Quantification of Integrity

Michael Clarkson and Fred B. Schneider
Cornell University

Slides due to Clarkson, with some modification by Hicks for CMSC838G, Spring 2011
Goal

Information-theoretic Quantification of programs’ impact on Integrity of Information

[Denning 1982]
What is Integrity?

Common Criteria:
- Protection of assets from unauthorized modification

Biba (1977):
- Guarantee that a subsystem will perform as it was intended
- Isolation necessary for protection from subversion
- Dual to confidentiality

Databases:
- Constraints that relations must satisfy
- Provenance of data
- Utility of anonymized data

...no universal definition

Clarkson: Quantification of Integrity
Our Notions of Integrity

**Corruption**: damage to integrity

<table>
<thead>
<tr>
<th>Starting Point</th>
<th>Corruption Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taint analysis</td>
<td>Contamination</td>
</tr>
<tr>
<td>Program correctness</td>
<td>Suppression</td>
</tr>
</tbody>
</table>
## Our Notions of Integrity

**Corruption**: damage to integrity

<table>
<thead>
<tr>
<th>Starting Point</th>
<th>Corruption Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taint analysis</td>
<td>Contamination</td>
</tr>
<tr>
<td>Program correctness</td>
<td>Suppression</td>
</tr>
</tbody>
</table>

**Contamination**: bad information present in output

**Suppression**: good information lost from output

...distinct, but interact
Information

Information is surprise.

$X$: random variable on set of events \{e, \ldots\}

$I(e)$: self-information conveyed by event $e$

$I(e) = - \log_2 \Pr[X=e]$ (unit is bits)
Value of Information

What if some bits are worth more than others? Not considered in this paper.

- Discrete worth: security levels
  - *Top secret, secret, confidential, unclassified*

- Continuous worth: ?
Contamination

Goal: model **taint analysis**

[Diagram showing trust relationships between attacker, user, and program]
Contamination

Goal: model **taint analysis**

Untrusted input *contaminates* trusted output
Contamination

\[ o := (t, u) \]

\( u \) contaminates \( o \)
Contamination

\[ o := (t, u) \]

\( u \) contaminates \( o \)

\((Can’t \ u \ be \ filtered \ from \ o?)\)
Quantification of Contamination

Use information theory: information is surprise

$X, Y, Z$: distributions

$I(X, Y)$: mutual information between $X$ and $Y$ (in bits)
$I(X, Y \mid Z)$: conditional mutual information
Quantification of Contamination

untrusted

Attacker

Program

Attacker

trusted

User

Program

User
Quantification of Contamination

untrusted

trusted

Program

\( U_{in} \)

\( T_{in} \)

\( T_{out} \)
Quantification of Contamination

Contamination = \( I(U_{in}, T_{out} \mid T_{in}) \)

[Newsome et al. 2009]

Dual of [Clark et al. 2005, 2007]
Example of Contamination

\[ o := (t, u) \]

Contamination = \( I(U, O \mid T) = k \) bits

*if \( U \) is uniform on \([0,2^k-1]\)*
Contamination vs. Leakage

\[
\text{Contamination} = I(U_{\text{in}}, T_{\text{out}} \mid T_{\text{in}})
\]

\[
\text{Leakage} = I(S_{\text{in}}, P_{\text{out}} \mid P_{\text{in}})
\]

Clarkson: Quantification of Integrity
Contamination vs. Leakage

Contamination = $I(U_{in}, T_{out} \mid T_{in})$

Leakage = $I(S_{in}, P_{out} \mid P_{in})$

Contamination vs. Leakage

Contamination = $I(U_{\text{in}}, T_{\text{out}} \mid T_{\text{in}})$

⇒ Contamination is dual to leakage

Leakage = $I(S_{\text{in}}, P_{\text{out}} \mid P_{\text{in}})$

### Our Notions of Integrity

**Corruption:** damage to integrity

<table>
<thead>
<tr>
<th>Starting Point</th>
<th>Corruption Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taint analysis</td>
<td>Contamination</td>
</tr>
<tr>
<td>Program correctness</td>
<td>Suppression</td>
</tr>
</tbody>
</table>

**Contamination:** bad information present in output

**Suppression:** good information lost from output
Program Suppression

Goal: model program (in)correctness

(Specification must be deterministic)
Program Suppression

Goal: model program (in)correctness

Sender -> Specification -> Receiver

\textit{correct}

\textit{untrusted} Attacker -> Implementation -> Attacker

\textit{trusted} Sender -> Implementation -> Receiver

\textit{real}
Program Suppression

Goal: model program (in)correctness

Implementation might suppress information about correct output from real output

Clarkson: Quantification of Integrity
Example of Program Suppression

Spec.

```plaintext
for (i=0; i<m; i++)
{  s := s + a[i]; }
```

a[0..m-1]: trusted
Example of Program Suppression

Spec.

\[
\text{for (i=0; i<m; i++)} \\
\{ s := s + a[i]; \}
\]

Impl. 1

\[
\text{for (i=1; i<m; i++)} \\
\{ s := s + a[i]; \}
\]

Suppression—\(a[0]\)
missing
No contamination
Example of Program Suppression

Spec.

\[
\text{for } (i=0; i<m; i++) \quad \{ \text{s := s + a[i]; } \}
\]

a[0..m-1]: trusted

Impl. 1

\[
\text{for } (i=1; i<m; i++) \quad \{ \text{s := s + a[i]; } \}
\]

Suppression—a[0] missing
No contamination

Impl. 2

\[
\text{for } (i=0; i<=m; i++) \quad \{ \text{s := s + a[i]; } \}
\]

a[m]: untrusted

Suppression—a[m] added
Contamination
Suppression vs. Contamination

output := input

Contamination

Attacker

Suppression

*
Quantification of Program Suppression

Sender → Specification → Receiver

untrusted
Attacker → Implementation → Attacker

trusted
Sender → Implementation → Receiver

Clarkson: Quantification of Integrity
Quantification of Program Suppression

Sender \rightarrow Specification \rightarrow Receiver

In

Spec

Sender

Receiver

Attacker

Attacker

untrusted

trusted

Implementation

Sender

Receiver

Clarkson: Quantification of Integrity
Quantification of Program Suppression

Sender \rightarrow \text{Specification} \rightarrow Receiver

\begin{itemize}
\item Untrusted
  \begin{itemize}
  \item Attacker
  \item U_{in}
  \item Implementation
  \item Attacker
  \item Impl
  \item Receiver
  \item Tin
  \item Trusted
  \begin{itemize}
  \item Sender
  \item T_{in}
  \item Specification
  \item Receiver
  \end{itemize}
\end{itemize}
\end{itemize}
Quantification of Program Suppression

Program transmission = I(Spec, Impl)
Quantification of Program Suppression

$H(X)$: entropy (uncertainty) of $X$

$H(X|Y)$: conditional entropy of $X$ given $Y$

Program Transmission = $I(\text{Spec}, \text{Impl})$

= $H(\text{Spec}) - H(\text{Spec} \mid \text{Impl})$
Quantification of Program Suppression

$H(X)$: entropy (uncertainty) of $X$

$H(X|Y)$: conditional entropy of $X$ given $Y$

Program Transmission $= I(\text{Spec}, \text{Impl})$

$= H(\text{Spec}) - H(\text{Spec} \mid \text{Impl})$

Total info to learn about Spec
Quantification of Program Suppression

H(X): entropy (uncertainty) of X
H(X|Y): conditional entropy of X given Y

Program Transmission = I(Spec, Impl)
= H(Spec) − H(Spec | Impl)

Total info to learn about Spec

Info actually learned about Spec by observing Impl
Quantification of Program Suppression

\[ H(X): \text{entropy (uncertainty) of } X \]

\[ H(X|Y): \text{conditional entropy of } X \text{ given } Y \]

Program Transmission = \( I(\text{Spec}, \text{Impl}) \)

\[ = H(\text{Spec}) - H(\text{Spec} | \text{Impl}) \]

- Info actually learned about Spec by observing Impl
- Info NOT learned about Spec by observing Impl

Total info to learn about Spec
Quantification of Program Suppression

H(X): entropy (uncertainty) of X
H(X|Y): conditional entropy of X given Y

Program Transmission = I(Spec, Impl)
   = H(Spec) - H(Spec | Impl)

Program Suppression = H(Spec | Impl)
Echo Specification

\[ output := \text{input} \]

trusted

Sender \[ T_{in} \] Receiver

Clarkson: Quantification of Integrity
Echo Specification

trusted

Sender → output := input → Receiver

Sender

untrusted

Attacker → Implementation → Attacker

Sender

trusted

Sender → Output := Input → Receiver

Sender
Echo Specification

trusted Sender $\rightarrow$ output := input $\rightarrow$ Receiver

untrusted Attacker $\rightarrow$ Implementation $\rightarrow$ Attacker

trusted Sender $\rightarrow$ Receiver

\[ T_{in} \]

\[ U_{in} \]

\[ T_{in} \]

\[ T_{out} \]
Simplifies to information-theoretic model of channels, with attacker
Channel Suppression

Channel transmission = $I(T_{in}, T_{out})$

Channel suppression = $H(T_{in} \mid T_{out})$

$(T_{out}$ depends on $U_{in}$)
Example of Program Suppression

Spec.

```
for (i=0; i<m; i++)
{ s := s + a[i]; }
```

Impl. 1

```
for (i=1; i<m; i++)
{ s := s + a[i]; }
```

Impl. 2

```
for (i=0; i<=m; i++)
{ s := s + a[i]; }
```

Suppression = $H(A)$

Suppression $\leq H(A)$

$A =$ distribution of individual array elements
Suppression vs. Contamination

\[ o := (t, u) \]

\[ n := \text{rnd}(); \]
\[ o := t \oplus n \]

\[ o := t \oplus u \]

- \( o \) contaminated by \( u \)
  - no suppression

- \( t \) suppressed by noise
  - no contamination

- \( t \) suppressed by \( u \)
  - \( o \) contaminated by \( u \)
Belief-based Metrics

What if user’s/receiver’s distribution on unobservable inputs is wrong?

Belief-based information flow [Clarkson et al. 2005]

Belief-based generalizes information-theoretic:

- On single executions, the same
- In expectation, the same …if user’s/receiver’s distribution is correct
Suppression and Confidentiality

**Declassifier:** program that reveals (leaks) some information; suppresses rest


**Thm.** Leakage + Suppression is a constant

⇒ What isn’t leaked is suppressed
Database Privacy

Statistical database anonymizes query results:

…sacrifices utility for privacy’s sake
Database Privacy

Statistical database anonymizes query results:

• ...sacrifices utility for privacy’s sake
• ...suppresses to avoid leakage

anon. resp. := resp.
Database Privacy

Statistical database anonymizes query results:

…sacrifices utility for privacy’s sake
…suppresses to avoid leakage
…sacrifices integrity for confidentiality’s sake
Database Privacy Security Conditions

**k-anonymity:** [Sweeney 2002]
- Every individual must be anonymous within set of size $k$.
  - Every output corresponds to $k$ inputs.
    - ...no bound on leakage or suppression

**L-diversity:** [Øhrn and Ohno-Machado 1999, Machanavajjhala et al. 2007]
- Every individual’s sensitive information should appear to have $L$ roughly equally likely values.
  - Every output corresponds to $L$ (roughly) **equally likely** inputs
    - ...implies suppression $\geq \log L$

**Differential privacy:** [Dwork et al. 2006, Dwork 2006]
- No individual loses privacy by including data in database
  - Output reveals almost no information about individual input beyond what other inputs already reveal
    - ...implies almost all information about individual suppressed
    - ...quite similar to noninterference
**L-diversity**

Every individual’s sensitive information should appear to have $L$ (roughly) equally likely values.

[Machanavajjhala et al. 2007]

Entropy $L$-diversity: $H(\text{anon. block}) \geq \log L$

[Øhrn and Ohno-Machado 1999, Machanavajjhala et al. 2007]

$H(T_{in} \mid t_{out}) \geq \log L$ (if $T_{in}$ uniform)

…implies suppression $\geq \log L$
Summary

Measures of information corruption:

- **Contamination** (generalizes taint analysis, dual to leakage)
- **Suppression** (generalizes program correctness, no dual)

Application: database privacy

(model anonymizers; relate utility and privacy; security conditions)
More Integrity Measures

- Channel suppression
  - ...same as channel model from information theory, but with attacker
- Attacker- and program-controlled suppression

Granularity:
- Average over all executions
- Single executions
- Sequences of executions
  - ...interaction of attacker with program

Application: Error-correcting codes
Quantification of Integrity

Michael Clarkson and Fred B. Schneider
Cornell University

IEEE Computer Security Foundations Symposium
July 17, 2010
Confidentiality Dual to Suppression?

→ Classic duality of confidentiality and integrity is incomplete
“To Measure is to Know”

When you can measure what you are speaking about...you know something about it;
but when you cannot measure it...your knowledge is...meager and unsatisfactory...
You have scarcely...advanced to the state of Science.

—Lord Kelvin