General Framework for Opacity Supervision

Nour Elhouda SOUID & Kais Klai

{souid, kais.klai}@lipn.univ-paris13.fr

LIPN Research Lab
University Sorbonne Paris Nord

23 March 2022
Outline

- Introduction
- Background
- Proposed Approach
- Web-Service Use Case
- Developed Tool
- Conclusion & Perspectives
Motivation: Cybersecurity

- Vulnerable systems used daily
- The severity of the damages caused by recent attacks (ransomware\(^1\), Deny of Service\(^2\)).

→ In this context, formal methods appear as a reliable technique to model systems and verify their security properties⇒ information flow

- Opacity: a malicious third party is able to deduce that the system is in a secret state?

\(^1\) (e.g., TeslaCrypt in 2015, WannaCry in 2017)
\(^2\) (e.g., the MiraiKrebs, OVH DDoS in 2016)
Preliminaries: The Opacity Property

- Defined w.r.t secret predicate (a set of secret states/ runs) & an observer considered as an attacker.
- The predicate \( \phi \) is opaque if no attacker can ever conclude from its provided interface (observation) that the current run \( r \) of the system satisfies \( \phi \) (\( r \models \phi \)).
- Formal Definition: \( \forall r \in L(T) \) such that \( r \models \phi \), there exists \( r' \in L(T) \) such that \((r \sim r') \land (r' \not\models \phi)\)

Attacker observation= \{a, b\}

Opaque system!
Preliminaries: Symbolic Observation Graph (SOG)

Verifying the opacity\(\rightarrow\) State explosion problem \(\Rightarrow\) regroup states into "aggregates" \(\Rightarrow\) SOG

- Deterministic graph where each node is a set of states linked by unobservable actions and each arc is labeled with an observable action.
- Nodes of the SOG are called aggregates\(\rightarrow\) managed efficiently using decision diagram techniques
- Complexity?
- SOG opaque \(\Leftrightarrow\) NONE of its aggregates is included in the secret
A formal framework for modeling and control of Discrete Event Systems (DESs).

Objective: synthesize a supervisor → can prevent some actions from occurring to enforce security properties.

Supervisor: Partial observer ($\Sigma_m$) & controls only a subset of events ($\Sigma_c$).

The supervisor can be viewed as a function ($\gamma$) : returns a set of actions to be disabled after the observation of a trace. $\Rightarrow \gamma(tr) = \{c1, c2\}$

Permissiveness
Approach

Reinforcing the opacity of a (DES) from the SCT perspective

Suggest a novel methodology to synthesize a maximal supervisor

→ restricts the behavior without any hypothesis on the relationship between the attacker and the supervisor observations.
General Framework for Opacity Reinforcement: HSOG

Notation:
- Attacker Observation $\Sigma_a = \{ a \}$
- Supervisor Observation $\Sigma_m = \{ b \}$

- **Hyper Symbolic Observation Graph**
  - **Nodes**: [super aggregates]: sets of aggregates (not single states)
  - **Actions** in $\Sigma_m \setminus \Sigma_a$ and
  - **Arcs** are labeled with actions in $\Sigma_a$

  $\Rightarrow$ Representing state space in a condensed manner
  $\Rightarrow$ Alleviate the explosion state problem
How to obtain an HSOG?

1. Build the SOG of the system based on $\Sigma_a \cup \Sigma_m$

Notation:
- Attacker Observation $\Sigma_a$
- Supervisor Observation $\Sigma_m$
How to obtain an HSOG?

2. Consider the obtained SOG as a LTS

3. Build the corresponding SOG based on $\Sigma_a$ only.

HSOG opaque $\Leftrightarrow$ NONE of its aggregates is included in the secret

Notation:
- Attacker Observation $\Sigma_a$
- Supervisor Observation $\Sigma_m$
Approach: How it works

Step 1: On-the-fly HSOG Construction
  → Abstraction of the state space according to the attacker's observation

Step 2: Check the condition of opacity violation
  → A super-aggregate [node] is totally included in the secret?

Step 3: Enforce the opacity
  → Backtracking + disable the last controllable event
Notation:
- Attacker Observation $\Sigma_a = \{a\}$
- Supervisor Observation $\Sigma_m = \{b, c\}$
- Supervisor controls $\Sigma_c \subseteq \Sigma_m = \{c\}$
- $\Sigma \setminus (\Sigma_m \cup \Sigma_a) = \{u\}$

Supervisor: $y(u b u a u) = \{c\}$
Application to a Web Service Use Case: Scenario Description

- B2B (business-to-business) e-commerce
- Supply chain relationship between:
  - a car dealer
  - a manufacturer
  - a part supplier

Labelled Transition System representing the case study

- Attacker’s observation={access, acc_car_service, conf, dec}
- Secret states={4, 42, 43, 41, 6, 61, 918}
- Supervisor’s observation={log_supplier, acc_inventory, parts_not_found}
- Supervisor’s control={log_in, order, request1, request2, conf_request}
Application to a Web Service Use Case: Supervision

Super aggregate $\subseteq$ secret??

- Supervisor:

$\gamma(\varepsilon) = \{\text{conf_request}\}$
GoSup
General Opacity Supervision

https://depot.lipn.univ-paris13.fr/gosup/gosup

Developed Tool:
- C++ language based tool
- A tool to reinforce the opacity of DESs.
- Open source.

• Input:
  ○ The system [PNML file]
  ○ The confidential information [set of states]
  ○ The observable behaviour of the system [set of states]
  ○ The desired supervisor:
    ■ What to control
    ■ What to observe

• Output:
  ○ Supervision function → what actions to enable/disable
Conclusion

- Proposed a **GENERAL** and **REDUCED-COST** algorithm → reinforce the opacity based on a novel graph called **HSOG**.

- **ON-THE-FLY** computation of the supervisor [performed while abstracting the system].

- Prove that the obtained supervisor language K is controllable, observable, supremal, ensures the opacity.

- Use case sample: security of a B2B e-commerce application.

Why’s next?

- Quantifying the opacity property
  - Modular systems
  - More attackers
Thank you for your attention
Preliminaries: Verifying the opacity using the SOG
General Framework for Opacity Reinforcement: Approach

- Define the supervisor's behavior through a supervision function $\gamma$.
- Prove that the obtained supervisor language $K$ is
  - controllable
  - observable
  - supremal
  - ensures the opacity.
- Propose an algorithm based on an on-the-fly construction of a new version of the SOG$^1$ called Hyper Symbolic Observation Graph (HSOG)
Developed Tool: GoSup

General Opacity Supervision

https://depot.lipn.univ-paris13.fr/gosup/gosup
Application to a Web Service Use Case: SOG of the use case