Belief in Information Flow

Michael Clarkson
Cornell University

Joint work with Andrew Myers and Fred B. Schneider
Newcastle University
July 3, 2006

Password Checker

PWC: if $p = u$ then $a := 1$ else $a := 0$

- $p$: stored password
- $u$: user-entered password
- $a$: authentication flag

Is this program secure?

Guards

- Inspect information
- Permit, deny, transform
- Implement security policy
Quantitative Security Policies

- Examples:
  - Expected rate of flow is at most k bits per second
  - At most k bits leak in any execution
- This work: A model for reasoning about whether programs satisfy quantitative information flow (QIF) policies

Information Flow Policies

- Control release and propagation of information
  - Information flows = dependencies
- Type of policies:
  - Confidentiality (secrecy, privacy)
    - Information not leaked by computation
  - Integrity
    - Information not damaged by computation
  - Availability
    - Information is obtainable by/from computation
- End-to-end security

Access Control Policies

- Control release but not propagation

```bash
$ ls -l ~/.ssh/id_rsa
-r------- ... id_rsa
$ cat ~/.ssh/id_rsa
----- BEGIN RSA PRIVATE KEY-----
...
```

- Weaker than information flow policies

Noninterference

“Low-security behavior of the program is not affected by any high-security data”

[Goguen & Meseguer 82]

Confidentiality: H = secret, L = public
Integrity: H = untrusted, L = trusted

Qualitative vs. Quantitative

- Noninterference (qualitative confidentiality) is too strong for PWC

```plaintext
PWC: if p then a := 1 else a := 0
```

- Previous work is almost all qualitative
  - [Language-based survey: Sabelleli & Myers 03]

- Need quantitative model for PWC intuition (“leaks little information”)

Traditional Model for QIF

```
S

H_in

L_in

H_out

L_out
```

Information flows when uncertainty is decreased

- Flow = Uncertainty(H_in) - Uncertainty(H_out, L_out)
- Uncertainty measured with some variant of entropy
  - [Denning 82, McIver and Morgan 03, Clark, Hunt, and Meleardi 05]
Adding Beliefs to Model

- Model attacker’s uncertainty about \( H \) inputs as a probability distribution
- We call this distribution a belief

Analyzing PWC

1. **Prebelief**
   - \( p = A: 0.98 \)
   - \( B: 0.01 \)
   - \( C: 0.01 \)
   - a little uncertainty

2. **Postbelief**
   - \( p = A: 0 \)
   - \( B: 0.5 \)
   - \( C: 0.5 \)
   - more uncertainty

Uncertainty = closeness to uniform distribution

Why Uncertainty Fails

Uncertainty-based approach addresses objective probabilities on system but **not** subjective probabilities of attacker

Metric for Belief

- \( d_1 > d_2 \) postbelief closer to reality because of observation of program

Accuracy

Accuracy: Distance from a belief to reality

- **Accuracy is the correct metric for QIF**
Belief in Information Flow

- **Experiment protocol**
  Describes how attackers revise beliefs from observation of program execution

- **Accuracy metric**
  How change in accuracy of belief can be used to measure the amount of information flow

- **Extensions**
  Repeated experiments, other metrics, misinformation, and insiders

Experiments

**Experiment**: How an attacker revises his belief

<table>
<thead>
<tr>
<th>Prebelief</th>
<th>Postbelief</th>
</tr>
</thead>
<tbody>
<tr>
<td>p = A 0.98, B 0.01, C 0.01</td>
<td>p = A 0, B 0.5, C 0.5</td>
</tr>
</tbody>
</table>

Experiment Protocol

1. Attacker chooses prebelief \( \text{preB} \)
2. System and attacker choose inputs \( H_{in}, L_{in} \)
3. System executes \( S \) and produces observation

Execution modeled as a distribution transformer semantics:

\[ [S] : \text{Dist} \rightarrow \text{Dist} \]
Experiment Protocol

1. Attacker chooses prebelief \( \text{preB} \).
2. System and attacker choose inputs \( H_{in}, L_{in} \).
3. System executes \( S \) and produces observation \( \text{observation} \).
4. Attacker conducts thought-experiment to obtain prediction \( \text{prediction} \).

Belief Revision in PWC

\[
\text{preB} = \begin{pmatrix} p = A, 0.5 \ & B, 0.5 & C, 0.5 \end{pmatrix} \quad \text{and} \quad \text{postB} = \begin{pmatrix} \text{prediction} & \text{observation} \end{pmatrix}
\]

Accuracy Metric

Amount of flow \( Q \) is improvement in accuracy of belief i.e. (initial error) – (final error)

\[
Q = D(\text{preB} \rightarrow H_{in}) - D(\text{postB} \rightarrow H_{in})
\]

Belief Distance

Relative entropy:

\[
D(a \rightarrow b) = \sum_{x} b(x) \log_2 \frac{b(x)}{a(x)}
\]

Unit is (information theoretical) bits.
**Amount of Flow from PWC**

\[ Q = D((0.8, 0.1, 0.1) \to (0, 0, 1)) - D((0, 0.5, 0) \to (0, 0, 1)) \approx 5.6 \text{ bits} \]

*Information is in the eye of the beholder*

Max leakage of lg 3 bits implies uniform p-belief:

\[ p = A, B, C \ 1/3 \text{ each} \]

\[ \langle 0, 0, 1 \rangle \quad \langle 1/3, 1/3, 1/3 \rangle \quad \langle 0, 0.5, 0 \rangle \]

**What Is a Bit of Information?**

Attacker gains one bit of information

*Probability attacker ascribes to reality doubles*

e.g: \( D((0, 0.5, 0) \to (0, 0, 1)) = 1 \text{ bit} \)

**Repeated Experiments**

Experiment protocol is compositional

- \( B \) \rightarrow \( \text{Exp} \) \rightarrow \( B' \) \rightarrow \( \text{Exp} \) \rightarrow \( B'' \)

*Information flow is also compositional*

The amount of information flow over a series of experiments is equal to the sum of the amount of information flow in each individual experiment.

**Extensions of Metric**

- So far: exact flow for a single execution
- Extend to:
  - [Expected, maximum] amount of information flow
    - for a given experiment, over all experiments
- Language for quantitative flow policies

**Misinformation**

FPWC: if \( p = g \) then \( a := 1 \) else \( a := 0 \);
if random() < .1 then \( a := !a \)

Non-probabilistic programs cannot create misinformation.

**Insider Choice**

- Agents so far: attacker and system
- Add insider:
  - Goal: help attacker learn secret information
  - Cannot directly communicate with attacker
  - Can observe entire program state
  - Can influence execution of programs through insider choice:
    \[ S_2 \left[ S_2 \right] \]

\[ h := h \mod 2; \]
\[ 1 := 0 \]
\[ 1 := 1 \]
Modeling Insiders

• Formally, a function of type \textbf{State} $\rightarrow [0,1]$

• Modifications to model:
  – Experiment functions use insider semantics for attacker prediction and observation

Insiders and Nondeterminism

• \textit{Observational determinism}: program behaves as deterministic function from low input to low output, regardless of nondeterministic choices and high inputs. [McLean 92, Roscoe 95, Zdancewic and Myers 01]

  Zero information flow in insider model

  - Observational determinism

• Insider choice not secure under refinement:

  $1 := h \quad l := 0 \quad l := 1$

Summary

• Attackers have beliefs

• Quantifying information flow with beliefs requires accuracy
  – Traditional uncertainty model is inappropriate

• Presented more expressive, fine-grained model of quantitative information flow
  – Compositional experiment protocol
  – Probabilistic language semantics
  – Accuracy-based metric
  – Insider choice

Related Work

• Information theory in information flow
  – [Denning 82], [Mills 87], [Wirtbold, Johnson 90], [Gray 91]

• Quantitative information flow
  – Using uncertainty: [Lowe 02], [Mcver, Morgan 03], [Clark, Hunt, Malacaria 01-05]
  – Using sampling theory: [di Pierro, Hankin, Wiklicky 00-05]

• Database privacy
  – Using relative entropy: [Efsenievski, Gebrke, Srikant 03]

Future Work

• Lattice of security levels
  – Some bits worth more than others

• Enforcement mechanisms

• Language for quantitative security policies

• Quantitative integrity and availability

• Computational limitations on attackers and insiders

Belief in Information Flow

Michael Clarkson
Cornell University

Joint work with Andrew Myers and Fred B. Schneider
Newcastle University
July 3, 2006
Internet Voting

- Clear interest
  - SERVE, Debian (de-votee), program committees, etc.

- CIVS: Cornell Internet Voting Service
  - http://www.cs.cornell.edu/~andru/civs/
  - Users have run ~100 elections with 10-1700 voters

- Offers security guarantees:
  - Whether a voter votes remains secret, even if server storage compromised
  - But assuming trusted server software, and without verifiability

- Redesign to get verifiability and coercion resistance without a trusted server
- Reimplement in Jif (= Java + [information flow])

Trust Model

- We have to trust client software

- Move rest of trust into:
  - Cryptography
  - Anonymous channel
  - Mutually distrusting set of tellers

Overview of Voting Scheme

- Based on [Juels, Catalano, Jakobsson 05]
- Voter registers and receives capability c
- Voter selects vote v
- Voter submits $E(c), E(v)$
- Tellers tally votes by:
  - Eliminating any pairs with invalid capabilities
  - Decrypting and summing all votes
  - While preserving anonymity of voters
  - And without trusting each other

CIVS Enhancements to [JCJ]

- Distributed construction of capabilities
- Condorcet voting methods
- Implemented (!) using Jif

Ranked Voting Methods

- Voters submit ordering of candidates:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vanilla</td>
</tr>
<tr>
<td>2</td>
<td>Chocolate</td>
</tr>
<tr>
<td>3</td>
<td>Strawberry</td>
</tr>
<tr>
<td>4</td>
<td>Cookie dough</td>
</tr>
<tr>
<td>5</td>
<td>Mint chocolate chip</td>
</tr>
</tbody>
</table>

- Captures more information about “the will of the people” than binary voting methods
- Condorcet, STV/IRV, Borda, ...
Condorcet Methods

- Many argue they produce superior results (at least over FPTP)

- Condorcet winner (CW) is the candidate who would defeat every other candidate in a one-on-one plurality vote
  - Chocolate beats Vanilla 60-40
  - Chocolate beats Mint 90-10
  - Strongly democratic: majority rule is enforced

- Resistant to strategic voting
  - Voters have strong incentive to vote true preferences

Condorcet Ballots

- Simple: Decompose rankings into a C * C binary matrix
  - C = number of candidates
  - Cell (i,j) = 1 if voter prefers candidate i to j, 0 otherwise

- Treat each cell as a separate vote
  - Voter casts C(C-1)/2 votes

- Coercion resistant:
  - Tellers anonymize each vote individually
  - Sets not identifiable, so neither are low-order preferences

Secure Implementation

- “We have to trust client software”
- But we can gain assurance in the implementation

- Jif = Java + Information Flow [Myers 99]
  - Core language is Java minus threads, reflection
  - Adds static checking of information flow policies expressed in decentralized label model
    - int(Alice → Alice; Bob); x;

Advantages of Jif Implementation

- Programmer can’t inadvertently violate policies
- Attacker can’t insert source to deliberately violate policies

- However: Haven’t eliminated all trust in software
  - Jif compiler is trusted
  - Voter must trust software stack

Information Flow for E-Voting

- Voter’s vote is only readable by voter, until encrypted (confidentiality)
- Information posted to a public bulletin board cannot be changed (integrity)
- Client software must forget vote after submitting it (erasure)
- Votes cannot be decrypted until anonymized (declassification-until)
Conclusions

- Secure Internet voting
  - Verifiable
  - Coercion-resistant
  - Condorcet voting

- Implementation
  - Statically checked information-flow policies

Michael Clarkson
Cornell University
Joint work with Stephen Chong, Andrew Myers, and Peter Ryan
Newcastle University
July 3, 2006