Robust Declassification

Steve Zdancewic
Andrew Myers
CSFW ’01

CS711, 12 Nov 03
Stephen Chong

Declassification

- Real systems intentionally leak (declassify) confidential information
  - Purchase of information
  - Aggregated data
  - Encryption
  - Security protocols
    - Commit-reveal, challenge-verify, ...
  - E.g. Password checker

Password Example

```c
// passwd is the password
// h is secret and shouldn’t be revealed
// guess is the user’s guess
// t is time (0 before guess checked; 1 after)
// r is the result (1 if passwd == guess)

if (passwd == guess) {
  r := 1;
} else {
  r := 0;
}
t := t + 1;
```

Life in a World Without Noninterference

- What useful info flow security properties can we describe that permit declassification?
  - Statically check authority (as in JIf, PKI)
  - Intransitive noninterference
  - Quantify information declassified
  - Robust declassification
  - Other notions?

- Still trying to define suitable security properties
  - Proving/guaranteeing properties another issue

Password Example

```c
// passwd is the password
// h is secret and shouldn’t be revealed
// guess is the user’s guess
// t is time (0 before guess checked; 1 after)
// r is the result (1 if passwd == guess)

if (declassify(passwd == guess)) {
  r := 1;
} else {
  r := 0;
}
t := t + 1;
```

Program does not satisfy noninterference!
In general, declassification violates noninterference.

Robust Declassification: Definitions

- **System** $S=(\Sigma, \alpha)$
  - $\Sigma$: set of states
  - $\alpha \subseteq \Sigma \times \Sigma$: transition relation

- **Trace** $\tau$
  - A finite sequence of states
  - $\sigma_0, \sigma_1, \ldots, \sigma_i$
  - Equivalent up to stuttering
    - $\sigma_0, \ldots, \sigma_i \approx \sigma_0, \ldots, \sigma_i$

- **View** $\approx$
  - An equivalence relation on $\Sigma$
  - What observations can be made on a state
  - E.g. Low-equivalence: low security locations can be observed, high security locations cannot.
### Password example

\[
\{p, h, g, r, t\}
\]

- passwd
- secret
- guess
- result
- time

---

### Lattice of Views

- \( I(\Sigma) \): the set of all views of the system
- Forms a lattice:
  - \( \approx \)
  - \( \forall \sigma, \sigma' \in I \approx \sigma \sigma' \approx \sigma', \sigma' \sigma \approx \sigma' \approx \sigma \)

- Example
  - Consider states with 2 locations, each location having value 0 or 1.
  - 4 possible states: 00, 01, 10, 11

---

### Password Observations

#### Observations of \( S \) wrt \( \sigma_0 \) and \( \approx \)

- All sequences of equivalence classes of traces of \( S \) starting from \( \sigma_0 \)
- \( \text{Obs}(S, \approx, \sigma_0) = \{[\sigma_0], [\sigma_1], \ldots | \sigma_0, \sigma_1, \ldots \sigma_n \text{ is a trace of } S\} \)

#### Traces

- \( \sigma: (0, 0, 0, 0, 0, 0, 1, 1, 1) \)
- \( \tau: (0, 0, 0, 0, 1, 0, 0, 1, 1) \)
- \( \nu: (0, 0, 0, 0, 1, 0, 0, 0, 0) \)
- \( \eta: (0, 0, 0, 0, 1, 0, 0, 0, 0, 0) \)
- \( \ldots \)

#### Obs(\( \Sigma, S, \approx, (0, 0, 0, 0, 0, 0) \)) = \{\)

- \( (*, *, 0, 0, 0, 0, 0, 0, 0, 0, 0) \)
- \( (*, *, 0, 0, 0, 0, 0, 0, 0, 0, 0) \)
- \( (*, *, 0, 0, 0, 0, 0, 0, 0, 0, 0) \)
- \( (*, *, 0, 0, 0, 0, 0, 0, 0, 0, 0) \)
- \( \ldots \)
Observational Equivalence

- $\text{Obs}(S, \approx) : \Sigma \rightarrow \text{Observations}$ induces another equivalence relation $\Sigma \approx$
  - $(\sigma, \sigma') \in \Sigma \approx$
    $\iff$ $\text{Obs}(S, \approx, \sigma) = \text{Obs}(S, \approx, \sigma')$
    $\iff$ "$\sigma, \sigma'$ are observationally equivalent"

- Password example
  - $(0, 0, 0, 1, 0)$ and $(0, 1, 0, 1, 0)$ are obs. equivalent
  - $(0, 0, 0, 1, 0)$ and $(1, 0, 0, 1, 0)$ are not obs. Equivalent

- $\Sigma \approx$ gives at least as much info as $\approx$
  - I.e. $|\Sigma \approx| \leq |\approx|

$\approx$-Secure System

- $S$ is $\approx$-secure
  - iff "all $\approx$-equiv. states are obs. equiv."
  - iff $\forall \sigma, \sigma' : \sigma \approx \sigma' \implies (\sigma, \sigma') \in \Sigma \approx$
  - iff $\Sigma \approx \models \approx$
  - iff $\Sigma \approx \models \approx$

- Intuition: a passive attacker with view $\approx$ cannot learn anything new about the initial state by watching the system execute.
  - Essentially noninterference
  - Initial state contains all "important" information

A Limit to Information

- Recall: $\Sigma \approx$ is an equivalence relation on $\Sigma$, with $\approx \models \Sigma \approx$
  - $\Sigma \approx$ is
  - $\Sigma \approx = \approx$
  - $\Sigma \approx \models \Sigma \approx$
  - $\Sigma \approx \models \approx$

- Intuition: $\Sigma \approx$ is the lowest view that can see all of the information that $\Sigma$ will declassify
  - For any system $\Sigma$ and view $\approx$, $\Sigma$ is $\approx\Sigma$-secure

Active Attackers

- Assume we have an attacker with view $\approx A$, and a system $\Sigma$ that intentionally declassifies information
  - $\Sigma$ is not $\approx A$-secure

- Could an active attacker make $\Sigma$ reveal more information than $\Sigma$ meant to?
  - I.e. laundering attacks

- Active attackers
  - Can add transitions $\alpha_{\text{Att}}$ to $\Sigma$
    - i.e. $(\Sigma, \alpha_{\text{Att}})$
  - "Fairness": $\alpha_{\text{Att}}$ is limited to transitions that don’t themselves declassify data, i.e. must be laundering attacks.
    - An $\alpha_{\text{Att}}$-attack is a system $\mathcal{A} = (\Sigma, \alpha_{\text{Att}})$ such that $\mathcal{A}$ is $\approx A$-secure.
      - Write $\mathcal{A} \cup S$ for $(\Sigma, \alpha_{\text{Att}} \cup \alpha)$

- What sort of attacks does this correspond to?
  - Attacker injecting code in the system that satisfies noninterference
    - Randomly flipping bits in the machine, e.g. passing a magnet over it

Robustness (at last)

- A system $\Sigma=(\Sigma, \alpha)$ is robust with respect to a class $\mathcal{B}$ of $\approx A$-attacks if
  - $\forall \mathcal{A} = (\Sigma, \alpha_{\text{Att}}) \in \mathcal{B}$. $(\mathcal{A} \cup S) \models \approx A \models \approx$

- Intuition: Watching the attacked system reveals no more information than watching the original system
Attacking the Password Program

- Add attack transitions:
  \[(p, h, g, r, 0) \xrightarrow{\text{Att}} (h, h, g, r, 0)\]
- Note: \(\text{Att} = (\Sigma, \alpha_{\text{Att}})\) is \(\approx\)-secure
- Password program is not robust against \(\text{Att}\), since
  \[((0, 1, 0, 0, 0), (0, 0, 0, 0, 0)) \notin (S \cup \text{Att})|_{\approx_L}\]
  but
  \[((0, 1, 0, 0, 0), (0, 0, 0, 0, 0)) \in S|_{\approx_L}\]
  i.e. \((S \cup \text{Att})|_{\approx_L} \nsupseteq S|_{\approx_L}\)

≈₄-security and Robustness

- If \(S\) is \(\approx₄\)-secure, then \(S\) is robust to all \(\approx₄\)-attacks
  i.e. If a system doesn’t do any declassification, an attacker cannot launder any data.

Dude, Where’s my Language?

- Use language-level constructs/analysis to rule out attacks that the system would not be robust against
  - High integrity for the data to declassify
  - High integrity for the decision to declassify
  - But...
    - Vulnerable to attacks outside language abstraction
    - What is the interaction with \(\text{endorse}\), the dual of \(\text{declassify}\)?

Language level attacks

- High integrity for data to declassify
  
  ```java
  passwd = h;
  if (declassify(passwd == guess)) {
    r := 1;
  } else {
    r := 0;
  }
  t := t + 1;
  ```

Language level attacks

- High integrity for decision to declassify
  
  ```java
  int revealAliceBid() {
    return declassify(aliceBid);
  }
  ...
  aliceBid = ...;
  ...
  bobBid = ...;
  ...
  if (revealAliceBid() > revealBobBid()) {
    // Alice wins
  }
  ```
Language level attacks

- High integrity for decision to declassify

```java
int revealAliceBid() {
    return declassify(aliceBid);
}
...
aliceBid = ...;
...
bobBid = revealAliceBid() + 1;
...
if (revealAliceBid() > revealBobBid()) {
    // Alice wins
}
```

Summary and Discussion Points

- Definition of view equivalence of system traces
  - Lattice of views
    - More general than security lattices
    - Useful?
- Definition of a couple of useful security properties
  - ≈-secure
    - For passive attackers
    - Like noninterference
  - Robustness
    - Active attackers
- What else would we like?
  - Language setting?
    - Ongoing work
    - Endorse: dual of declassify, yet different...
  - Given a system  S , what is the lowest view  ≈S such that  S is robust to all ≈-attacks?