Online Synthesis of Adaptive Side-Channel Attacks Based On Noisy Observations

Lucas Bang, Nicolás Rosner, Tevfik Bultan
Verification Lab (VLab)
Department of Computer Science
University of California Santa Barbara
UC Santa Barbara Verification Lab + Collaborations

- Phan, Bang, Pasareanu, Malacaria, Bultan. [CSF 17]
  “Synthesis of Adaptive Side-Channel Attacks.”
- Bang, Aydin, Phan, Pasareanu, Bultan. [FSE 2016]
  “String Analysis for Side Channels with Segmented Oracles.”
- Bang, Rosner, Bultan. [Euro S&P 2018]
  “Online Synthesis of Adaptive Side-Channel Attacks Based On Noisy Observations.”
- Brennan, Tsiskaridze, Rosner, Aydin, Bultan. [FSE 2017]
  “Constraint normalization and parameterized caching for quantitative program analysis.”

More Related Work

- Köpf, Basin. [CCS 2007]
  “An information-theoretic model for adaptive side-channel attacks”
- Pasareanu, Phan, Malacaria. [CSF 2016]
  “Multi-run Side-Channel Analysis Using Symbolic Execution and Max-SMT.”
- Jia Chen, Yu Feng, Isil Dillig. [CCS 2017]
  “Precise Detection of Side-Channel Vulnerabilities using Quantitative Cartesian Hoare Logic.”
- Antonopoulos, Gazzillo, Hicks, Koskinen, Terauchi, Wei. [PLDI 2017]
  “Decomposition instead of self-composition for proving the absence of timing channels.”
Delivery people at various Domino's pizza outlets in and around Washington claim that they have learned to anticipate big news baking at the White House or the Pentagon by the upsurge in takeout orders. Phones usually start ringing some 72 hours before an official announcement. "We know," says one pizza runner. "Absolutely. Pentagon orders doubled up the night before the Panama attack; same thing happened before the Grenada invasion." Last Wednesday, he adds, "we got a lot of orders, starting around midnight. We figured something was up." This time the big news arrived quickly: Iraq's surprise invasion of Kuwait.
Delivery people at various Domino's pizza outlets in and around Washington claim that they have learned to anticipate big news baking at the White House or the Pentagon by the upsurge in takeout orders. Phones usually start ringing some 72 hours before an official announcement. "We know," says one pizza runner. "Absolutely. Pentagon orders doubled up the night before the Panama attack; same thing happened before the Grenada invasion." Last Wednesday, he adds, "we got a lot of orders, starting around midnight. We figured something was up." This time the big news arrived quickly: Iraq's surprise invasion of Kuwait.
Delivery people at various Domino's pizza outlets in and around Washington claim that they have learned to anticipate big news baking at the White House or the Pentagon by the upsurge in takeout orders. Phones usually start ringing some 72 hours before an official announcement. "We know," says one pizza runner. "Absolutely, Pentagon orders doubled up the night before the Panama attack. Same thing happened before the Grenada invasion." Last Wednesday, he adds, "we got a lot of orders, starting around midnight. We figured something was up." This time the big news arrived quickly: Iraq's surprise invasion of Kuwait.
What is a side channel?

Side channel: learn secrets through indirect observation. secret correlates with observation ⇒ reveal secrets
Secret Data

Program
private s = getBufferSize();

Program
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
private s = getBufferSize();

public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation; // 2 s
  return 0;
}
```java
private int s = getBufferSize();

public int compare(int i) {
    if (s <= i) {
        log.write("too large"); // 1 s
    } else {
        some computation; // 2 s
    }
    return 0;
}
```
```java
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```

\[ s \leq i \Rightarrow o = 1 \]
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}

\[ s \leq i \Rightarrow o = 1 \]
```
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```

\[ s \leq i \implies o = 1 \]

\[ s > i \implies o = 2 \]
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}

\[
\begin{align*}
    s \leq i & \implies o = 1 \\
    s > i & \implies o = 2
\end{align*}
\]
input, $i$

$1 \leq s \leq 8$

$s$?

output, 0

$s \leq i \Rightarrow o = 1$

$s > i \Rightarrow o = 2$
1 private s = getBufferSize();

4 public int compare(int i){
3       if(s <= i)
6           log.write("too large"); // 1 s
7       else
8           some computation; // 2 s
9       return 0;
10     }

\[ s \leq i \Rightarrow o = 1 \]
\[ s > i \Rightarrow o = 2 \]
private s = getBufferSize();

public int compare(int i) {
    if (s <= i) {
        log.write("too large"); // 1 s
    } else {
        some computation; // 2 s
    }
    return 0;
}
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5     if(s <= i)
6         log.write("too large"); // 1 s
7     else
8         some computation; // 2 s
9     return 0;
10 }

\[ s \leq i \Rightarrow o = 1 \]
\[ s > i \Rightarrow o = 2 \]
```java
private s = getBufferSize();

public int compare(int i)
{
    if (s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```

- \( s \leq i \Rightarrow o = 1 \)
- \( s > i \Rightarrow o = 2 \)
```java
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```

- If $s \leq i$, then $o = 1$ (log.write("too large");)
- If $s > i$, then $o = 2$ (some computation;)

Input $i$, observe time.

Constraints:
- $1 \leq s \leq 8$
```java
private int s = getBufferSize();

public int compare(int i) {
    if (s <= i) {
        log.write("too large"); // 1 s
    } else {
        some computation; // 2 s
    }
    return 0;
}
```

Input, \( i \)

1 \( \leq s \leq 8 \)

- \( o = 1 \Rightarrow s \leq i \)
- \( o = 2 \Rightarrow s > i \)
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}

\[ o = 1 \Rightarrow s \leq i \]
\[ o = 2 \Rightarrow s > i \]
```java
1 private s = getBufferSize();
2
3 public int compare(int i){
4     if(s <= i)
5         log.write("too large"); // 1 s
6     else
7         some computation; // 2 s
8     return 0;
9 }
```
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}

ATTACKER can binary search on s using i and o.

input, 4

1 ≤ s ≤ 8

observe time

o = 1 ⇒ s ≤ 4

o = 2 ⇒ s > 4
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large");  // 1 s
    else
        some computation;  // 2 s
    return 0;
}
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```java
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```java
private int s = getBufferSize();

public int compare(int i) {
    if (s <= i) {
        log.write("too large"); // 1 s
    } else {
        some computation; // 2 s
    }
    return 0;
}
```
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```java
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```
private s = getBufferSize();

public int compare(int i) {
    if (s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}

\[ s \leq i \Rightarrow o = 1 \]
```java
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```
private s = getBufferSize();

public int compare(int i) {
    if (s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```java
private s = getBufferSize();

public int compare(int i) {
    if (s <= i) {
        log.write("too large"); // 1 s
    } else {
        some computation; // 2 s
    }
    return 0;
}
```
```java
private s = getBufferSize();

public int compare(int i){
    if(s <= i)
        log.write("too large"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```
Attacker Belief? $s$?
Attacker Belief?

$s$?
Attacker Belief?  
Input Choice?

\[ s? \]

\[ i^* \]
Attacker Belief? $s$? 

Input Choice? $i^* = 5$ 

Observation? $s \leq 5$, $s > 5$
Attacker Belief? $s$?  

Input Choice? $i^* = 5$  

Observation? $s \leq 5, s > 5$

$t = 4.12$
Attacker Belief? $s_?$

Input Choice? $i^* = 5$

Observation? $s \leq 5, s > 5$

$t = 4.12$
Attacker Belief?

Input Choice?

Observation?

\[ s? \]

\[ i^* = 5 \]

\[ s \leq 5 \]

\[ s > 5 \]
Attacker Belief?  

\[ s ? \]

\[
\begin{array}{ccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

Input Choice?  

\[ i^* = 5 \]

Observation?  

\[ s \leq 5 \quad s > 5 \]
Attacker Belief? \( s? \)

Input Choice? \( i^* = 5 \)

Observation?

- \( s \leq 5 \)
- \( s > 5 \)

\( t = 2.3 \)
Attacker Belief?

\( s \)?

Input Choice?

\( i^* = 5 \)

Observation?

\( s \leq 5 \)

\( s > 5 \)
Attacker Belief?  

$$s?$$

Input Choice?  

$$i^* = 5$$

Observation?  

$$s \leq 5$$  

$$s > 5$$

$$p(s|o, i^*)$$
Attacker Belief?  

\[ s? \]

Input Choice?  

\[ i^* = 5 \]

Observation?  

\[ s \leq 5 \quad s > 5 \]
Attacker Belief?

\[ p(s|o, i^*) \]

Input Choice?

\[ i^* = 5 \]

Observation?

\[ p(o|s, i) \]

\[ s \leq 5 \]

\[ s > 5 \]
Attacker Belief?  Input Choice?  Observation?

\[ \text{Attacker Belief?} \quad s? \]

\[ \text{Input Choice?} \quad i^* = 5 \]

\[ \text{Observation?} \quad s \leq 5 \quad s > 5 \]

\[ p(s|o, i^*) \quad p(o|s, i) \]
$\text{Attacker Belief?}$

$s$?

$\text{Input Choice?}$

$i^* = 5$

$\text{Observation?}$

$p(o|s,i)$

$p(s|i,o)$

$p(s|o,i^*)$
Attacker Belief?  Input Choice?  Observation?

$s$?

$i^* = 5$

$s \leq 5$

$s > 5$

$p(s|o, i^*) \quad p(o|s, i^*) \quad p(o|s, i)$
Atacker Belief? $s$?

Input Choice? $i^* = 5$

Observation?

\[ p(s|o, i^*) \]

\[ p(o|s, i^*) \]

\[ p(o|s, i) \]
Attacker Belief?  

$s$?  

Input Choice?  

$i^* = 5$  

Observation?  

$s \leq 5$  

$s > 5$  

$p(s|o,i^*)$  

$p(s|o,i^*)$  

$p(o|s,i)$
Attacker Belief?  

Input Choice?  

Observation?  

\[
p(o|s, i^*) \\text{Bayes' Rule} \\ p(s|o, i^*) \\ p(s|o, i^*) \\ p(o|s, i) \\
\]

\( i^* = 5 \)

\( s \leq 5 \)

\( s > 5 \)

\( \frac{1}{8} \)

\( 1 2 3 4 5 6 7 8 \)
Attacker Belief?  
Input Choice?  
Observation?

\[ s \]

\[ i^* = 5 \]

\[ p(s \mid o, i^*) \]

Bayes’ Rule

\[ p(o \mid s, i) \]
Attacker Belief?  
Input Choice?  
Observation?

$p(s|o, i^*)$

Bayes' Rule

$max \mathcal{I}(i)$

$p(o|s, i)$

$p(s|o, i^*)$
Our Approach

\[ \mathcal{I}(i) \] is a symbolic expression over program inputs \( i \) that measures how much information is gained by an attacker when making input \( i \).

Find \( i \) that maximizes \( \mathcal{I}(i) \) to get the attacker’s best input at every step.
1. Offline Static Analysis
1. Offline Static Analysis

2. Offline Dynamic Analysis
1. Offline Static Analysis

2. Offline Dynamic Analysis

3. Online Attack Synthesis
1. Offline Static Analysis

2. Offline Dynamic Analysis

3. Online Attack Synthesis
$P(s, i)$
Source Code
$P(s, i)$  \[\rightarrow\] Symbolic Execution  \[\rightarrow\] $\{PC_j(s, i)\}$  

path constraints
$P(s, i)$

Source Code

Symbolic Execution

$\{PC_j(s, i)\}$

Path constraints

$\{w_j = (s_j, i_j)\}$

PC models (witnesses)
Each PC characterizes an observable program behavior
Each PC characterizes an observable program behavior

\[(s, i) \models PC_j \quad (s', i') \models PC_j\]
Each PC characterizes an observable program behavior

\[(s, i ) \models PC_j \quad (s', i' ) \models PC_j \]

\[P (s, i ) \quad P (s', i' )\]
Each PC characterizes an observable program behavior

\[(s, i) \models PC_j \quad \quad (s', i') \models PC_j\]

\[P(s, i) \quad ? \quad ? \quad P(s', i')\]
Each PC characterizes an observable program behavior

\[(s, i) \models PC_j \quad (s', i') \models PC_j\]

\[P(s, i) \quad ? \quad P(s', i')\]

\[PC_j (s, i)\] characterizes indistinguishable behaviors

\[P(s, i)\] is a representative of all behaviors in that class
$P(s, i)$

Source Code

Symbolic Execution

$\{PC_j(s, i)\}$

path constraints

$\{w_j = (s_j, i_j)\}$

PC models (witnesses)
1. Offline Static Analysis

2. Offline Dynamic Analysis

3. Online Attack Synthesis
1. Offline Static Analysis

2. Offline Dynamic Analysis

3. Online Attack Synthesis
Characterize effect of noise on each class of program behaviors using the witness for that behavior.
Characterize effect of noise on each class of program behaviors using the witness for that behavior.
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

\[ \{ w_j = (s_j, i_j ) \} \]
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

\[ \{ w_j = (s_j, i_j ) \} \]
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

$$\{w_j = (s_j, i_j)\}$$
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

\[ w_j = (s_j, i_j) \]
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

\[ w_j = (s_j, i_j) \] \times 1000

\{w_j = (s_j, i_j)\}
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

\[ w_j = (s_j, i_j) \]
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

\[ \times 1000 \]
\[ \{ w_j = (s_j, i_j ) \} \]

\[ P(s, i) \]

\[ \text{Network} \]

\[ \text{HW / OS} \]

\[ \{ p(o|s_j, i_j ) \} \]

Smooth Kernel Density Estimation
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

\[ \times 1000 \quad \{ w_j = (s_j, i_j) \} \]

\[ P(s, i) \]

\[ \text{HW / OS} \]

\[ \sum_{k=1}^{N} K \left( \frac{o - o_k}{\delta} \right) \]

Smooth Kernel Density Estimation
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

\[ w_j = (s_j, i_j) \]

\[ P(s, i) \]

HW / OS

Network

\[ p(o|s_j, i_j) \]

\[ \times 1000 \]

\[ \{ w_j = (s_j, i_j) \} \]

Merging via Hellinger Distance

\( d_H = 0.068 \) (a) \( d_H = 0.491 \) (b) \( d_H = 0.978 \) (c)

Merging via Hellinger Distance
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

\[
\{w_j = (s_j, i_j )\}
\]

\[
P(s, i)
\]

\[
\times 1000
\]

\[
p(o|s_j, i_j)
\]

\[
d_H(p, q) = \sqrt{\frac{1}{2} \int_{-\infty}^{\infty} \left( \sqrt{p(x)} - \sqrt{q(x)} \right)^2 \, dx}
\]

Merging via Hellinger Distance
1. Offline Static Analysis

2. Offline Dynamic Analysis

3. Online Attack Synthesis
Belief $p(s)$

$p(o|PC')$
The image shows a diagram with the following components:

- A set of variables represented as $\{\text{Belief}, p(s), p(o|PC)\}$.
- An arrow pointing to a MAX operator labeled $\mathcal{I}$.
- An arrow pointing from the MAX operator to $i^*$.

The variables $p(s)$ and $p(o|PC)$ are inputs to the MAX operator, which outputs $i^*$. The diagram illustrates a decision-making process where belief states and their associated probabilities are evaluated to select the optimal action $i^*$. The devil emoji indicates a potential adversarial or uncertain context in the decision process.
\[
\{ \mathcal{I} \} \quad p(o|PC) \\
\quad p(s) \quad \text{MAX } \mathcal{I} \\
\quad i^* \quad P(s,i) \\
\]

Belief
\{ A, A, \ldots \} \quad p(o|PC')

p(s|o, i^*)

MAX \mathcal{I} \quad \rightarrow \quad i^*

\text{Network} \quad P(s, i)

Belief

Bayesian Update

observe o

HW / OS
\{X, Y, \ldots\} \quad p(o|PC')

p(s|o, i^*)

MAX \mathcal{I}

\implies \quad i^*

\implies \quad Belief

\implies \quad \text{Bayesian Update}

\implies \quad \text{observe } o

\implies \quad \text{Network}

\implies \quad P(s, i)

\implies \quad \text{HW / OS}
\[
\mathcal{I}(s; PC_j|i) = -\sum_{j=1}^{n} p(PC_j|i) \log_2 p(PC_j|i)
\]
\[
\mathcal{I}(s; PC_j|i) = -\sum_{j=1}^{n} p(PC_j|i) \log_2 p(PC_j|i)
\]

Expected info gain given attacker input

Path constraint probabilities

\[
#(PC_j|i) = \sum_{s \in S} \left\{ \begin{array}{cl} 1 & \text{if } (s, i) \models PC_j \\ 0 & \text{otherwise} \end{array} \right. 
\]
\[
\mathcal{I}(s; PC_j | i) = - \sum_{j=1}^{n} p(PC_j | i) \log_2 p(PC_j | i)
\]

Expected info gain given attacker input

Path constraint probabilities

\[
\#(PC_j | i) = \sum_{s \in S} \begin{cases} 
1 & \text{if } (s, i) \models PC_j \\
0 & \text{otherwise}
\end{cases}
\]

Model Counting
\[ I(s; PC_j | i) = - \sum_{j=1}^{n} p(PC_j | i) \log_2 p(PC_j | i) \]

\[ p(PC_j | i) = \sum_{s \in S} p(s) \times \begin{cases} 
1 & \text{if } (s, i) \models PC_j \\
0 & \text{otherwise}
\end{cases} \]

Belief

Bayesian Update

\( P(s, i) \)

Network

Path constraint probabilities

Expected info gain given attacker input

Weighted Model Counting

observe \( o \)
\[
\mathcal{I}(s; PC_j|i) = -\sum_{j=1}^{n} p(PC_j|i) \log_2 p(PC_j|i)
\]

\[
p(PC_j|i) = \sum_{s \in S} p(s) \times \begin{cases} 
1 & \text{if } (s, i) \models PC_j \\
0 & \text{otherwise}
\end{cases}
\]

Expected info gain given attacker input

Path constraint probabilities

Belief

Bayesian Update

observe \( o \)

\[
p(o|PC)
\]

\[
p(s|o, i^*)
\]

\[
\{ \}
\]

\[
\{, \ldots \}
\]

\[
\text{MAX } \mathcal{I}
\]\n
Network

\[
P(s, i)
\]

HW / OS

\[
\text{Expected info gain given attacker input}
\]

\[
\text{Path constraint probabilities}
\]

\[
\text{Weighted Model Counting}
\]

\[
\text{Barvinok}
\]
1. Offline Static Analysis
2. Offline Dynamic Analysis
3. Online Attack Synthesis
Implementation

NASA Symbolic PathFinder (SPF)  Z3 Constraint Solver

Python Profiler Client  Intel NUC Server

Barvinok
Weighted Symbolic Model Counting

Mathematica
Symbolic Entropy Computation Numeric Maximization

\[ P(s, i) \]
Case Study: LawDB

From Defense Advanced Research Projects Agency (DARPA) Space-Time Analysis for Cybersecurity (STAC) Project

41 classes, 2844 line of code. DB: key = employee ID. Some employee IDs have restricted access.

SEARCH midID maxID

List of employees.

Server

41 classes, 2844 line of code. DB: key = employee ID. Some employee IDs have restricted access.

Writes to log file depending on

\[ ID_{res} \in [minID, maxID] \]
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 0: SEARCH

Observed time: --

Entropy = 6.64386
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

**STEP 1: SEARCH 19 52**

*Observed time:* 0.00444

*Entropy* = 6.27408
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

**STEP 2: SEARCH 10 63**

Observed time: 0.00436

Entropy = 5.81014
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 3: SEARCH 1 63

Observed time: 0.0043

Entropy = 5.28658
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

**STEP 4: SEARCH 63 85**

Observed time: 0.00733

Entropy = 3.53218
\[ 1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92 \]

STEP 5: SEARCH 70 73

Observed time: 0.00447

Entropy = 3.19249
1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92

STEP 6: SEARCH 67 74

Observed time: 0.00427

Entropy = 2.74012
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 7: SEARCH 63 74

Observed time: 0.00452

Entropy = 2.41548
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 8: SEARCH 63 70

Observed time: 0.00435

Entropy = 2.07286
\[ 1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92 \]

STEP 9: SEARCH 74 75

Observed time: 0.00431

Entropy = 2.46103
1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92

STEP 10: SEARCH 74 75

Observed time: 0.00435
Entropy = 2.39414
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 11: SEARCH 63 100

Observed time: 0.00732

Entropy = 4.19456
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

**STEP 12: SEARCH 74 100**

Observed time: 0.00743

Entropy = 4.73142
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 13: SEARCH 78 100

Observed time: 0.00733

Entropy = 4.70767
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 14: SEARCH 86 100

Observed time: 0.00728

Entropy = 4.68363
\[ 1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92 \]

**STEP 15: SEARCH 87 99**

- Observed time: 0.00716
- Entropy = 4.37901
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 16: SEARCH 87 95

Observed time: 0.00727

Entropy = 3.83405
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 17: SEARCH 91 95

Observed time: 0.00731

Entropy = 3.87438
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 18: SEARCH 92 95

Observed time: 0.0072
Entropy = 2.9822
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

**STEP 19: SEARCH 92 94**

Observed time: 0.00729

Entropy = 2.98878
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 20: SEARCH 92 93

Observed time: 0.00735

Entropy = 2.22644
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 21: SEARCH 92 92

Observed time: 0.00739

Entropy = 0.767476
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 22: SEARCH 92 92

Observed time: 0.00715

Entropy = 0.170871
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 23: SEARCH 92 92

Observed time: 0.00746

Entropy = 0.026079
1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92

STEP 24: SEARCH 92 92

Observed time: 0.00721

Entropy = 0.026084
<table>
<thead>
<tr>
<th>ID Range</th>
<th># Employees</th>
<th>Offline Analysis</th>
<th>Attack time (m)</th>
<th># steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-100</td>
<td>3</td>
<td>57s</td>
<td>2m38s</td>
<td>25</td>
</tr>
<tr>
<td>1-10000</td>
<td>4</td>
<td>2m21s</td>
<td>2m43s</td>
<td>45</td>
</tr>
<tr>
<td>1-100000</td>
<td>5</td>
<td>6m30s</td>
<td>3m08s</td>
<td>48</td>
</tr>
<tr>
<td>1-100000</td>
<td>10</td>
<td>42m09s</td>
<td>4m31s</td>
<td>77</td>
</tr>
</tbody>
</table>
Our Approach

Symbolic Execution
System Profiling
Entropy

Automatically synthesize side-channel attacks!
Thanks!

Questions?