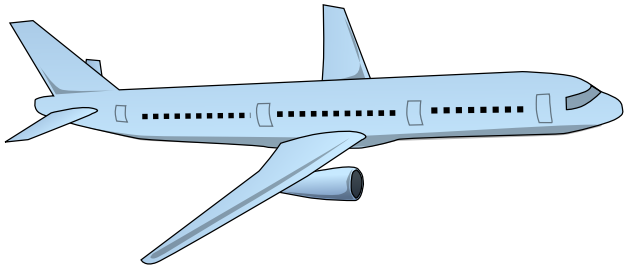
Verifying Security Policies using Host Attributes

34th IFIP International Conference on Formal Techniques for Distributed Objects, Components and Systems

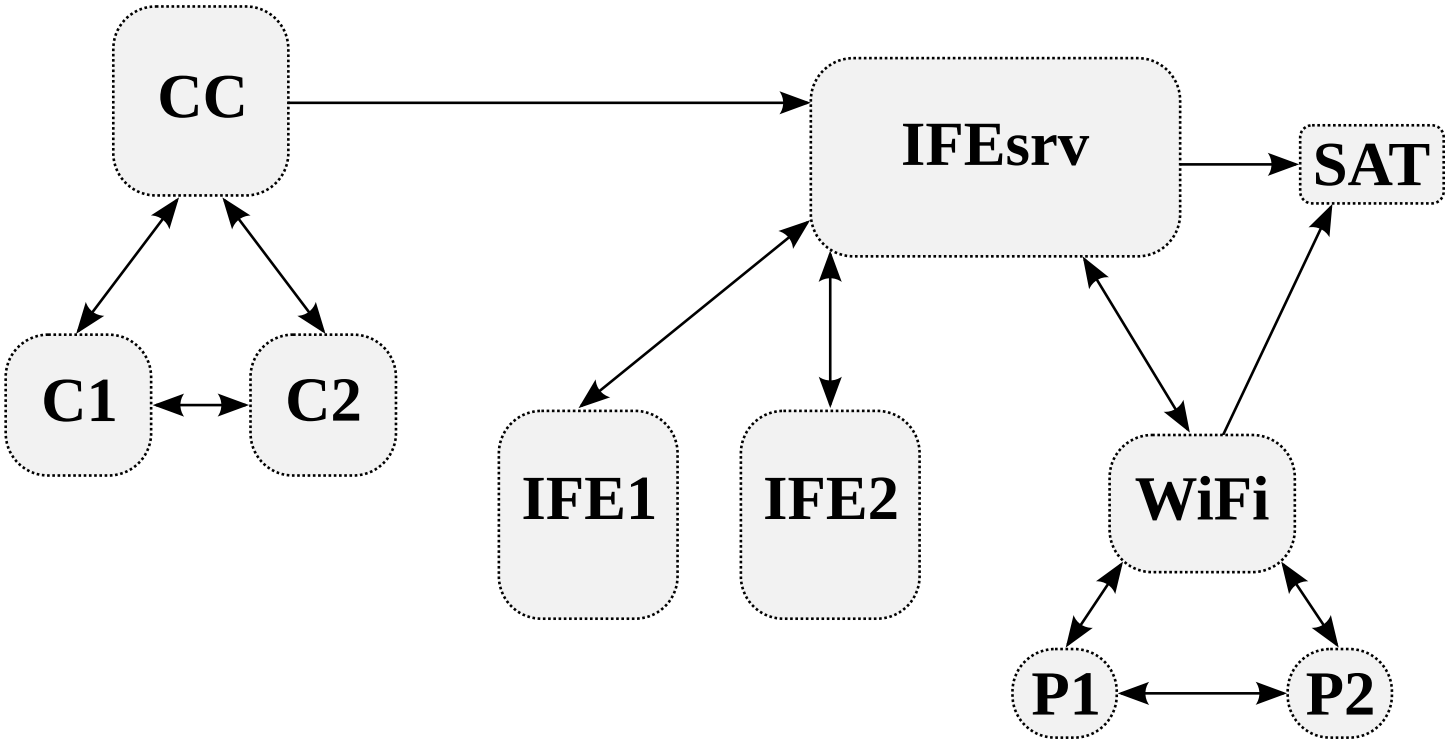
Cornelius Diekmann[1] Stephan-A. Posselt[1]
Heiko Niedermayer[1] Holger Kinkelin[1] Oliver Hanka[2]
Georg Carle[1]

[1]Technische Universität München [2]Airbus Group Innovations

Example: Aircraft Cabin Data Network

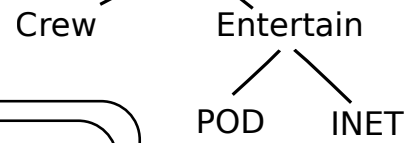


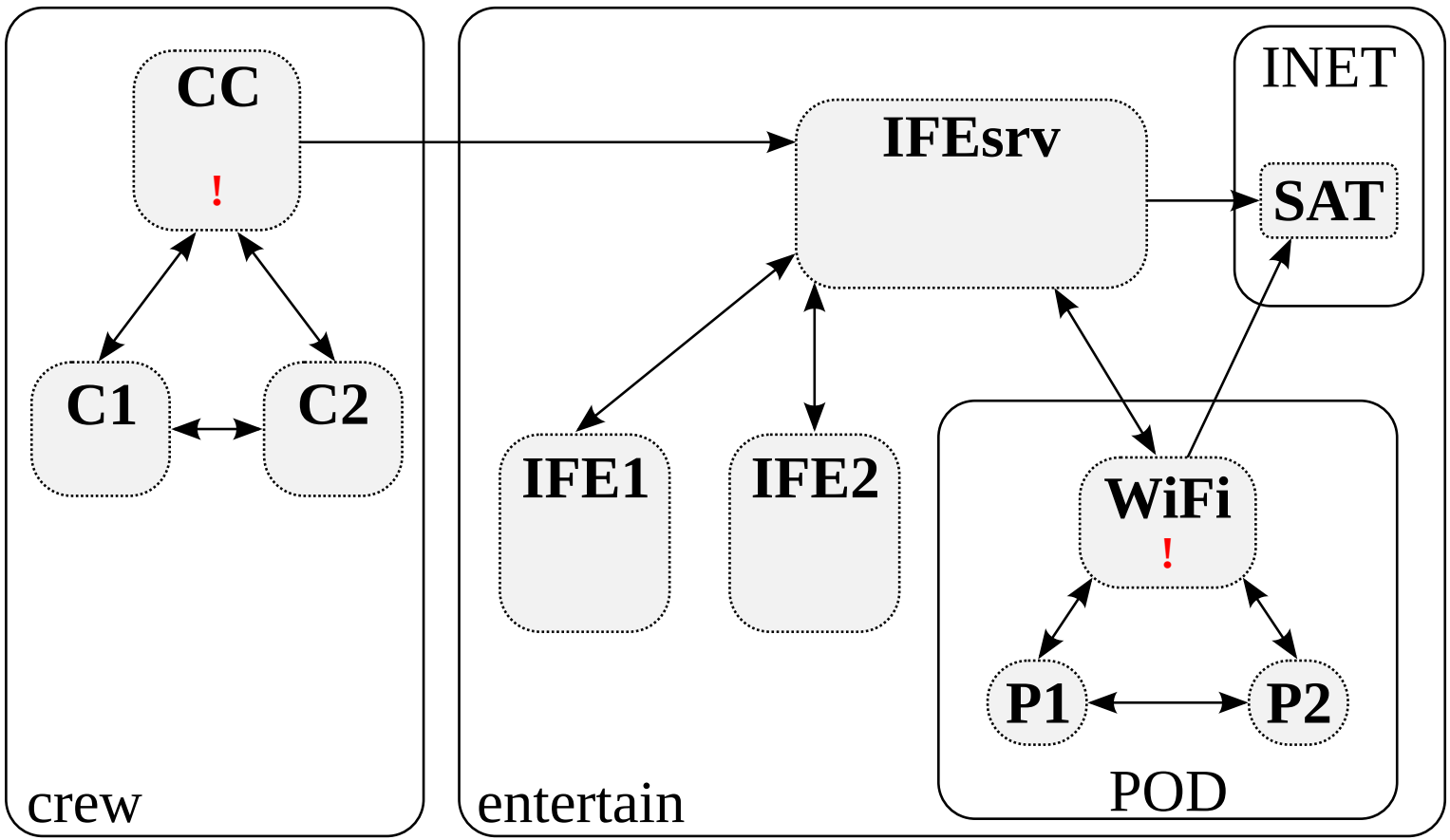
Example: Aircraft Cabin Data Network – Policy



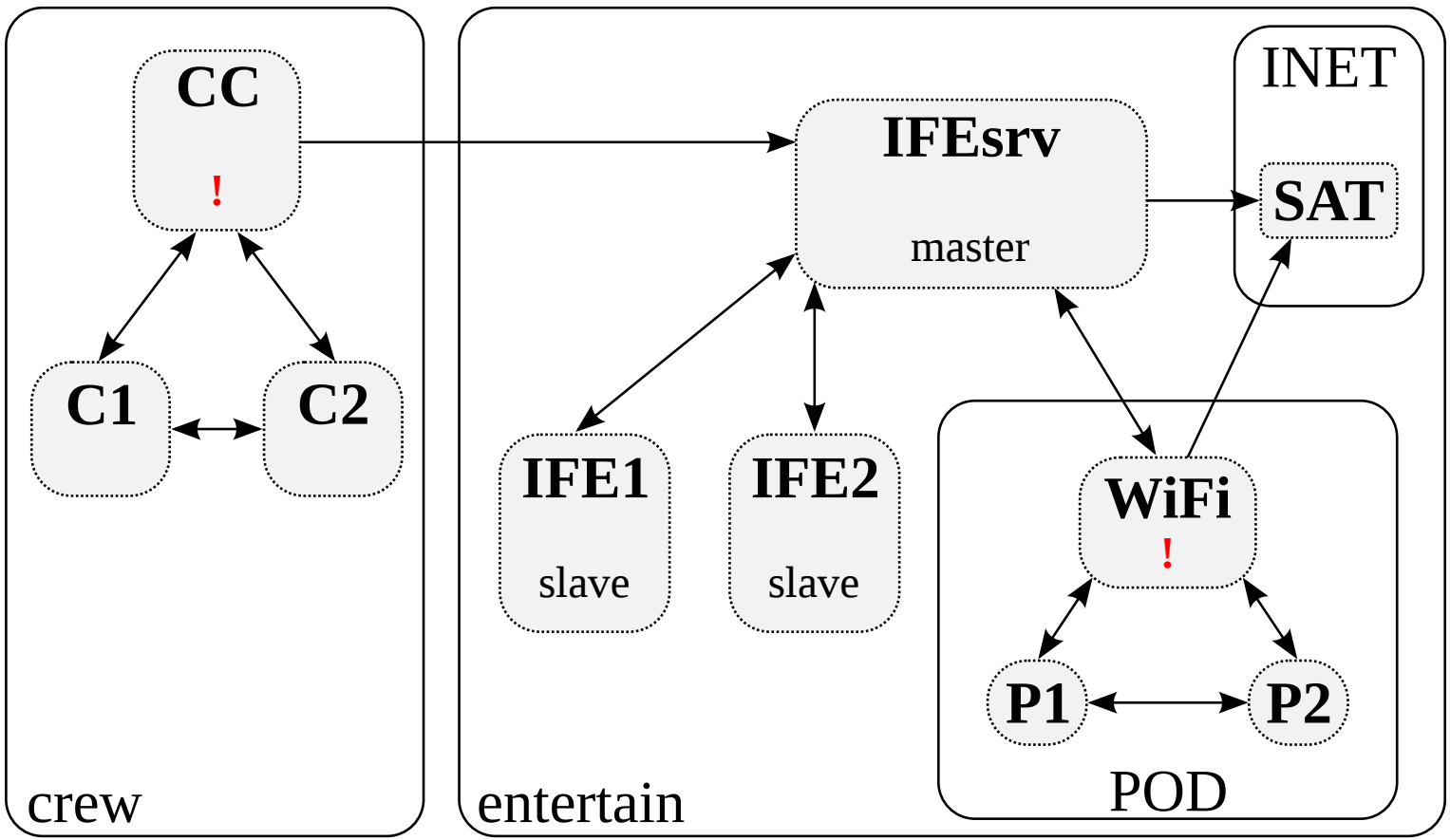
Policy $\equiv$ Access Control Matrix $\equiv$ Network Connectivity Structure

Example: Security Invariants as Host Attributes

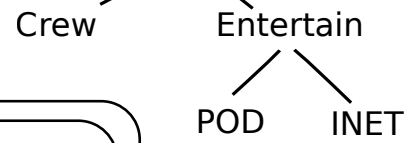


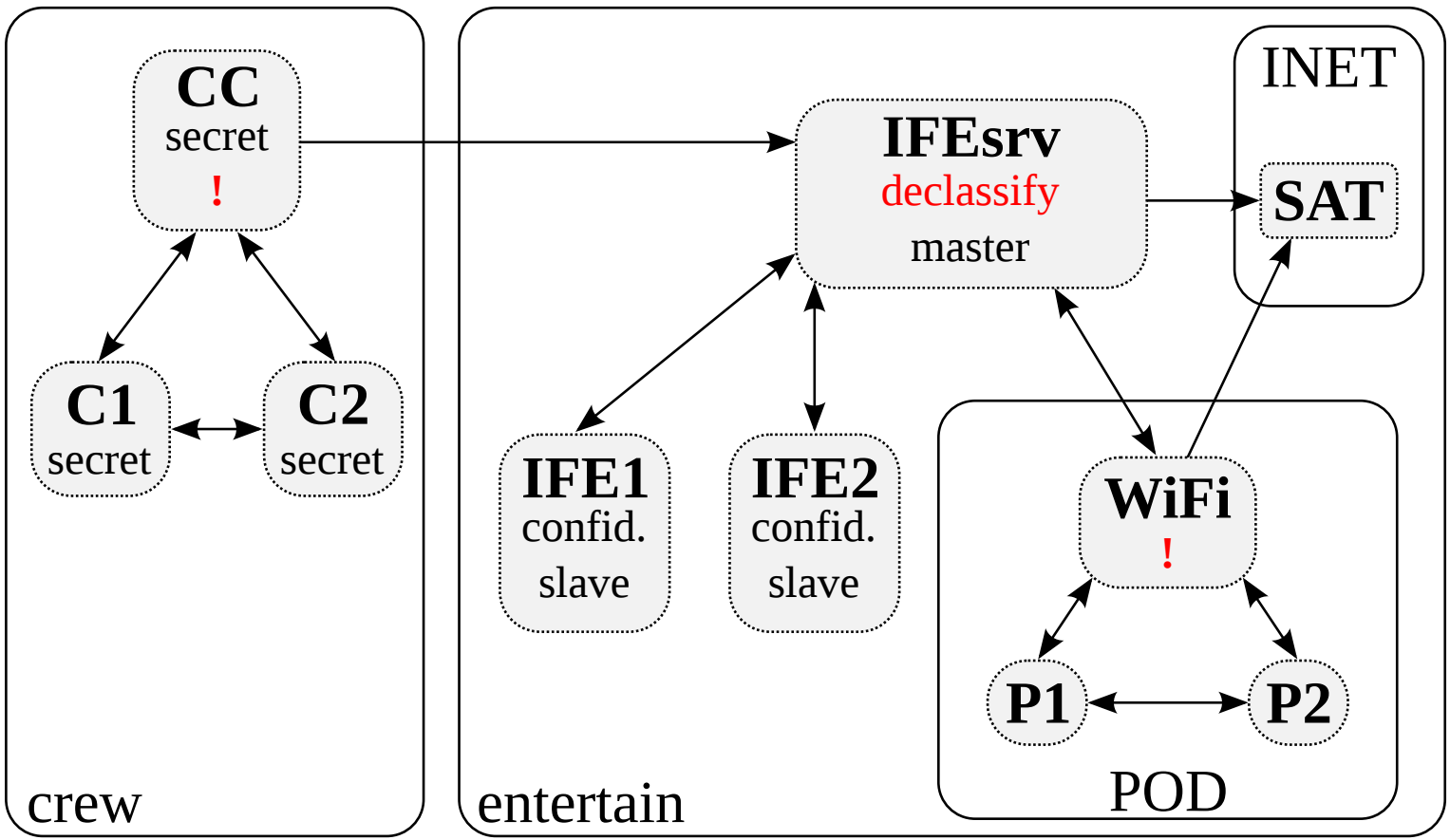


Example: Security Invariants as Host Attributes



Example: Security Invariants as Host Attributes





Big Picture: Verification

- Demo

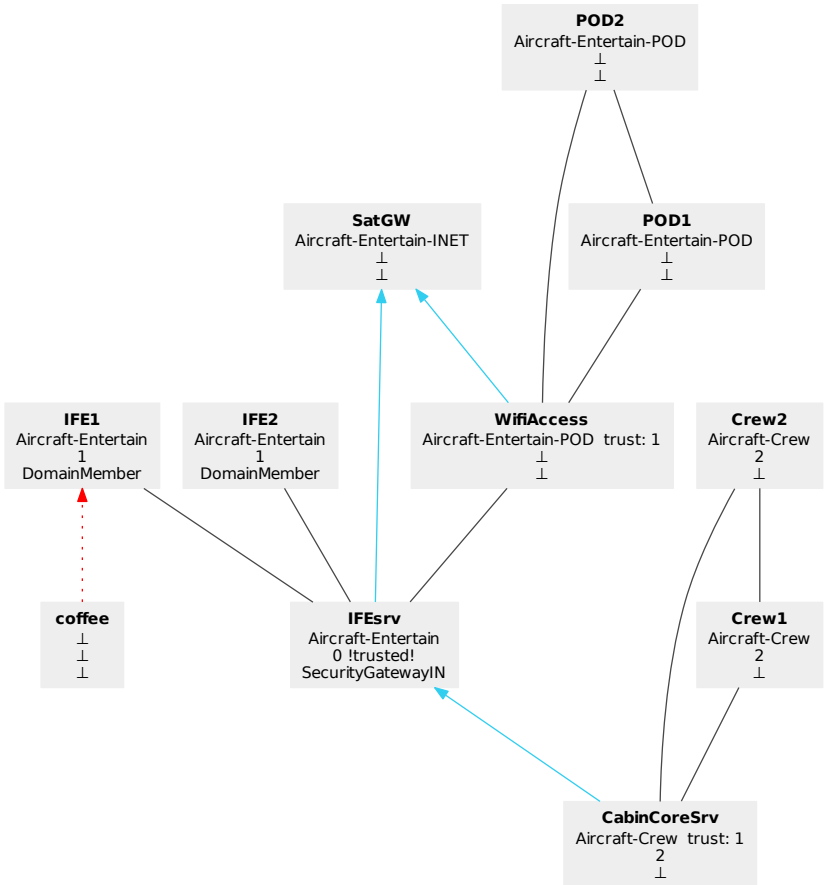
```
lemma "all_security_requirements_fulfilled security_invariants policy" by eval
```

- Formal verification
- Executable, no need for hand-crafted proofs

Big Picture: Debugging

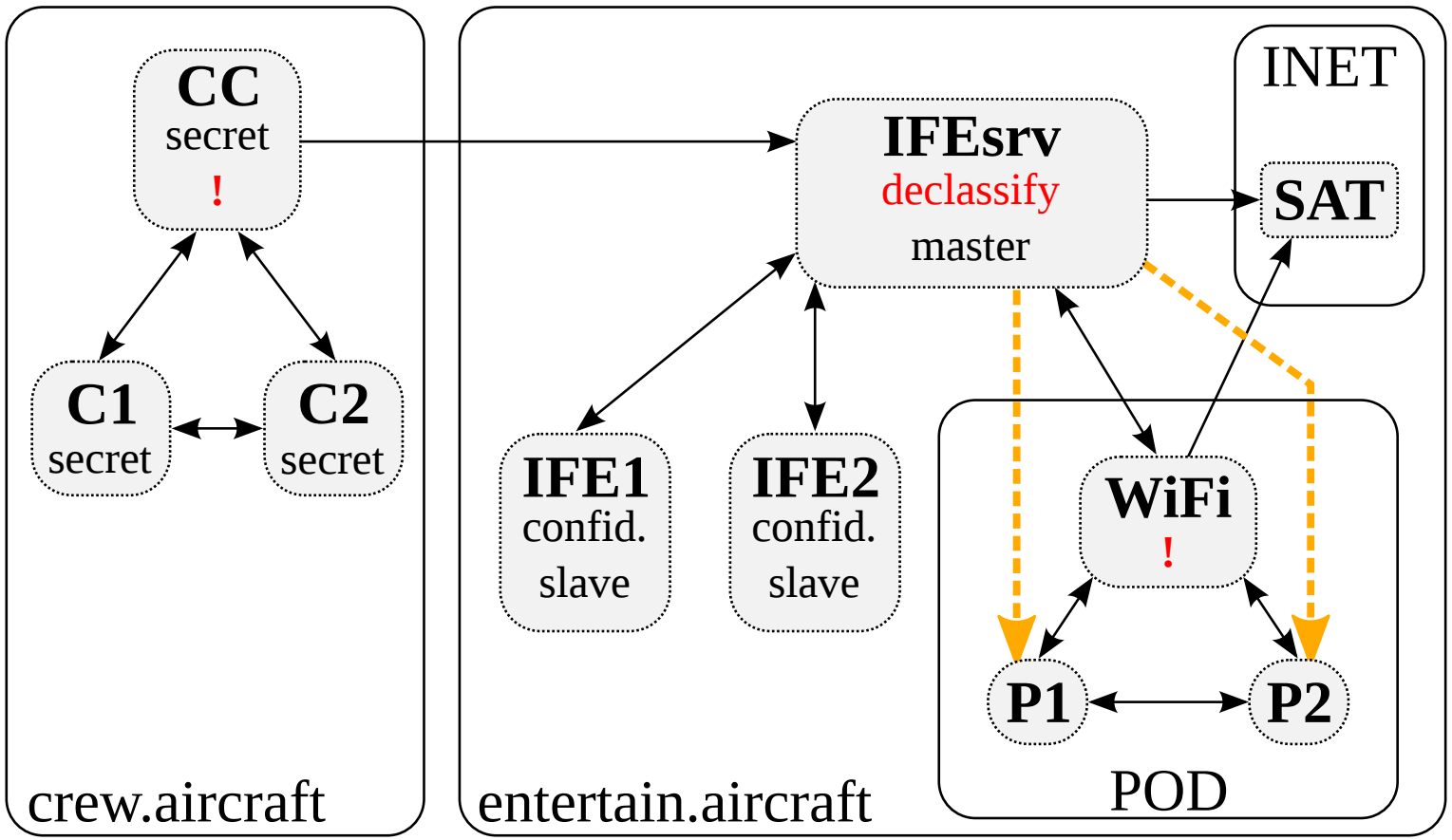
- Demo
- add coffee machine, connect it to IFE1

```
[invalid] inv1 :  ...
[invalid] inv2 :  ...
[valid] inv3 :  ...
```



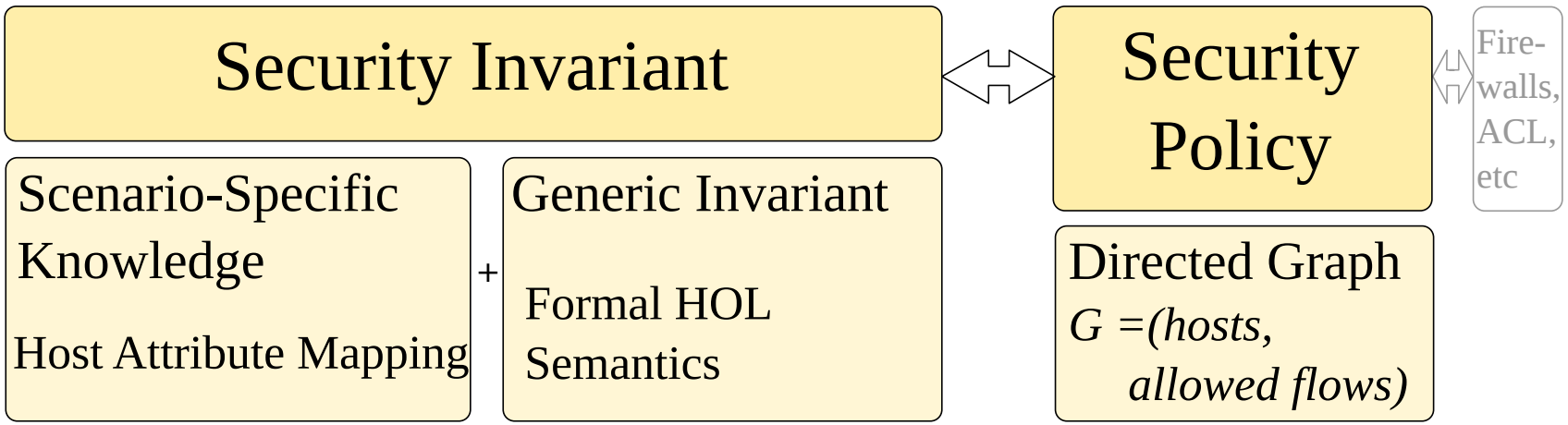
Big Picture: Invariant Feedback

- Demo



This Talk

- policy $\models$ security invariants
- security invariants $\Longrightarrow$ policy
- It's all about the security invariants!



Definitions

- Policy $G = (V, E)$
  - directed graph
  - $\mathcal{G} = (\mathcal{V}\ set) \times ((\mathcal{V} \times \mathcal{V})\ set)$
- Host mapping P: $\mathcal{V} \Rightarrow \Psi$
  - total function
  - maps a host to an attribute

Definitions

- Policy $G = (V, E)$
  - directed graph
  - $\mathcal{G} = (\mathcal{V}\ set) \times ((\mathcal{V} \times \mathcal{V})\ set)$
- Host mapping P: $\mathcal{V} \Rightarrow \Psi$
  - total function
  - maps a host to an attribute
- Security invariant template m: $m(\mathcal{G}, (\mathcal{V} \Rightarrow \Psi))$
  - Predicate (total, Boolean-valued function)
  - first argument: security policy
  - second argument: host attribute mapping
  - $m(G, P) \longleftrightarrow G$ fulfills the invariant specified by m and P

Example

- Label-based information flow security: Bell LaPadula model
- *Security clearances*, i.e. host attributes
  $\Psi = \{\mathit{unclassified}, \mathit{confidential}, \mathit{secret}, \mathit{topsecret}\}$
- Invariant template, i.e. no read-up and no write-down:
  $m\big((V, E), P\big) \equiv \forall (s, r) \in E.\ P(s) \leq P(r)$

Example

- Label-based information flow security: Bell LaPadula model
- *Security clearances*, i.e. host attributes
  $\Psi = \{\mathit{unclassified}, \mathit{confidential}, \mathit{secret}, \mathit{topsecret}\}$
- Invariant template, i.e. no read-up and no write-down:
  $m\big((V, E), P\big) \equiv \forall (s, r) \in E.\ P(s) \leq P(r)$

- Scenario-specific knowledge
  - $P(db_1) = \mathit{confidential}$
  - all other hosts are *unclassified*
- $P = (\lambda h.\ \textbf{if}\ h = db_1\ \textbf{then}\ \mathit{confidential}\ \textbf{else}\ \mathit{unclassified})$
- $m(G, P) \longleftrightarrow db_1$ does not leak confidential information
  - there is no non-reflexive outgoing edge from db_1

Security Invariants cont.

- Security strategy
  - Information Flow Strategy (IFS)
    - confidentiality
  - Access Control Strategy (ACS)
    - integrity, controlled access
  - m must be either IFS or ACS
- Monotonicity
  - Prohibiting more does not harm security
  - $m((V, E), P)$ and $E' \subseteq E$ then $m((V, E'), P)$
- Composition
  - Composability and modularity enabled by design
  - $m_1(G, P_1) \wedge \cdots \wedge m_k(G, P_k)$

Offending Flows

- If a security invariant is violated $\neg\, m(G, P)$
- Flows $F \subseteq E$ responsible for the violation
- $set\_offending\_flows\big(G, P\big)$ is the set of all such minimal F

Offending Flows

- If a security invariant is violated $\neg\, m(G, P)$
- Flows $F \subseteq E$ responsible for the violation
- $set\_offending\_flows(G, P)$ is the set of all such minimal F

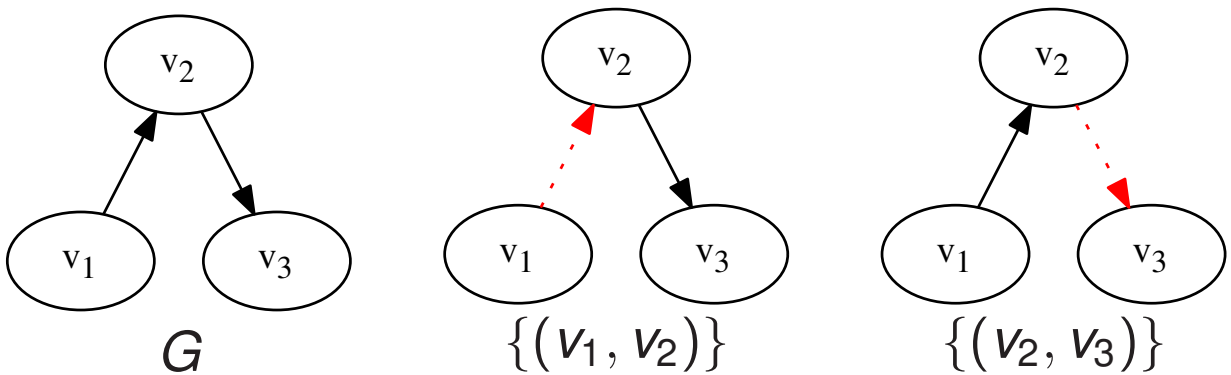
- i.e. the set of all minimal sets which violate m

$$
\begin{aligned}
&set\_offending\_flows(G, P) = \\
&\Big\{ F \subseteq E \mid \neg\, m(G, P) \ \wedge\ m((V, E \setminus F), P) \wedge \\
&\forall (s, r) \in F.\ \neg\, m((V, (E \setminus F) \cup \{(s, r)\}), P) \Big\}
\end{aligned}
$$

Offending Flows

- If a security invariant is violated $\neg\, m(G, P)$
- Flows $F \subseteq E$ responsible for the violation
- $\mathit{set\_offending\_flows}(G, P)$ is the set of all such minimal F

- Example: m is that v_1 must not access v_3 transitively



- $\mathit{set\_offending\_flows}(G, P) = \{\{(v_1, v_2)\}, \{(v_2, v_3)\}\}$

Example: Offending Flows for the Bell LaPadula Model

- Recall: $m((V, E), P) \equiv \forall (s, r) \in E.\ P(s) \leq P(r)$

$$set\_offending\_flows(G, P) = \begin{cases} \{\{(s, r) \in E \mid P(s) > P(r)\}\} & \textbf{if } \neg\, m(G, P) \\ \emptyset & \textbf{if } m(G, P) \end{cases}$$

- Uniquely defined
- Can be computed in linear time
- This holds for all similar-structured security invariants

Analysis: Offending Flows

- Do the offending flows always exist?

Theorem 1. *For m monotonic, let* $G_{\textit{deny-all}} = (V, \emptyset)$. *If* $\neg\, m(G, P)$ *then*

$$m\big(G_{\textit{deny-all}},\ P\big) \longleftrightarrow \textit{set\_offending\_flows}(G,\ P) \neq \emptyset$$

- A security violation can be fixed by tightening the policy *iff* the invariant holds for the deny-all policy

Example: Policy Construction

- $G_{all} = (V,\ V \times V)$
- $G_{max} = (V,\ V \times V \setminus \bigcup set\_offending\_flows(G_{all},\ P))$
- Iterate this over all m_i
- Sound for arbitrary m
- Complete for m similar to Bell LaPadula
  - G_{max} is the uniquely defined maximum policy

Who is responsible for a violation?

- A violation either happens at the sender or the receiver

ACS initiator provokes an access control restriction

IFS leak only occurs when the information reaches the unintended receiver

Definition. *For* $F \in set\_offending\_flows(G, P)$

$$offenders(F) = \begin{cases} \{ s \mid (s, r) \in F \} & \textbf{if } ACS \\ \{ r \mid (s, r) \in F \} & \textbf{if } IFS \end{cases}$$

Auto Completion of Host Mappings

- P is a *total* function $\mathcal{V} \Rightarrow \Psi$
- User-specified *incomplete* host mapping: $P_C \subseteq \mathcal{V} \times \Psi$
- Default value: $\bot \in \Psi$
- $P(v) = \begin{cases} \psi & \textbf{if } (v, \psi) \in P_C \\ \bot & \textbf{else} \end{cases}$

Auto Completion of Host Mappings

- P is a *total* function $\mathcal{V} \Rightarrow \Psi$
- User-specified *incomplete* host mapping: $P_C \subseteq \mathcal{V} \times \Psi$
- Default value: $\perp \in \Psi$
- $$P(v) = \begin{cases} \psi & \textbf{if } (v, \psi) \in P_C \\ \perp & \textbf{else} \end{cases}$$
- What is a *secure* $\perp$?
- Everything forbidden? Barely usable!
- $\perp$ can never solve an existing security violation
- $\perp$ must never mask potential security risks!

Secure Auto Completion of Host Mappings

- Replacing the host attribute of any offenders by $\bot$ must guarantee that no security-relevant information is masked.

Definition. *$\bot$ is a secure default attribute if*

$$\forall\, G\ P.\ \forall\, F \in \mathit{set\_offending\_flows}(G, P).$$
$$\forall v \in \mathit{offenders}(F).\ \neg\, m(G, P_{v \mapsto \bot})$$

- *Secure* and *permissive*
  - $\bot = \mathrm{everything\ forbidden}$, security proof easy
  - $\bot$ can be more permissive, e.g., Bell LaPadula
    - $\bot =$ *unclassified* hosts can freely interact
    - $P_C = (db_1, \mathit{confidential})$
    - Security invariant for *complete* network
    - restrictions *only* for interactions with db_1

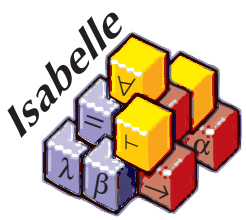
Example

- $m(G, P)$
- New host x (P is not updated)
- $P(x) = \bot$
- Magic oracle $P_{oracle}(x) = \psi$
- x is an attacker
- $\neg\ m(G, P_{oracle})$
- Is the security violation exposed without the oracle?
- If $\neg\ m(G, P_{oracle})$ then $\neg\ m(G, P_{x \mapsto \bot})$
- Hence $\neg\ m(G, P)$
- The security violation is never masked!

Conclusion

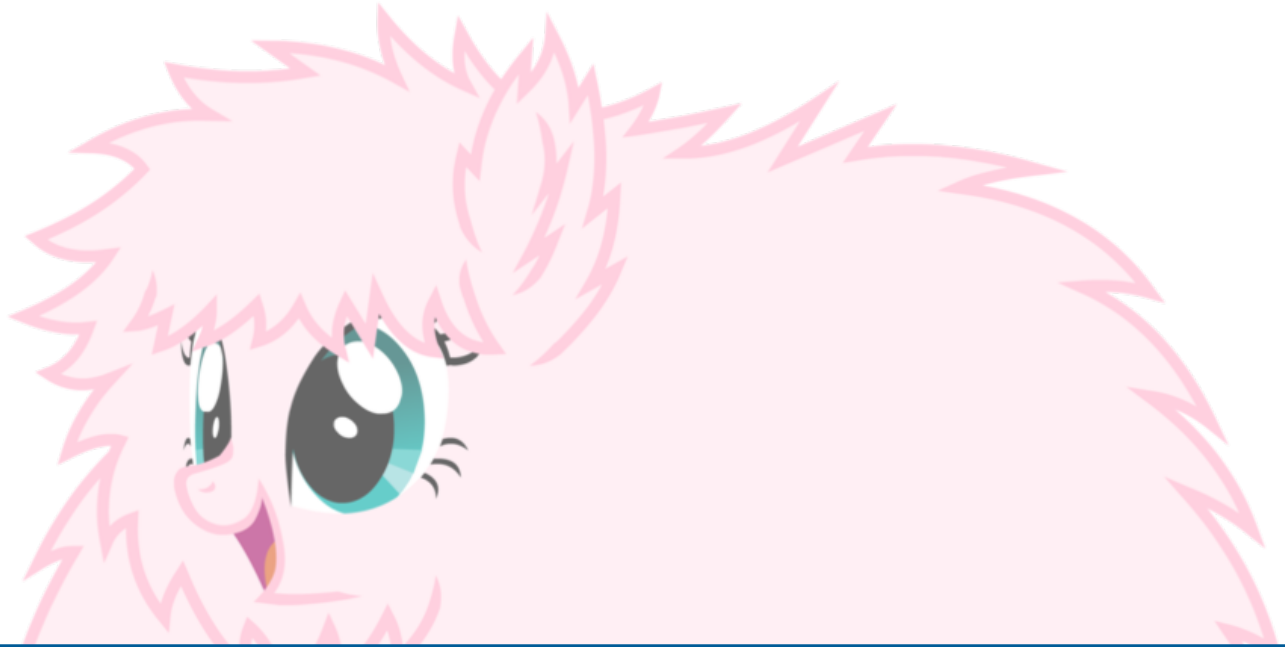
- Monotonic Security Invariants
  - Verify policy ✓
  - Fix security violations ✓
  - Construct policy ✓
  - Construct easily and end-user-friendly ✓
- Fully machine-verified with Isabelle/HOL ✓

`https://github.com/diekmann/topoS`



Thanks for your attention!

Questions?



Backup Slides

Example: Aircraft Cabin Data Network

CC The Cabin Core Server (essential aircraft features).
- air conditioning, crew telecommunication ...

C1, C2 Two crew mobile devices.
- identify passenger calls, make announcements ...

IFEsrv The In-Flight Entertainment server.
- movies, Internet ...

IFE1, IFE2 Two In-Flight entertainment displays (thin client).
- Mounted at the back of seats
- Movies and Internet ...

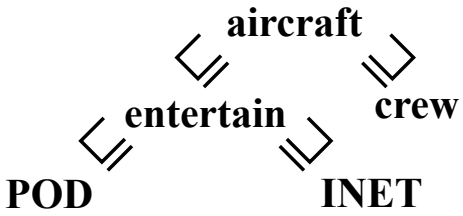
Wifi A wifi hotspot.
- Internet access for passengers owned devices

Sat A satellite uplink to the Internet.

P1, P2 Two passenger owned devices.
- laptops, smartphones ...

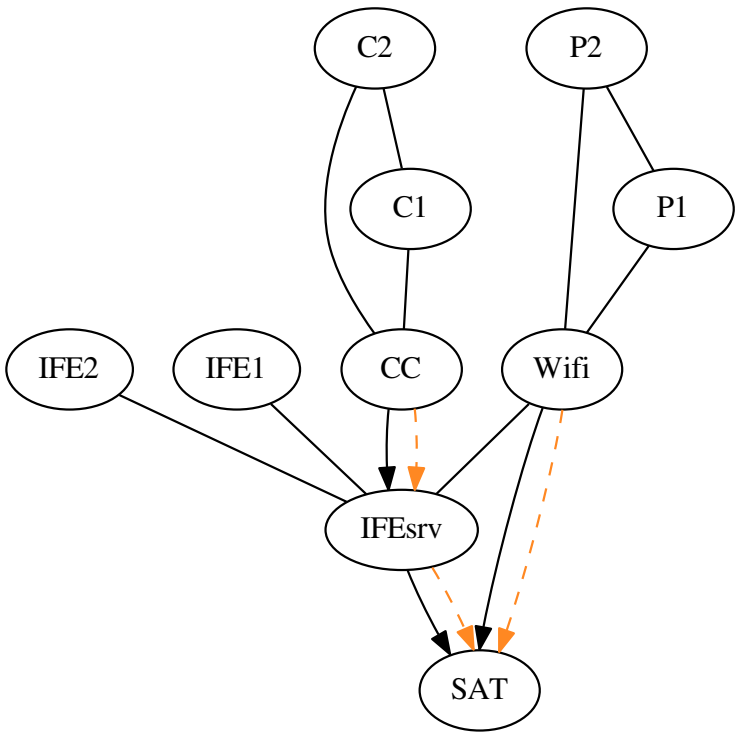
Example: Security Invariants

- Invariant 1: Domain Hierarchy
  - Different protection domains
  - Devices may send to their own (sub-)domains
  - Trust level are available



- Invariant 2: Security Gateway
  - Slave devices are bound to a master
  - Master controls all communication to/between them.
  - IFE displays (thin clients) are strictly bound to IFEsrv
- Invariant 3: Bell LaPadula
  - Information flow restrictions
  - *secret* crew communication
  - *confidential* ($<$ *secret*) passenger communication (privacy)

Stateful Implementation



Cornelius Diekmann, Lars Hupel, and Georg Carle. *Directed Security Policies: A Stateful Network Implementation*. In ESSS, Singapore, May 2014.

Bell LaPadula with Trust

- Prevent information leakage
- $P(v).sc$ is v's security clearance
- $P(v).trust$

$$m\big((V, E), P\big) \equiv \forall (s, r) \in E. \begin{cases} True & \textbf{if } P(r).trust \\ P(s).sc \leq P(r).sc & \textbf{otherwise} \end{cases}$$

Domain Hierarchy

- Hierarchical structures
- Ψ = fully qualified domain names
  - $\{a.b.c,\ b.c,\ c,\ ...\}$
- '$\sqsubseteq$' $\equiv$ '*is below or at the same hierarchy level*'
  - $a.b.c \sqsubseteq a.b.c$
  - $a.b.c \sqsubseteq b.c$
  - $a.b.c \sqsubseteq c$
- Trust: Act as if were in higher hierarchy position
  - Trust 1 = one position higher
  - $chop(a.b.c,\ 1) = a.c$

$$m((V, E),\ P) \equiv \forall (s, r) \in E.\ P(r).level \sqsubseteq chop(P(s).level,\ P(s).trust)$$

Security Gateway

- Approve intra-domain communication between domain members by a central instance (security gateway)

$$m\big((V, E),\ P\big) \ \equiv\ \forall (s, r) \in E,\ s \neq r.\ \texttt{table}\big(P(s),\ P(r)\big)$$

snd	rcv	rslt	explanation
sgw	*	✓	Can send to the world.
sgwa	*	✓	— "—
memb	*sgw*	✓	Can contact its security gateway.
memb	*sgwa*	✓	— "—
memb	*memb*	✗	Must not communicate directly. May communicate via $sgw(a)$.
memb	*default*	✓	No restrictions for direct access to outside world. Outgoing accesses are not within the invariant's scope.
default	*sgw*	✗	Not accessible from outside.
default	*sgwa*	✓	Accessible from outside.
default	*memb*	✗	Protected from outside world.
default	*default*	✓	No restrictions.