*: (\exists i \in X_0) \ [f(i,s) \text{ is defined}]\}$
- $E = E_o \cup E_{uo}$
- $P(e) = e \text{ if } e \in E_o; P(e) = \varepsilon \text{ if } e \in E_{uo} \cup \{\varepsilon\}$

$E_o = \{a, b\}$
What Is The Opacity Problem?

- The system is partially observable
- The system has a secret
  - initial state, current state, sublanguage, initial-and-final state
- The intruder knows the system structure
What Is The Opacity Problem?

- The system is partially observable
- The system has a secret
  - initial state, current state, sublanguage, initial-and-final state
- The intruder knows the system structure

The secret is **opaque** if for every secret behavior, there is a nonsecret behavior that is observationally-equivalent
**Current State Opacity**

**Definition (Current-State Opacity)**

Given $G = (X,E,f,X_0)$, a set of secret states $X_S$, and a set of non-secret states $X_{NS}$, the automaton is current-state opaque if $\forall i \in X_0, \forall t \in L(G,i)$ such that $f(i,t) \in X_S$, $\exists j \in X_0, \exists t' \in L(G,j)$ such that (i) $f(j,t') \in X_{NS}$, (ii) $P(t) = P(t')$.
System G:

E₀={a,b}

Verify Current-State Opacity

opaque

Not opaque
System G:

$$E_0 = \{a, b\}$$

Verify Current-State Opacity

Not opaque

Current-State Estimator:
Opacity Enforcement Problem
How can we enforce the secret to be opaque?
Existing Opacity Enforcement Mechanisms

• Supervisory control
  – Only partial system behavior is allowed
  – [Dubreil et al. 10][Saboori et al. 12]

• Dynamic observer
  – Create new observable behavior
  – [Cassez et al. 09]
Existing Opacity Enforcement Mechanisms

• Supervisory control
  – Only partial system behavior is allowed
  – [Dubreil et al. 10][Saboori et al. 12]

• Dynamic observer
  – Create new observable behavior
  – [Cassez et al. 09]

We enforce opacity such that
• All system behavior is allowed to occur
• No new observable behavior is created
Our Approach: Insertion Functions

• A monitoring interface
• Insert extra observable events
  – Intruder cannot distinguish between inserted events and system’s observable events
I-Enforcing Property for Insertion Functions

- Admissible: allows all system’s output behavior
I-Enforcing Property for Insertion Functions

- Admissible: allows all system's output behavior
- Safe: behavior after insertion must look like existing non-secret strings
**I-Enforcing Property for Insertion Functions**

- Admissible: allows all system's output behavior
- Safe: behavior after insertion must look like existing non-secret strings

**Definition:**

- i-enforcing = admissible + safe
- i-enforceable opacity property
Is Opacity Always i-enforceable?

\[ E_0 = \{a, b\} \]
Is Opacity Always i-enforceable?

\[ E_0 = \{a, b\} \]
Is Opacity Always i-enforceable?

$E_0 = \{a, b\}$

The diagram shows nodes labeled 0, 1, and 2 with edges connecting them. Nodes 0 and 2 are marked as 'secret', while node 1 is marked as 'nonsecret'.
Is Opacity Always i-enforceable?

I-enforceability Verification Problem
Is opacity i-enforceable?
IsOpacityAlwaysi-enforceable?

I-enforceability Verification Problem
Is opacity i-enforceable?

Insertion Function Synthesis Problem
How to synthesize an i-enforcing insertion function?

E_0 = \{a, b\}

0 1 2
a b

nonsecret secret
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 1] i-Verifier

Current-State Estimator:
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 1] i-Verifier

Current-State Estimator:
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 1] i-Verifier
The All-Insertion Structure (AIS)

Enumerate all $i$-enforcing insertion functions

[Step 1] $i$-Verifier

Current-State Estimator:
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 1] i-Verifier

Current-State Estimator:
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Test 1] $L( \quad ) = P(L(G))$ ? *

• “No”: No i-enforcing insertion function exists. Stop.
• “Yes”: Go to Step 2 and complete the AIS

*Should be included in the paper
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Test 1] \( L( ) = P(L(G)) \) ? *

- “Yes”: Go to Step 2 and complete the AIS

*Should be included in the paper
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 2] Unfolded i-Verifier $V_u$

- determinization and unfolding

---

Yi-Chin Wu, Stéphane Lafortune

Enforcement of Opacity Security Properties Using Insertion Functions
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 2] Unfolded i-Verifier $V_u$
- determinization and unfolding

```
[Diagram]
```

- all deterministic, safe insertions

---

Yi-Chin Wu, Stéphane Lafortune

Enforcement of Opacity Security Properties Using Insertion Functions

15
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 2] Unfolded i-Verifier $V_u$

- determinization and unfolding

\[\text{i-verifier}\]

\[\text{all deterministic, safe insertions}\]
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 2] Unfolded i-Verifier $V_u$

- determinization and unfolding
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 2] Unfolded i-Verifier $V_u$

- determinization and unfolding

---

Yi-Chin Wu, Stéphane Lafortune

Enforcement of Opacity Security Properties Using Insertion Functions
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 2] Unfolded i-Verifier $V_u$

- determinization and unfolding
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 3] The AIS

Unfolded i-Verifier
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 3] The AIS
The All-Insertion Structure (AIS)

Enumerate all $i$-enforcing insertion functions

[Step 3] The AIS

Unfolded $i$-Verifier
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 3] The AIS
The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Step 3] The AIS

Unfolded i-Verifier

Supremal controllable sublanguage

all deterministic i-enforcing insertions
The AIS Construction Workflow

The AIS enumerates all i-enforcing insertion functions

\[
G \xrightarrow{\text{i-Verifier}} L_1 = L_2 \xrightarrow{\text{Y}} \text{Unfolded i-Verifier} \xrightarrow{\text{AIS}}
\]

- Not i-enforceable
The AIS Construction Workflow

Theorem (I-Enforceability)

An opacity property is i-enforceable iff the AIS is not the empty automaton
Synthesis of I-Enforcing Insertion Functions

Given the AIS

\[
\begin{align*}
&((bc^*)a^*)^* \\
&((bc^*)a^*)^* \\
&((bc^*)a^*)^* \\
&(ca^*b)^*c \\
&(ca^*b)^*c \\
&(ca^*b)^*c \\
&(a^* (bc^*)^* b) \\
&(a^* (bc^*)^* b) \\
&(ca^* b) \\
&(ca^* b) \\
&(ca^* b) \\
&b \\
&b \\
&b \\
&b \\
&a^* \\
&a^* \\
&(ca^* b)^* c \\
&(ca^* b)^* c \\
&(ca^* b)^* c \\
&\end{align*}
\]
Synthesis of I-Enforcing Insertion Functions

Given the AIS

• Select one insertion at every circle state

- One insertion choice
Synthesis of I-Enforcing Insertion Functions

Given the AIS

- Select one insertion at every circle state
Synthesis of I-Enforcing Insertion Functions

Given the AIS

• Select one insertion at every circle state
• Translate into the insertion automaton
Insertion Enforcement Mechanism

System $G$  Proj. $P$  System’s Output Behavior  Insertion Function  additional observable events  Modified Behavior  Intruder
Insertion Enforcement Mechanism

- A finite encoding of an insertion function
Conclusion

- A new opacity enforcement mechanism using insertion functions
- Characterization of the “i-enforceability” property
- An algorithmic procedure to check i-enforceability
- An algorithmic procedure to synthesize i-enforcing insertion functions

Future work

- Improve the algorithm for AIS construction
- Optimal insertion function
- Location-based service applications