Predictive Mitigation of Timing Channels in Interactive Systems

Danfeng Zhang, Aslan Askarov, Andrew C. Myers

Cornell University

CCS 2011
Timing Channels

Hard to detect and prevent
Timing Channels: Examples

• **Cryptographic timing attacks** [Kocher 96, Brumley&Boneh 05, Osvik et. al. 06]
  – RSA, AES keys are leaked by decryption time

• **Cross-site timing attacks** [Bortz&Boneh 07]
  – Load time of a web-page reveals login status, as well as the size and contents of shopping cart

• **Use as covert channels** [Meer&Slaviero 07]
  – Transmit confidential data by controlling response time, e.g., combined with SQL injection

• **Timing channels are big threats to security!**
Timing Channel Mitigation

• Limitations of known approaches
  – Delay to the worst-case execution time – bad performance
  – Add random delays – linear leakage
  – Input blinding – specialized to cryptography

• Our solution:
  – Asymptotically logarithmic leakage
  – Effective in practice
  – Applies to general computation
Outline

• Background on predictive black-box mitigation (CCS’10)

• Predictive mitigation for interactive systems (e.g., web services)
  – Prediction with public information
  – Generalized penalty policy & leakage analysis
  – Composition of mitigators

• Evaluation
Strong attacker model: timing of source events may be controlled
Example: Doubling

When mitigator expects to deliver events

predictions

Mitigator starts with a fixed schedule $S$

$S(i)$ – prediction for $i$-th event
Example: Doubling

When event comes before or at the prediction – delay the event

little information leaked
Example: Doubling

Adversary observes mispredictions
New fixed schedule $S_2$ penalizes the event source

information leaked!
Example: Doubling

**Epoch**: period of time during which mitigator meets all predictions

Little information leaked in each epoch
Leakage & Variations

- Leakage measurement (log of # timing variations)
  - Also bounds
    - mutual information (Shannon entropy)
    - min-entropy

Variations observable by the attacker
Important Features

• Information leaks via *mispredictions*!
• General class of timing mitigators
  – Doubling scheme
  – Adaptive transitions
  – ...

\[
\text{Leakage} \leq O(\log^2 T) \text{ bits}
\]
Outline

• Background on predictive black-box mitigation (CCS’10)

• Predictive mitigation for interactive systems (e.g., web services)
  – Prediction with public information
  – Generalized penalty policy & leakage analysis
  – Composition of mitigators

• Evaluation
Insight: Use Public Information

• Previous black-box model
  – No misprediction: events delivered according to schedule
  – Misprediction: entire schedule is \textit{statically} determined (difficult for interactive systems)
Insight: Use Public Information

• Previous black-box model
  – Schedule is **dynamically** calculated by prediction algorithm
  – No misprediction: schedule is deterministic given public info.
  – Misprediction: select a new prediction algorithm
Outline

• Background on predictive black-box mitigation (CCS’10)

• Predictive mitigation for interactive systems (e.g., web services)
  – Prediction with public information
  – Generalized penalty policy & leakage analysis
  – Composition of mitigators

• Evaluation
Public Information

• Public information in interactive systems
  – *Request types*: public payloads in requests, such as URLs
    • www.example.com/index.html vs. www.example.com/background.gif
  – *Public information in system*: such as input times
  – *Concurrency model*
Prediction with Public Information

• Prediction for request type $r$: $p(N, r)$

• Schedule (output time) for $i^{th}$ event in the $N^{th}$ epoch
  - Single thread
    \[ S_N(i) = \max(inp_i, S_N(i - 1)) + p(N, r_i) \]
  - Multiple, concurrent threads
    • Calculated in similar ways

Schedules are computed dynamically within each epoch using only *public* information
Information Leakage

• Information still leaks via *mispredictions*!
• Formal result

\[
\text{Leakage} \leq N \times \log(M + 1) \text{ bits}
\]
Outline

• Background on predictive black-box mitigation (CCS’10)

• Predictive mitigation for interactive systems (e.g., web services)
  – Prediction with public information
  – Generalized penalty policy & leakage analysis
  – Composition of mitigators

• Evaluation
Penalty Policy – What

Penalty policy

– *Which* type should be penalized?
– *How much* it should be penalized?
Penalty Policy – Why

- Concurrency & request types also bring new threats
- Request types are penalized separately (local penalty policy)
  - Attacker controls the timing of $R$ request types
    - $N$ is proportional to $R$
- Penalize all request types (global penalty policy)
  - Performance is bad

\[
\text{Leakage} \leq N \times \log(M + 1) \text{ bits}
\]
Grace Period Penalty Policy

• Better trade off?
  – Information leaks via *mispredictions*!
  – “Well-behaved” types
    • Few mispredictions, leak little information
    • Share little penalty from other types

• $\lambda$-level grace period policy
  – type $i$ is penalized by other types only when it triggers more than $\lambda$ mispredictions
Leakage Analysis

• Difficult for general penalty policies
  – Influences between request types
  – Different predictions
  – Need to consider *all* possible input sequences

• Principled way of bounding total leakage
  – Transform into optimization problem with \( R \) constraints
    (formal proof & details provided in paper)
Leakage Analysis

Leakage

- Global: $O(\log T \times \log M)$
- Local: $O(R \times \log T \times \log M)$
- Grace period: $O(\log T \times \log M)$
  - Better trade-off
Outline

• Background on predictive black-box mitigation (CCS’10)

• Predictive mitigation for interactive systems (e.g., web services)
  – Prediction with public information
  – Generalized penalty policy & leakage analysis
  – Composition of mitigators

• Evaluation
Composition of Mitigators

• Security guarantee on an interactive system that is
  – composed of mitigated subsystems

• Decompose complicated systems into 2 gadgets
  – Sequential
  – Parallel
Sequential Case

Theoretical results

– Leakage in $O_2 \leq$ Leakage in $O_1$
– Valid for
  • mutual information
  • min-entropy
Parallel Case

Theoretical result

- Leakage in $O_1$ and $O_2$ ≤ Leakage in $O_1$ + Leakage in $O_2$
Outline

• Background on predictive black-box mitigation (CCS’10)

• Predictive mitigation for interactive systems (e.g., web services)
  – Prediction with public information
  – Generalized penalty policy & leakage analysis
  – Composition of mitigators

• Evaluation
Evaluation

Real-world web applications (with HTTP(S) proxy)

Real-world applications

Proxy

Client

10/20/2011
Mitigating Proxy
Evaluation

Measurements

– **Performance**: round-trip latency from the client side
– **Security**: leakage bounds in bits
Experiments with Web Applications

Parameters

- 5-level grace period policy
- Doubling scheme
- Various request types
  - TYPE/HOST
  - HOST+URLTYPE
  - TYPE/URL
Experiments with Web Applications

Mitigating department homepage via HTTP

(49 different requests)

- Predictive mitigation gains good balance (HOST+URLTYPE)
  - About 30% latency overhead
  - At most 850 bits for 100,000 inputs
Experiments with Web Applications

Mitigating department webmail server via HTTPS

– Secure-sensitive (URL is encrypted)
– At most 300 bits for 100,000 inputs
– At most 450 bits for 32M inputs (1 input/sec for one year)

Performance

Latency (ms)

<table>
<thead>
<tr>
<th></th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auth</td>
<td></td>
<td>300</td>
</tr>
<tr>
<td>Login</td>
<td>500</td>
<td>450</td>
</tr>
<tr>
<td>List</td>
<td>900</td>
<td>1200</td>
</tr>
<tr>
<td>Email</td>
<td>1100</td>
<td>1500</td>
</tr>
</tbody>
</table>

Less than 1 second

Security

Webmail leakage

Leakage bound in bits

Number of inputs (X1000)

0 20 40 60 80 100

0 50 100 150 200 250 300 350

10/20/2011
Related Work

• **Timing mitigation for cryptographic operations** [Kocher 96, Kopf & Durmuth 09, Kopf & Smith 10]
  – Assumes input blinding

• **NRL Pump/Network Pump** [Kang et. al. 93, 96]
  – Only address covert channels from input acks
  – Linear bound

• **Information theory community** [Hu 91, Giles&Hajek 02]
  – Timing mitigation based on random delays
  – Linear bound
Conclusion

Predictive mitigation of timing channels

– Generalized predictive mitigation model
  More public information → better performance

– General penalty policies & leakage analysis

– Composition of mitigated systems

– Evaluation suggests this technique is practical