Google

A Formal Foundation for Recoverability:

Defining recoverability for arbitrary security properties

Paul Crews[1], Christopher Hahn[2], Jon McCune[1], Caroline Trippel[2]
ptcrews@google.com
[1]Google, [2]Stanford University
2022-10-28

Motivation

Problem: Systems will get compromised

- Software vulnerabilities, logic bugs, side-channels
- Systems that are both security-critical and must be publicly available

Idea: Design systems that can *recover* from a compromise

- Depends on a semantic notion of security
    - A compromise depends on what guarantees the system designers wanted to provide
- Not all systems or security properties can be (easily) recovered

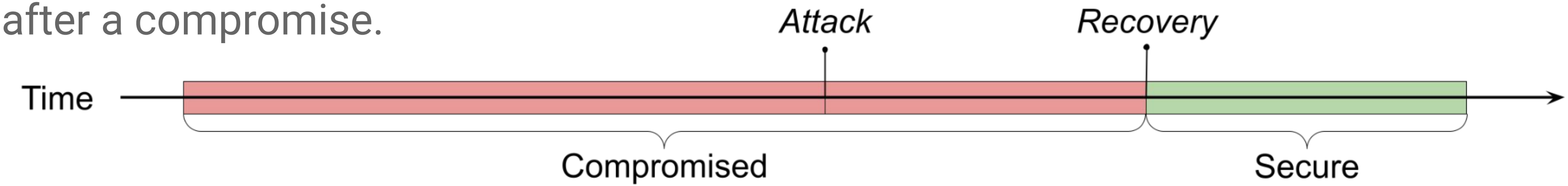
Overview

Trace properties and *hyperproperties* can define arbitrary security properties for a system.

A *compromise* is a violation of these properties, modeled as an arbitrary mutation of a trace after a point in time:

$$\texttt{MUTATE}(\pi) = \{(m, i) | i \geq 0, m = [\pi_0, \pi_1, \ldots, \pi_{i-1}, m_i, m_{i+1}, \ldots]\}$$

A property is *recoverable* if there exists a set of actions such that the property holds after a compromise.



(a): Recoverability

Recoverability (Trace Property)

A recoverable (trace) property is a property for which there exists an action such that, if this action occurs after a compromise in any trace, the compromised trace is still in the property.

$$\texttt{RECV} \equiv \{\Pi \in \texttt{Prop} | \exists r \in \Psi_{FIN}, \forall \pi \in \Pi, (m, i) \in \texttt{MUTATE}(\pi) : \\ (r \subset_E m[i+1:] \Rightarrow m \in \Pi)\}$$

Equivalently: A recoverable property is a set of traces closed under a mutation (compromise) followed by a recovery action.

Recoverability (Hyperproperty)

A recoverable hyperproperty is a hyperproperty for which there exists an action such that, if this action occurs after a compromise in any trace, the compromised trace is still in the set of traces.

$$\mathbf{RECV} \equiv \{\mathbf{\Pi} \in \mathbf{HP} | (\forall \Pi \in \mathbf{\Pi} : (\exists r \in \Psi_{FIN}, \forall \pi \in \Pi, (m, i) \in \texttt{MUTATE}(\pi) : (r \subset_E m[i+1:] \Rightarrow \Pi \setminus \pi \cup \{m\} \in \mathbf{\Pi}))))\}$$

Equivalently: A recoverable hyperproperty is a hyperproperty where each set of traces are closed under a mutation (compromise) followed by a recovery action.

Example: Guaranteed Service

Guaranteed Service: All requests have corresponding responses[1].

$$GS \equiv \{\pi \in \Psi_{INF} | (\forall k \in \mathbb{N} : isReq(\pi[k]) \Rightarrow (\exists j > k : isRespToReq(\pi[j], \pi[k])))\}$$

Recoverable Guaranteed Service: All requests after the recovery actions have corresponding responses.

$$\begin{aligned} RGS \equiv \{\pi \in \Psi_{INF} | (\forall k \in \mathbb{N}, \forall (m, i) \in \texttt{MUTATE}(\pi) : \\ (\exists r \subset_E m[i:] \land k > r_j \land isReq(m[k]) \\ \Rightarrow (\exists j \in \mathbb{N}, j > k : isRespToReq(m[j], m[k]))))\} \end{aligned}$$

[1]: Michael R Clarkson and Fred B Schneider. 2010. Hyperproperties. *Journal of Computer Security* 18, 6 (2010).

Example: Observational Determinism

Observational Determinism: If any two pairs of traces have low-equivalent starting states, then the entire traces are low-equivalent[1].

$$\mathbf{OD} \equiv \{\Pi \in \texttt{Prop} | (\forall \pi, \pi' \in \Pi : \pi[0] =_L \pi'[0] \Rightarrow \pi \approx_L \pi')\}$$

Recoverable Observational Determinism: If any two pairs of traces have low-equivalent starting states after the recovery actions, then the remainder of the traces are low-equivalent.

$$ROD \equiv \{\Pi \in \texttt{Prop} | (\forall \pi, \pi' \in \Pi : (\forall (m, i) \in \texttt{MUTATE}(\pi) : (\exists r \subset_E m[i:], r' \subset_E \pi' : (m[r_j] =_L \pi'[r'_j] \Rightarrow m[r_j :] \approx_L \pi'[r'_j :])))))\}$$

Google

[1]: Michael R Clarkson and Fred B Schneider. 2010. Hyperproperties. *Journal of Computer Security* 18, 6 (2010).

Formal Verification: Approach

In Isabelle/HOL:

- Formalize definitions:
  - Define the MUTATE operator
  - Define recoverability, bounded lookback
  - Define Guaranteed Service (GS), Recoverable Guaranteed Service (RGS)
  - Define Observational Determinism (OD), Recoverable Observational Determinism (ROD)
- Prove correctness of definitions:
  - Prove GS, OD aren't recoverable
  - Prove RGS, ROD are recoverable
- Implement a real-world system
- Prove the system is recoverable

Note: We rely on the Hyperproperties library from Bueno et. al. in this work[2].

[2]: Denis Bueno and Michael Clarkson. 2008. Hyperproperties: Verification of Proofs. https://hdl.handle.net/1813/11153

Formal Verification: Progress

In Isabelle/HOL:

- ✅ ~~Formalize definitions:~~
  - ~~Define the MUTATE operator~~
  - ~~Define recoverability, bounded lookback~~
  - ~~Define Guaranteed Service (GS), Recoverable Guaranteed Service (RGS)~~
  - ~~Define Observational Determinism (OD), Recoverable Observational Determinism (ROD)~~
- ⛔ Prove correctness of definitions:
  - Prove GS, OD aren't recoverable
  - Prove RGS, ROD are recoverable
- ⛔ Implement a real-world system
- ⛔ Prove the system is recoverable

Note: We rely on the Hyperproperties library from Bueno et. al. in this work[2].

Google

[2]: Denis Bueno and Michael Clarkson. 2008. Hyperproperties: Verification of Proofs. https://hdl.handle.net/1813/11153