Ph.D. Proposal

Software Side-Channel Analysis

Lucas Bang

Department of Computer Science, University of California, Santa Barbara

UCSB
INTRODUCTION

What is a side channel?
What is a side channel?

Monday, Aug. 13, 1990

And Bomb The Anchovies

By Paul Gray

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?

Monday, Aug. 13, 1990

And Bomb The Anchovies

By Paul Gray

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?

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?

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?

TIME
Monday, Aug. 13, 1990
And Bomb The Anchovies
By Paul Gray

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. Phone lines 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 information correlates with observation ⇒ reveal secrets
What is a side channel?

Side channel: learn secrets through indirect observation. Secret information correlates with observation $\Rightarrow$ reveal secrets.
What is a software side channel?

input, $i$ \rightarrow \text{secret data, } s \rightarrow \text{output}

Program
What is a software side channel?

input, $i$ \rightarrow \text{secret data, } s \rightarrow \text{output}

\text{Program}

\text{side-channel observation, } o
\text{time, memory, network packet size}
What is a software side channel?

input, $i$ → secret data, $s$ → output

Program

side-channel observation, $o$
time, memory, network packet size

correlation between $(i, o)$ and $s$ ⇒ vulnerability
Side Channels and Searching

\[ i_0 \in I \]

secret \( s \in S \)
Side Channels and Searching

secret \( s \in S \)

\[
i_0 \in I \\
P(i_0, s)
\]
$i_0 \in I$

$P(i_0, s)$

secret $s \in S$
$i_0 \in I$

$P(i_0, s)$

secret $s \in S$
Side Channels and Searching

secret $s \in S$

$i_0 \in I$

$P(i_0, s)$
Side Channels and Searching

\begin{align*}
i_0 & \in I \\
P(i_0, s) & \\
i_1 & \in I \\
P(i_1, s) &
\end{align*}
Side Channels and Searching

secret $s \in S$

$i_0 \in I$
$P(i_0, s)$

$i_1 \in I$
$P(i_1, s)$
Side Channels and Searching

secret $s \in S$

$i_0 \in I$
$P(i_0, s)$

$i_1 \in I$
$P(i_1, s)$
$i_0 \in I$

$P(i_0, s)$

$i_1 \in I$

$P(i_1, s)$

secret $s \in S$
Side Channels and Searching

secret $s \in S$

$i_0 \in I$
$P(i_0, s)$

$i_1 \in I$
$P(i_1, s)$
Secret $s \in S$

- $i_0 \in I$
  - $P(i_0, s)$
- $i_1 \in I$
  - $P(i_1, s)$
- $i_2 \in I$
  - $P(i_2, s)$
secret $s \in S$

$i_0 \in I$ 
$P(i_0, s)$

$i_1 \in I$ 
$P(i_1, s)$

$i_2 \in I$ 
$P(i_2, s)$
What is a software side channel?
What is a software side channel?

Secret Data

Program
What is a software side channel?

```
1 private s = getMaxBytes();
```

Program
What is a software side channel?

```java
private s = getMaxBytes();

public int compare(int i){
    if(s <= i)
        log.write("too many bytes"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```
What is a software side channel?

```java
private s = getMaxBytes();

public int compare(int i){
    if(s <= i)
        log.write("too many bytes"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```
What is a software side channel?

Can this code leak information about `input, i`?

```java
private s = getMaxBytes();

public int compare(int i){
    if(s <= i)
        log.write("too many bytes"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```
What is a software side channel?

```java
private s = getMaxBytes();

public int compare(int i){
    if(s <= i)
        log.write("too many bytes"); // 1 s
    else
        some computation; // 2 s
    return 0;
}
```
What is a software side channel?

```java
private s = getMaxBytes();

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

\[
s \leq i \Rightarrow o = 1
\]
What is a software side channel?

Private $s = \text{getMaxBytes}()$

Public int compare(int $i$) {
    if ($s \leq i$)
        log.write("too many bytes"); // 1 s
    else
        some computation; // 2 s
    return 0;
}

$s \leq i \Rightarrow o = 1$
What is a software side channel?

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

\[ s \leq i \implies o = 1 \]
\[ s > i \implies o = 2 \]
What is a software side channel?

```java
private int s = getMaxBytes();

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

\[
s \leq i \implies o = 1
\]
\[
s > i \implies o = 2
\]
What is a software side channel?

```
private s = getMaxBytes();

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

\[
\begin{align*}
s \leq i & \Rightarrow o = 1 \\
\text{s > i} & \Rightarrow o = 2
\end{align*}
\]
What is a software side channel?

```java
private s = getMaxBytes();

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

$$s \leq i \Rightarrow o = 1$$
$$s > i \Rightarrow o = 2$$
What is a software side channel?

```
private s = getMaxBytes();

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

\[ s \leq i \Rightarrow o = 1 \]
\[ s > i \Rightarrow o = 2 \]
What is a software side channel?

```java
private s = getMaxBytes();

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

\[ s \leq i \Rightarrow o = 1 \]
\[ s > i \Rightarrow o = 2 \]
What is a software side channel?

```
private s = getMaxBytes();

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

\[ s \leq i \Rightarrow o = 1 \]
\[ s > i \Rightarrow o = 2 \]
What is a software side channel?

```java
private int s = getMaxBytes();

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

- If \( s \leq i \) then \( o = 1 \)
- If \( s > i \) then \( o = 2 \)
What is a software side channel?

```
private s = getMaxBytes();

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

Input, i

\[ s \leq i \Rightarrow o = 1 \]
\[ s > i \Rightarrow o = 2 \]
What is a software side channel?

private s = getMaxBytes();

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

input, i

s?

observe time

o = 1 \Rightarrow s \leq i

o = 2 \Rightarrow s > i
What is a software side channel?

```java
private s = getMaxBytes();

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

input, $i$

$s$?

observe time

$s = 1 \Rightarrow s \leq i$

$s = 2 \Rightarrow s > i$

$o = 1$

$s$

$o = 2$
What is a software side channel?

```
private s = getMaxBytes();

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

**Side channel:** $(o, i)$ correlates with $s \Rightarrow$ reveal secret information

- $o = 1 \Rightarrow s \leq i$
- $o = 2 \Rightarrow s > i$
Goal:
Goal:

Given a program, P, determine if P is vulnerable to side channel attacks
Goal:
Given a program, $P$, determine if $P$ is vulnerable to side channel attacks.

How?
Synthesize an attack!
Adaptive Attack Trees
Adaptive Attack Trees

\[ o = 1 \implies s \leq i \quad \text{and} \quad o = 2 \implies s > i \]
Adaptive Attack Trees

\[ o = 1 \Rightarrow s \leq i \quad o = 2 \Rightarrow s > i \]

\[
\begin{array}{cccccccc}
S & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]
Adaptive Attack Trees

\[ o = 1 \Rightarrow s \leq i \quad o = 2 \Rightarrow s > i \]

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

\[ i = 6 \quad 😞 \]
Adaptive Attack Trees

\[ o = 1 \implies s \leq i \quad o = 2 \implies s > i \]

\[ S = \{1, 2, 3, 4, 5, 6, 7, 8\} \]

\[ i = 6 \]
Adaptive Attack Trees

\[ o = 1 \implies s \leq i \quad o = 2 \implies s > i \]

\[ S \]

\[ 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \]

\[ i = 6 \]

\[ o = 1 \]
Adaptive Attack Trees

\[ o = 1 \implies s \leq i \quad o = 2 \implies s > i \]

\[ S = \{1, 2, 3, 4, 5, 6, 7, 8\} \]

\[ i = 6 \]

\[ o = 1 \implies s \leq 6 \]
Adaptive Attack Trees

\[
o = 1 \Rightarrow s \leq i \\
o = 2 \Rightarrow s > i
\]

\[
S = \begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{pmatrix}
\]

\[
i = 6
\]

\[
o = 1 \Rightarrow s \leq 6
\]

\[
S = \begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{pmatrix}
\]
Adaptive Attack Trees

<table>
<thead>
<tr>
<th>S</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>o = 1 ⇒ s ≤ i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>o = 2 ⇒ s &gt; i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- If \( o = 1 \), then \( s \leq i \)
- If \( o = 2 \), then \( s > i \)

![Diagram](image-url)

- If \( i = 6 \), then \( s \leq 6 \)
Adaptive Attack Trees

\[ o = 1 \Rightarrow s \leq i \quad \quad o = 2 \Rightarrow s > i \]

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

\[ i = 6 \]

\[ o = 1 \Rightarrow s \leq 6 \]

\[ S = \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array} \]
Adaptive Attack Trees

\[ o = 1 \Rightarrow s \leq i \quad o = 2 \Rightarrow s > i \]

\[ S = \{1, 2, 3, 4, 5, 6, 7, 8\} \]

\[ i = 6 \]

\[ o = 1 \Rightarrow s \leq 6 \]

\[ o = 2 \]

\[ S = \{1, 2, 3, 4, 5, 6, 7, 8\} \]
Adaptive Attack Trees

\[ o = 1 \Rightarrow s \leq i \quad o = 2 \Rightarrow s > i \]

\[ S \]

\[ i = 6 \]

\[ o = 1 \Rightarrow s \leq 6 \quad o = 2 \]

\[ S \]
Adaptive Attack Trees

\[ o = 1 \Rightarrow s \leq i \quad \text{and} \quad o = 2 \Rightarrow s > i \]

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

\[ i = 6 \]

\[ o = 1 \Rightarrow s \leq 6 \quad \text{and} \quad o = 2 \Rightarrow s > 6 \]

\[ S = \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array} \]
Adaptive Attack Trees

\[ o = 1 \Rightarrow s \leq i \quad o = 2 \Rightarrow s > i \]

\[ S \]

\[ i = 6 \]

\[ o = 1 \Rightarrow s \leq 6 \quad o = 2 \Rightarrow s > 6 \]
Adaptive Attack Trees

\[ o = 1 \implies s \leq i \quad \text{and} \quad o = 2 \implies s > i \]

attacker’s \((i, o)\) partitions \(S\) domain
Adaptive Attack Trees

\[ o = 1 \implies s \leq i \quad o = 2 \implies s > i \]

S

1 2 3 4 5 6 7 8

i = 6

1 2 3 4 5 6 7 8

i = 5

1 2 3 4 5 6 7 8

i = 8
Adaptive Attack Trees

$\text{attacker's } (i, o) \text{ sequences partition the } S \text{ domain}$
Adaptive Attack Trees

\[ o = 1 \Rightarrow s \leq i \quad \text{and} \quad o = 2 \Rightarrow s > i \]

\[ S \]

1 2 3 4 5 6 7 8

\[ i = 6 \]

1 2 3 4 5 6

\[ i = 5 \]

1 2 3 4 5

\[ i = 8 \]

1 2 3

11C
Adaptive Attack Trees

\[ o = 1 \Rightarrow s \leq i \quad o = 2 \Rightarrow s > i \]

\( S \)

1 2 3 4 5 6 7 8

\( i = 6 \)

1 2 3 4 5 6 7 8

\( i = 5 \)

1 2 3 4 5 6 7 8

\( i = 8 \)
Adaptive Attack Trees

\[ o = 1 \implies s \leq i \quad \quad o = 2 \implies s > i \]

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

\[ i = 6 \]

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

\[ i = 5 \]

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

\[ i = 8 \]
Adaptive Attack Trees

\[ o = 1 \Rightarrow s \leq i \quad o = 2 \Rightarrow s > i \]

\[ S = \{1, 2, 3, 4, 5, 6, 7, 8\} \]

1. \( o = 1 \Rightarrow s \leq i \)
   - \( i = 6 \)
   - \( i = 5 \)
   - \( i = 8 \)

2. \( o = 2 \Rightarrow s > i \)
Adaptive Attack Trees

\[ o = 1 \implies s \leq i \quad \text{and} \quad o = 2 \implies s > i \]

S

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

\[ i = 6 \]

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

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
i = 8 \\
\end{array}
\]
Adaptive Attack Trees

\[ o = 1 \Rightarrow s \leq i \quad o = 2 \Rightarrow s > i \]

How to choose the best partition?
Entropy: Side Channels and Searching

secret $s \in S$

$i_0 \in I$
Entropy: Side Channels and Searching

$i_0 \in I$

secret $s \in S$
Entropy: Side Channels and Searching

secret $s \in S$

$i_0 \in I$
Entropy: Side Channels and Searching

\[ i_0 \in I \]

\[ \text{secret } s \in S \]
secret $s \in S$

$i_0 \in I$
Entropy: Side Channels and Searching

secret $s \in S$

$i_0 \in I$

$o_1$

$o_2$
Entropy: Side Channels and Searching

\[ i_0 \in I \]

\[ \text{secret } s \in S \]

Good outcome, very unlikely.
Entropy: Side Channels and Searching

\[ i_0 \in I \]

Bad outcome, very likely.

secret \( s \in S \)
secret $s \in S$

$i_0 \in I$
Entropy: Side Channels and Searching

secret $s \in S$

$i_0 \in I$
$i_1 \in I$
$i_2 \in I$
Entropy: Side Channels and Searching

\[ s \in S \]

\[ i_0 \in I \]
\[ i_1 \in I \]
\[ i_2 \in I \]
Entropy: Side Channels and Searching

secret $s \in S$

$i_0 \in I$

$i_1 \in I$

$i_2 \in I$

$p(s \in \text{region})$
Entropy: Side Channels and Searching

secret $s \in S$

$i_0 \in I$
$i_1 \in I$
$i_2 \in I$

$$p(s \in \text{ } \ ) = \frac{| }{| } | $$
Entropy: Side Channels and Searching

secret \( s \in S \)

\[ i_0 \in I \]
\[ i_1 \in I \]
\[ i_2 \in I \]
secret $s \in S$

\[ i_0 \in I \]
\[ i_1 \in I \]
\[ i_2 \in I \]

Quantify expected info gain measured in bits.
secret $s \in S$

$i_0 \in I$
$i_1 \in I$
$i_2 \in I$

Quantify expected info gain measured in bits.

$$\frac{1}{p_j}$$
secret $s \in S$

\[ i_0 \in I \]
\[ i_1 \in I \]
\[ i_2 \in I \]

Quantify expected info gain measured in bits.

\[ \log_2 \frac{1}{p_j} \]
secret $s \in S$

$i_0 \in I$

$i_1 \in I$

$i_2 \in I$

Quantify expected info gain measured in bits.

$$\sum_{j=1}^{n} p_j \log_2 \frac{1}{p_j}$$
Entropy: Side Channels and Searching

secret $s \in S$

$i_0 \in I$

$i_1 \in I$

$i_2 \in I$

Quantify expected info gain measured in bits.

$$\mathcal{H} = \sum_{j=1}^{n} p_j \log_2 \frac{1}{p_j}$$
Entropy: Side Channels and Searching

Quantify expected info gain measured in bits.

\[ H = \sum_{j=1}^{n} p_j \log_2 \frac{1}{p_j} \]
$o = 1 \Rightarrow s \leq i \quad o = 2 \Rightarrow s > i$
\[ o = 1 \implies s \leq i \quad \text{or} \quad o = 2 \implies s > i \]
\[ o = 1 \implies s \leq i \quad o = 2 \implies s > i \]
\[ o = 1 \Rightarrow s \leq i \quad o = 2 \Rightarrow s > i \]
\[ o = 1 \implies s \leq i \quad \text{and} \quad o = 2 \implies s > i \]
\[ o = 1 \implies s \leq i \quad \text{and} \quad o = 2 \implies s > i \]
max $\mathcal{H} \Rightarrow$ Binary Search
max $\mathcal{H}$ $\Rightarrow$ Binary Search

$o = 1 \Rightarrow s \leq i$  

$o = 2 \Rightarrow s > i$
\[
\max H \Rightarrow \text{Binary Search}
\]
\[
o = 1 \Rightarrow s \leq i \quad o = 2 \Rightarrow s > i
\]
\[
\max H \Rightarrow \text{Optimal Search}
\]
any program constraints
max $\mathcal{H} \Rightarrow$ Binary Search

$\alpha = 1 \Rightarrow s \leq i$ \hspace{1cm} $\alpha = 2 \Rightarrow s > i$

max $\mathcal{H} \Rightarrow$ Optimal Search

any program constraints

How to maximize $\mathcal{H}$?
Overall Approach [CSF 2017]

\[ P(s, i) \]
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \]
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \]
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \rightarrow \text{SYMBOLIC EXECUTION} \]
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \rightarrow \text{SYMBOLIC EXECUTION} \]

\[ \{ \phi_j(s, \vec{i}) \} \]
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \rightarrow \text{SYMBOLIC EXECUTION} \rightarrow \{ \phi_j(s, \vec{i}) \} \]

partition constraints
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \rightarrow \text{SYMBOLIC EXECUTION} \]

\[ \{ \phi_j (s, \vec{i}) \} \rightarrow \text{MODEL COUNTER} \]
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \rightarrow \text{SYMBOLIC EXECUTION} \]

\[ \{ \phi_j(s, \vec{i}) \} \rightarrow \text{MODEL COUNTER} \]

\[ \{ f_j(\vec{i}) \} \]

partition constraints
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \rightarrow \text{SYMBOLIC EXECUTION} \]

\[ \{\phi_j(s, \vec{i})\} \rightarrow \text{MODEL COUNTER} \]

\[ \{f_j(\vec{i})\} \]
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \rightarrow \text{SYMBOLIC EXECUTION} \]

\[ \{ \phi_j(s, \vec{i}) \} \rightarrow \text{MODEL COUNTER} \]

\[ \{ f_j(\vec{i}) \} \rightarrow \text{ENTROPY} \]
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \rightarrow \text{SYMBOLIC EXECUTION} \]

\[ \{\phi_j(s, \vec{i})\} \rightarrow \text{MODEL COUNTER} \]

\[ \{f_j(\vec{i})\} \rightarrow \text{ENTROPY} \]

\[ \mathcal{H}(\vec{i}) \]

partition constraints

partition cell sizes

objective function
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \rightarrow \text{SYMBOLIC EXECUTION} \]

\[ \{ \phi_j(s, \vec{i}) \} \rightarrow \text{MODEL COUNTER} \]

\[ \{ f_j(\vec{i}) \} \rightarrow \text{ENTROPY} \]

\[ H(\vec{i}) \rightarrow \text{MAXIMIZE} \]

- Partition constraints
- Partition cell sizes
- Objective function
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \rightarrow \text{SYMBOIC EXECUTION} \]
\[ \{\phi_j(s, \vec{i})\} \rightarrow \text{MODEL COUNTER} \]
\[ \{f_j(\vec{i})\} \rightarrow \text{ENTROPY} \]
\[ \mathcal{H}(\vec{i}) \rightarrow \text{MAXIMIZE} \]

- Partition constraints
- Partition cell sizes
- Objective function
- Optimal inputs
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \xrightarrow{\text{Symbolic Execution}} \{\phi_j(s, \vec{i})\} \xrightarrow{\text{Model Counter}} \{f_j(\vec{i})\} \xrightarrow{\text{Entropy}} \mathcal{H}(\vec{i}) \xrightarrow{\text{Maximize}} \vec{i}^* \Rightarrow T^* \]
Overall Approach [CSF 2017]

\[ P(s, \vec{i}) \xrightarrow{\text{Symbolic Execution}} \{\phi_j(s, \vec{i})\} \xrightarrow{\text{Model Counter}} \{f_j(\vec{i})\} \xrightarrow{\text{Entropy}} \mathcal{H}(\vec{i}) \xrightarrow{\text{Maximize}} \vec{i}^* \Rightarrow T^* \]

- \( P(s, \vec{i}) \): partition constraints
- \( \{\phi_j(s, \vec{i})\} \): symbolic execution
- \( \{f_j(\vec{i})\} \): model counter
- \( \mathcal{H}(\vec{i}) \): entropy
- \( \vec{i}^* \Rightarrow T^* \): optimal inputs, optimal tree
Symbolic Execution
Symbolic Execution

- Execute program on **symbolic** rather than concrete inputs.
Symbolic Execution

- Execute program on **symbolic** rather than concrete inputs.
- Maintain **path constraints**, PCs, $\phi_j$ over symbolic inputs.
Symbolic Execution

- Execute program on **symbolic** rather than concrete inputs.
- Maintain **path constraints**, PCs, $\phi_j$ over symbolic inputs.
- For branch instructions:
Symbolic Execution

- Execute program on **symbolic** rather than concrete inputs.
- Maintain **path constraints**, PCs, $\phi_j$ over symbolic inputs.
- For branch instructions:
  
  ```
  if(c) then s1; else s2;
  ```
Symbolic Execution

- Execute program on **symbolic** rather than concrete inputs.
- Maintain **path constraints**, PCs, $\phi_j$ over symbolic inputs.
- For branch instructions:
  
  ```
  if(c) then s1; else s2;
  ```
Symbolic Execution

- Execute program on **symbolic** rather than concrete inputs.
- Maintain **path constraints**, PCs, $\phi_j$ over symbolic inputs.
- For branch instructions:
  
  \[
  \text{if}(c) \text{ then } s_1; \text{ else } s_2;
  \]
Symbolic Execution

- Execute program on **symbolic** rather than concrete inputs.
- Maintain **path constraints**, PCs, $\phi_j$ over symbolic inputs.
- For branch instructions:
  
  $$\text{if}(c) \text{ then } s1; \text{ else } s2;$$

\[
\phi \leftarrow \phi \land c
\]
Symbolic Execution

- Execute program on **symbolic** rather than concrete inputs.
- Maintain **path constraints**, PCs, $\phi_j$ over symbolic inputs.
- For branch instructions:
  
  \[
  \text{if}(c) \text{ then } s_1; \text{ else } s_2;
  \]

  \[
  \phi \leftarrow \phi \land c \quad \phi \leftarrow \phi \land \neg c
  \]
Symbolic Execution

- Execute program on **symbolic** rather than concrete inputs.
- Maintain **path constraints**, PCs, $\phi_j$ over symbolic inputs.
- For branch instructions:
  
  ```
  if(c) then s1; else s2;
  ```
  
  $\phi \leftarrow \phi \land c$
  $\phi \leftarrow \phi \land \neg c$

- Check satisfiability of $\phi$ using constraint solvers (like Z3).
Symbolic Execution

- Execute program on **symbolic** rather than concrete inputs.
- Maintain **path constraints**, PCs, $\phi_j$ over symbolic inputs.
- For branch instructions:
  \[
  \text{if}(c) \text{ then } s1; \text{ else } s2;
  \]

$$
\begin{align*}
\phi & \leftarrow \phi \land c \\
\phi & \leftarrow \phi \land \neg c
\end{align*}
$$

- Check satisfiability of $\phi$ using constraint solvers (like Z3).
- Maintain **cost model** for every path constraint.
Symbolic Attack Tree via Symbolic Execution
Symbolic Attack Tree via Symbolic Execution

\[ o = 1 \implies s \leq i \]
\[ o = 2 \implies s > i \]
Symbolic Attack Tree via Symbolic Execution

\[ o = 1 \Rightarrow s \leq i \quad o = 2 \Rightarrow s > i \]

\[ i = i_0 \quad \text{??} \]
Symbolic Attack Tree via Symbolic Execution

\[ o = 1 \implies s \leq i \quad \text{and} \quad o = 2 \implies s > i \]

\[ i = i_0 \]
Symbolic Attack Tree via Symbolic Execution

\[ o = 1 \implies s \leq i \quad \text{cost: } 1 \]
\[ i = i_0 \]
\[ s \leq i_0 \]

\[ o = 2 \implies s > i \]

\[ o = 2 \implies s > i \]
Symbolic Attack Tree via Symbolic Execution

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

\[ o = 2 \implies s > i \]

\[ \text{cost: 1} \]
\[ s \leq i_0 \]

\[ i = i_0 \]
Symbolic Attack Tree via Symbolic Execution

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

\[ o = 2 \implies s > i \]

\[ i = i_0 \]

\[ s \leq i_0 \]

\[ s > i_0 \]

Cost: 1

Cost: 2
Symbolic Attack Tree via Symbolic Execution

\[ o = 1 \implies s \leq i \]
\[ o = 2 \implies s > i \]

\[ i = i_0 \]

\[
\begin{align*}
\text{cost: 1} & \\
\text{s } \leq i_0 & \\
\text{cost: 2} & \\
\text{s } > i_0 &
\end{align*}
\]
Symbolic Attack Tree via Symbolic Execution

\[ o = 1 \implies s \leq i \]
\[ o = 2 \implies s > i \]

\[ i = i_0 \]

- Cost: 1
- \( s \leq i_0 \)
- \( i = i_1 \)

- Cost: 2
- \( s > i_0 \)
Symbolic Attack Tree via Symbolic Execution

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

\[ o = 2 \Rightarrow s > i \]

\[
\begin{align*}
\text{cost: 1} & \quad s \leq i_0 \\
\text{cost: 2} & \quad s > i_0
\end{align*}
\]
Symbolic Attack Tree via Symbolic Execution

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

\[
o = 2 \Rightarrow s > i
\]
Symbolic Attack Tree via Symbolic Execution

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

\[ o = 2 \implies s > i \]

Cost: 1
\[ s \leq i_0 \]

Cost: 2
\[ s > i_0 \]

Cost: 2
\[ s \leq i_0 \]
\[ s > i_1 \]
Symbolic Attack Tree via Symbolic Execution

\[ o = 1 \implies s \leq i \quad o = 2 \implies s > i \]

- Cost: 1
  - \( s \leq i_0 \)
  - \( i = i_0 \)
  - Cost: 1
    - \( s \leq i_0 \)
    - \( s \leq i_1 \)
  - Cost: 2
    - \( s > i_0 \)
    - \( s > i_1 \)

- Cost: 2
  - \( s > i_0 \)
  - \( i = i_1 \)
  - Cost: 1
    - \( s \leq i_0 \)
    - \( s > i_1 \)
  - Cost: 2
    - \( s > i_0 \)
    - \( s > i_2 \)

- Cost: 1
  - \( s \leq i_0 \)
  - \( s \geq i_2 \)
  - Cost: 2
    - \( s > i_0 \)
    - \( s > i_2 \)

- Cost: 1
  - \( s \geq i_0 \)
  - \( s \geq i_1 \)
  - Cost: 2
    - \( s > i_0 \)
    - \( s > i_2 \)
Symbolic Attack Tree via Symbolic Execution

\[ o = 1 \implies s \leq i \quad \quad o = 2 \implies s > i \]

\[ i = i_0 \]

- Cost: 1
  - \[ s \leq i_0 \]
  - \[ s \leq i_1 \]

- Cost: 2
  - \[ s > i_0 \]
  - \[ s > i_1 \]

\[ i = i_1 \]

\[ i = i_2 \]

- Cost: 2
  - \[ s > i_0 \]
  - \[ s > i_2 \]
Symbolic Attack Tree via Symbolic Execution

\[ o = 1 \Rightarrow s \leq i \]
\[ o = 2 \Rightarrow s > i \]

Set of leaf constraints define a **symbolic partition**.

- \( i = i_0 \) (cost: 1)
  - \( s \leq i_0 \) (cost: 1)
  - \( s \leq i_1 \) (cost: 1)
  - \( i = i_1 \) (cost: 2)
  - \( s > i_0 \) (cost: 2)

- \( i = i_2 \) (cost: 2)
  - \( s > i_0 \) (cost: 2)
  - \( s > i_2 \) (cost: 2)
Overall Approach

\[ P(s, \vec{i}) \rightarrow \text{SYMBOLIC EXECUTION} \]

\( \{ \phi_j(s, \vec{i}) \} \rightarrow \text{MODEL COUNTER} \)

\( \{ f_j(\vec{i}) \} \rightarrow \text{ENTROPY} \)

\( \mathcal{H}(\vec{i}) \rightarrow \text{MAXIMIZE} \)

\( \vec{i}^* \Rightarrow T^* \)

partition constraints

partition cell sizes

objective function

optimal inputs

optimal tree
$P(s, \vec{i}) \rightarrow$ **SYMBOLIC EXECUTION**

$\{\phi_j(s, \vec{i})\} \rightarrow$ **MODEL COUNTER**

$\{f_j(\vec{i})\} \rightarrow$ **ENTROPY**

$\mathcal{H}(\vec{i}) \rightarrow$ **MAXIMIZE**

$\vec{i}^* \Rightarrow T^*$

**Overall Approach**

Partition constraints

Partition cell sizes

Objective function

Optimal inputs

Optimal tree
Overall Approach

\[ P(s, \vec{i}) \rightarrow \text{Symbolic Execution} \rightarrow \{ \phi_j(s, \vec{i}) \} \rightarrow \text{Model Counter} \rightarrow \{ f_j(\vec{i}) \} \rightarrow \text{Entropy} \rightarrow \mathcal{H}(\vec{i}) \rightarrow \text{Maximize} \rightarrow i^* \Rightarrow T^* \]

- Partition constraints
- Partition cell sizes
- Objective function
- Optimal inputs
- Optimal tree
Partition Cell Sizes via Symbolic Model Counting
Partition Cell Sizes via Symbolic Model Counting

\[ \{ \phi_j(s, \vec{i}) \} \xrightarrow{\text{MODEL COUNTER}} \{ f_j(\vec{i}) \} \]
Partition Cell Sizes via Symbolic Model Counting

\[ \{ \phi_j(s, \vec{i}) \} \xrightarrow{\text{Model Counter}} \{ f_j(\vec{i}) \} \]

partition constraints \( \{ \phi_j(s, \vec{i}) \} \) \( \rightarrow \) \( \{ f_j(\vec{i}) \} \)

partition cell sizes
Partition Cell Sizes via Symbolic Model Counting

\[
\{ \phi_j (s, \vec{i}) \} \xrightarrow{\text{Model Counter}} \{ f_j (\vec{i}) \}
\]

\( f_j (\vec{i}) \): size of partition cell \( j \)
Partition Cell Sizes via Symbolic Model Counting

\[ \{ \phi_j (s, \vec{i}) \} \xrightarrow{\text{Model Counter}} \{ f_j (\vec{i}) \} \]

\( f_j (\vec{i}) : \) size of partition cell \( j \)

\[ |\text{partition cell } j| = \# \text{satisfying solutions (models) for } \phi (s, \vec{i}) \]
Partition Cell Sizes via Symbolic Model Counting

\[ \{ \phi_j(s, \vec{i}) \} \xrightarrow{\text{Model Counter}} \{ f_j(\vec{i}) \} \]

\( f_j(\vec{i}) \): size of partition cell \( j \)

\(|\text{partition cell } j| = \# \text{satisfying solutions (models) for } \phi(s, \vec{i})\)

Model Counting Constraint Solvers:

Barvinok: Linear Integer Arithmetic
Partition Cell Sizes via Symbolic Model Counting

\[ \{ \phi_j (s, \vec{i}) \} \rightarrow \text{Model Counter} \rightarrow \{ f_j (\vec{i}) \} \]

\( f_j (\vec{i}) \): size of partition cell \( j \)

\[ |\text{partition cell } j| = \# \text{ satisfying solutions (models) for } \phi (s, \vec{i}) \]

Model Counting Constraint Solvers:

- Barvinok: Linear Integer Arithmetic
- ABC: Linear Integer Arithmetic + Strings [CAV ’15]
Partition Cell Sizes via Symbolic Model Counting

\[
\{ \phi_j (s, \vec{i}) \} \xrightarrow{\text{Model Counter}} \{ f_j (\vec{i}) \}
\]

\( f_j (\vec{i}) \): size of partition cell \( j \)

\[
|\text{partition cell } j| = \# \text{ satisfying solutions (models) for } \phi (s, \vec{i})
\]

Model Counting Constraint Solvers:

- Barvinok: Linear Integer Arithmetic
- ABC: Linear Integer Arithmetic + Strings [CAV ’15]
- SMC: Strings
- LattE: Linear Integer Arithmetic
Partition Cell Sizes via Symbolic Model Counting

\[ \{ \phi_j(s, \vec{i}) \} \xrightarrow{\text{Model Counter}} \{ f_j(\vec{i}) \} \]

\( f_j(\vec{i}) \): size of partition cell \( j \)

\[ |\text{partition cell } j| = \# \text{satisfying solutions (models) for } \phi(s, \vec{i}) \]

Model Counting Constraint Solvers:

- Barvinok: Linear Integer Arithmetic
- ABC: Linear Integer Arithmetic + Strings [CAV ’15]
- SMC: Strings
- LattE: Linear Integer Arithmetic
Partition Cell Sizes via Symbolic Model Counting
Partition Cell Sizes via Symbolic Model Counting

\[
\begin{array}{|c|}
\hline
\text{cost: 1} \\
\text{s} \leq i_0 \\
\text{s} \leq i_1 \\
\hline
\end{array}
\begin{array}{|c|}
\hline
\text{cost: 2} \\
\text{s} \leq i_0 \\
\text{s} > i_1 \\
\hline
\end{array}
\begin{array}{|c|}
\hline
\text{cost: 1} \\
\text{s} > i_0 \\
\text{s} \leq i_2 \\
\hline
\end{array}
\begin{array}{|c|}
\hline
\text{cost: 2} \\
\text{s} > i_0 \\
\text{s} > i_2 \\
\hline
\end{array}
\]
Partition Cell Sizes via Symbolic Model Counting

\[
\begin{align*}
\text{cost: 1} & \quad s \leq i_0 \\
& \quad s \leq i_1
\end{align*}
\]

\[
\begin{align*}
\text{cost: 2} & \quad s \leq i_0 \\
& \quad s > i_1
\end{align*}
\]

\[
\begin{align*}
\text{cost: 1} & \quad s > i_0 \\
& \quad s \leq i_2
\end{align*}
\]

\[
\begin{align*}
\text{cost: 2} & \quad s > i_0 \\
& \quad s \geq i_2
\end{align*}
\]
Partition Cell Sizes via Symbolic Model Counting

\[ s \leq i_0 \]
\[ s > i_1 \]
Partition Cell Sizes via Symbolic Model Counting

\[
\begin{align*}
& s \leq i_0 \\
& s > i_1
\end{align*}
\]

\[\text{Barvinok}(\phi_j(s, \vec{i}), \vec{i}^\prime): \text{ piecewise polynomial function } f_j(\vec{i})\]
Partition Cell Sizes via Symbolic Model Counting

Barvinok($\phi_j(s, \vec{i}), \vec{i}$): piecewise polynomial function $f_j(\vec{i})$
Partition Cell Sizes via Symbolic Model Counting

\[ s \leq i_0 \]
\[ s > i_1 \]

Barvinok(\( \phi_j (s, \vec{i}) \), \( \vec{i} \)): piecewise polynomial function \( f_j(\vec{i}) \)

\[
f (i_0, i_1) = \begin{cases} 
  i_0 - i_1 & \text{if } 1 \leq i_1 \leq i_0 \leq 8 \\
  i_0 & \text{if } i_1 < 1 \leq i_0 \leq 8 \\
  8 - i_1 & \text{if } 1 \leq i_1 \leq 8 \leq i_0 \\
  8 & \text{if } i_1 \leq 1 < 8 \leq i_0 \\
  0 & \text{otherwise}
\end{cases}
\]
Barvinok($\phi_j(s,i),\vec{i}$): piecewise polynomial function $f_j(\vec{i})$

$$f(i_0, i_1) = \begin{cases} 
  i_0 - i_1 & \text{if } 1 \leq i_1 \leq i_0 \leq 8 \\
  i_0 & \text{if } i_1 < 1 \leq i_0 \leq 8 \\
  8 - i_1 & \text{if } 1 \leq i_1 \leq 8 \leq i_0 \\
  8 & \text{if } i_1 \leq 1 < 8 \leq i_0 \\
  0 & \text{otherwise}
\end{cases}$$

$$f(6, 2) = 4$$
Partition Cell Sizes via Symbolic Model Counting

Barvinok($\phi_j(s, \vec{i})$, $\vec{i}$): piecewise polynomial function $f_j(\vec{i})$

$$f(i_0, i_1) = \begin{cases} 
  i_0 - i_1 & \text{if } 1 \leq i_1 \leq i_0 \leq 8 \\
  i_0 & \text{if } i_1 < 1 \leq i_0 \leq 8 \\
  8 - i_1 & \text{if } 1 \leq i_1 \leq 8 \leq i_0 \\
  8 & \text{if } i_1 \leq 1 < 8 \leq i_0 \\
  0 & \text{otherwise}
\end{cases}$$

$f(6, 2) = 4$  $f(5, -1) = 5$
Partition Cell Sizes via Symbolic Model Counting

$$\begin{align*}
  s &\leq i_0 \\
  s &> i_1
\end{align*}$$

Barvinok($\phi_j(s, \vec{i}), \vec{i}$): piecewise polynomial function $f_j(\vec{i})$

$$f(i_0, i_1) = \begin{cases} 
  i_0 - i_1 & \text{if } 1 \leq i_1 \leq i_0 \leq 8 \\
  i_0 & \text{if } i_1 < 1 \leq i_0 \leq 8 \\
  8 - i_1 & \text{if } 1 \leq i_1 \leq 8 \leq i_0 \\
  8 & \text{if } i_1 \leq 1 < 8 \leq i_0 \\
  0 & \text{otherwise}
\end{cases}$$

$$f(6, 2) = 4 \quad f(5, -1) = 5 \quad f(3, 7) = 0$$
\[ P(s, \vec{i}) \xrightarrow{\text{Symbolic Execution}} \{ \phi_j(s, \vec{i}) \} \xrightarrow{\text{Model Counter}} \{ f_j(\vec{i}) \} \xrightarrow{\text{Entropy}} \mathcal{H}(\vec{i}) \xrightarrow{\text{Maximize}} \vec{i}^* \Rightarrow T^* \]

- **Partition Constraints**: \( \phi_j(s, \vec{i}) \)
- **Partition Cell Sizes**: \( f_j(\vec{i}) \)
- **Objective Function**: \( \mathcal{H}(\vec{i}) \)
- **Optimal Inputs**: \( \vec{i}^* \)
- **Optimal Tree**: \( T^* \)
\( P(s, \vec{i}) \) → \text{SYMBOLIC EXECUTION} \{\phi_j(s, \vec{i})\} → \text{MODEL COUNTER} \{f_j(\vec{i})\} → \text{ENTROPY} \mathcal{H}(\vec{i}) → \text{MAXIMIZE} \vec{i}^* \Rightarrow T^*

- **Partition**
  - **Constraints**
- **Cell sizes**
- **Objective function**
- **Optimal inputs**
- **Optimal tree**
Computing Multi-Step Entropy Symbolically

- $i = i_0$
  - cost: 1
    - $s \leq i_0$
      - $i = i_1$
        - cost: 1
          - $s \leq i_0$
          - $s \leq i_1$
        - cost: 2
          - $s > i_0$
          - $s \leq i_1$
      - cost: 2
        - $s > i_0$
        - $s > i_1$
  - cost: 2
    - $s > i_0$
      - $i = i_2$
        - cost: 1
          - $s > i_0$
          - $s \leq i_2$
        - cost: 2
          - $s > i_0$
          - $s > i_2$
Computing Multi-Step Entropy Symbolically

- **Cost:** 1
  - **Condition:** \( s \leq i_0 \)
  - **Next State:** \( i = i_1 \)

- **Cost:** 2
  - **Condition:** \( s > i_0 \)
  - **Next State:** \( i = i_2 \)

- **Cost:** 1
  - **Condition:** \( s \leq i_1 \)

- **Cost:** 2
  - **Condition:** \( s > i_1 \)

- **Cost:** 1
  - **Condition:** \( s \leq i_2 \)

- **Cost:** 2
  - **Condition:** \( s > i_2 \)
### Computing Multi-Step Entropy Symbolically

<table>
<thead>
<tr>
<th>Condition</th>
<th>Symbol</th>
<th>Condition</th>
<th>Symbol</th>
<th>Condition</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s \leq i_0$</td>
<td>s</td>
<td>$s \leq i_0$</td>
<td>s</td>
<td>$s &gt; i_0$</td>
<td>s i_2</td>
</tr>
<tr>
<td>$s \leq i_1$</td>
<td>s</td>
<td>$s &gt; i_1$</td>
<td>s</td>
<td>$s \leq i_2$</td>
<td>s i_2</td>
</tr>
</tbody>
</table>
Computing Multi-Step Entropy Symbolically

\[ s \leq i_0 \]
\[ s \leq i_1 \]

\[ s \leq i_0 \]
\[ s > i_1 \]

\[ s > i_0 \]
\[ s \leq i_2 \]

\[ s > i_0 \]
\[ s > i_2 \]
Computing Multi-Step Entropy Symbolically

\[
f_2(\vec{i}) = \begin{cases} 
  i_0 - i_1 & \text{if } 1 \leq i_1 \leq i_0 \leq 8 \\
  i_0 & \text{if } i_1 < 1 \leq i_0 \leq 8 \\
  8 - i_1 & \text{if } 1 \leq i_1 \leq 8 \leq i_0 \\
  8 & \text{if } i_1 \leq 1 < 8 \leq i_0 \\
  0 & \text{otherwise}
\end{cases}
\]
Computing Multi-Step Entropy Symbolically

\[ s \leq i_0 \]
\[ s \leq i_1 \]

\[ s \leq i_0 \]
\[ s > i_1 \]

\[ s > i_0 \]
\[ s \leq i_2 \]

\[ s > i_0 \]
\[ s i_2 \]

Model Counter

\[ f_2(\vec{i}) \]
Computing Multi-Step Entropy Symbolically

\[ s \leq i_0 \]
\[ s \leq i_1 \]

Model Counter

\[ f_1(\vec{i}) \]

\[ s \leq i_0 \]
\[ s > i_1 \]

Model Counter

\[ f_2(\vec{i}) \]

\[ s > i_0 \]
\[ s \leq i_2 \]

Model Counter

\[ f_3(\vec{i}) \]

\[ s > i_0 \]
\[ s \leq i_2 \]

Model Counter

\[ f_4(\vec{i}) \]
Computing Multi-Step Entropy Symbolically

\[ p(s \in \mathcal{A}) = \frac{f_{j}(\vec{i})}{|S|} \]

\[ p_{j}(\vec{i}) = \frac{f_{j}(\vec{i})}{|S|} \]
Computing Multi-Step Entropy Symbolically

\[
p(s \in \mathcal{A}) = \sum_{j=1}^{n} p_i \log_2 \frac{1}{p_j}
\]

\[
\mathcal{H}(\vec{i}) = \sum_{j=1}^{n} p_i \log_2 \frac{1}{p_j}
\]
Computing Multi-Step Entropy Symbolically

\[ p(s \in A) = \frac{f_j(i)}{|S|} \]

\[ p_j(i) = \frac{f_j(i)}{|S|} \]

\[ H(i) = \sum_{j=1}^{n} p_i \log_2 \frac{1}{p_j} = \frac{f_1(i)}{8} \log_2 \frac{8}{f_1(i)} + \frac{f_2(i)}{8} \log_2 \frac{8}{f_2(i)} + \frac{f_3(i)}{8} \log_2 \frac{8}{f_3(i)} + \frac{f_4(i)}{8} \log_2 \frac{8}{f_4(i)} \]
\[ P(s, \vec{i}) \rightarrow \text{SYMBOLIC EXECUTION} \]

\[ \{ \phi_j(s, \vec{i}) \} \rightarrow \text{MODEL COUNTER} \]

\[ \{ f_j(\vec{i}) \} \rightarrow \text{ENTROPY} \]

\[ \mathcal{H}(\vec{i}) \rightarrow \text{MAXIMIZE} \]

\[ \vec{i}^* \Rightarrow T^* \]
\( P(s, \vec{i}) \rightarrow \text{Symbolic Execution} \)

\( \{ \phi_j(s, \vec{i}) \} \rightarrow \text{Model Counter} \)

\( \{ f_j(\vec{i}) \} \rightarrow \text{Entropy} \)

\( \mathcal{H}(\vec{i}) \rightarrow \text{Maximize} \)

\( \vec{i}^* \Rightarrow T^* \)
\[ H(\vec{i}) = \sum_{j=1}^{n} p_i \log_2 \frac{1}{p_j} = \frac{f_1(i)}{8} \log_2 \frac{8}{f_1(i)} + \frac{f_2(i)}{8} \log_2 \frac{8}{f_2(i)} + \frac{f_3(i)}{8} \log_2 \frac{8}{f_3(i)} + \frac{f_4(i)}{8} \log_2 \frac{8}{f_4(i)} \]
$$\mathcal{H}(\vec{i}) = \sum_{j=1}^{n} p_i \log_2 \frac{1}{p_j} = \frac{f_1(\vec{i})}{8} \log_2 \frac{8}{f_1(\vec{i})} + \frac{f_2(\vec{i})}{8} \log_2 \frac{8}{f_2(\vec{i})} + \frac{f_3(\vec{i})}{8} \log_2 \frac{8}{f_3(\vec{i})} + \frac{f_4(\vec{i})}{8} \log_2 \frac{8}{f_4(\vec{i})}$$

Objective function $\mathcal{H}(\vec{i}) \xrightarrow{\text{Maximize}}$
Numeric Maximization

\[ H(\vec{i}) = \sum_{j=1}^{n} p_i \log_2 \left( \frac{1}{p_j} \right) = \frac{f_1(i)}{8} \log_2 \frac{8}{f_1(i)} + \frac{f_2(i)}{8} \log_2 \frac{8}{f_2(i)} + \frac{f_3(i)}{8} \log_2 \frac{8}{f_3(i)} + \frac{f_4(i)}{8} \log_2 \frac{8}{f_4(i)} \]

Objective function \( H(\vec{i}) \) → \text{Maximize} → (Mathematica)
**Numeric Maximization**

\[
\mathcal{H}(\vec{i}) = \sum_{j=1}^{n} p_i \log_2 \left( \frac{1}{p_j} \right) = \frac{f_1(i)}{8} \log_2 \frac{8}{f_1(i)} + \frac{f_2(i)}{8} \log_2 \frac{8}{f_2(i)} + \frac{f_3(i)}{8} \log_2 \frac{8}{f_3(i)} + \frac{f_4(i)}{8} \log_2 \frac{8}{f_4(i)}
\]

Objective function \( \mathcal{H}(\vec{i}) \) \rightarrow \text{MAXIMIZE} \rightarrow \vec{i}^* \Rightarrow T^*

Optimal inputs optimal tree
\[ H(\vec{i}) = \sum_{j=1}^{n} p_i \log_2 \frac{1}{p_j} = \frac{f_1(i)}{8} \log_2 \frac{8}{f_1(i)} + \frac{f_2(i)}{8} \log_2 \frac{8}{f_2(i)} + \frac{f_3(i)}{8} \log_2 \frac{8}{f_3(i)} + \frac{f_4(i)}{8} \log_2 \frac{8}{f_4(i)} \]

objective function

\[ H(\vec{i}) \xrightarrow{\text{MAXIMIZE}} \vec{i}^* \Rightarrow \vec{T}^* \]

(optimal tree)

\[ \vec{i}^* = (i_0^*, i_1^*, i_2^*) \]
Numeric Maximization

\[ H(\vec{i}) = \sum_{j=1}^{n} p_i \log_2 \frac{1}{p_j} = \frac{f_1(\vec{i})}{8} \log_2 \frac{8}{f_1(\vec{i})} + \frac{f_2(\vec{i})}{8} \log_2 \frac{8}{f_2(\vec{i})} + \frac{f_3(\vec{i})}{8} \log_2 \frac{8}{f_3(\vec{i})} + \frac{f_4(\vec{i})}{8} \log_2 \frac{8}{f_4(\vec{i})} \]

Objective function \( H(\vec{i}) \) → MAXIMIZE \( \vec{i}^* \) ⇒ \( T^* \)

Optimal inputs

\[ \vec{i}^* = (i_0^*, i_1^*, i_2^*) \]

\[ \vec{i}^* = (4, 2, 6) \]
Numeric Maximization

\[ H(\vec{i}) = \sum_{j=1}^{n} p_i \log_2 \frac{1}{p_j} = \frac{f_1(\vec{i})}{8} \log_2 \frac{8}{f_1(\vec{i})} + \frac{f_2(\vec{i})}{8} \log_2 \frac{8}{f_2(\vec{i})} + \frac{f_3(\vec{i})}{8} \log_2 \frac{8}{f_3(\vec{i})} + \frac{f_4(\vec{i})}{8} \log_2 \frac{8}{f_4(\vec{i})} \]

Objective function: \( H(\vec{i}) \) → MAXIMIZE \( \vec{i}^* \Rightarrow T^* \) (MATHEMATICA)

\[ \vec{i}^* = (i_0^*, i_1^*, i_2^*) \]

\[ \vec{i}^* = (4, 2, 6) \]

Optimal inputs: \( i = i_0 \)

Optimal tree:

- Cost: 1
  - \( i = i_1 \)
    - Cost: 1
    - \( i = i_0 \)
  - Cost: 2
    - \( i = i_2 \)
      - Cost: 1
Numeric Maximization

\[ \mathcal{H}(\vec{i}) = \sum_{j=1}^{n} p_i \log_2 \frac{1}{p_j} = \frac{f_1(\vec{i})}{8} \log_2 \frac{8}{f_1(\vec{i})} + \frac{f_2(\vec{i})}{8} \log_2 \frac{8}{f_2(\vec{i})} + \frac{f_3(\vec{i})}{8} \log_2 \frac{8}{f_3(\vec{i})} + \frac{f_4(\vec{i})}{8} \log_2 \frac{8}{f_4(\vec{i})} \]

Objective function \( \mathcal{H}(\vec{i}) \) \( \xrightarrow{\text{MAXIMIZE}} \) \( \vec{i}^* \Rightarrow T^* \)

Optimal inputs optimal tree

\[ \vec{i}^* = (i_0^*, i_1^*, i_2^*) \]

\[ \vec{i}^* = (4, 2, 6) \]

\[ \begin{align*}
\text{cost: 1} & \\
i & = 2 \\
\text{cost: 1} & \\
i & = 4 \\
\text{cost: 1} & \\
i & = 6 \\
\text{cost: 2} &
\end{align*} \]
\[ P(s, \vec{i}) \rightarrow \text{Symbolic Execution} \]

partition constraints \[ \{ \phi_j(s, \vec{i}) \} \rightarrow \text{Model Counter} \]

partition cell sizes \[ \{ f_j(\vec{i}) \} \rightarrow \text{Entropy} \]

objective function \[ \mathcal{H}(\vec{i}) \rightarrow \text{Maximize} \]

optimal inputs \[ \vec{i}^* \Rightarrow T^* \]
Other Methods
Other Methods

My approach: reduce attack synthesis to numeric optimization problem.
Other Methods

**My approach:** reduce attack synthesis to **numeric optimization** problem.

- Limited by model counter.
- Numeric optimum not guaranteed.
Other Methods

My approach: reduce attack synthesis to **numeric optimization** problem.

- Limited by model counter.
- Numeric optimum not guaranteed.

**MaxSMT**: reduce attack synthesis to **Max SAT** problem.
Other Methods

**My approach:** reduce attack synthesis to **numeric optimization** problem.

- Limited by model counter.
- Numeric optimum not guaranteed.

**MaxSMT:** reduce attack synthesis to **Max SAT** problem.

- Reduces everything to bits.
- Upper bound on information leakage.
Other Methods

**My approach:** reduce attack synthesis to **numeric optimization** problem.

- Limited by model counter.
- Numeric optimum not guaranteed.

**MaxSMT:** reduce attack synthesis to **Max SAT** problem.

- Reduces everything to bits.
- Upper bound on information leakage.

**MARCO:** reduce attack synthesis to **Maximum SAT Subsets** problem.

- Reduces everything to bits.
- Exact optimal information leakage guaranteed.
Case Study: LawDB

From DARPA Space-Time Analysis for Cybersecurity (STAC)

Server

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

Search midID maxID

Client

List of employees.

\[ ID_{res} \in [\text{minID}, \text{maxID}] \]
LawDB Partition Constraints

\[ \{ h \leq l_1 \land 85 \leq l_1 \land 64 \leq l_1 \land 64 \geq l_2 \land h > 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    h > l_1 \land 85 \leq l_1 \land 64 \leq l_1 \land 64 \geq l_2 \land h > 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    85 > l_1 \land 64 \leq l_1 \land 64 \geq l_2 \land h > 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    64 > l_1 \land 64 \geq l_2 \land h > 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    h \leq l_1 \land 85 \leq l_1 \land 85 \geq l_2 \land 64 \leq l_2 \land h > 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    h > l_1 \land 85 \leq l_1 \land 85 \geq l_2 \land 64 \leq l_2 \land h > 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    85 > l_1 \land 85 \geq l_2 \land 64 < l_2 \land h > 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    h \leq l_1 \land h \geq l_2 \land 85 \leq l_2 \land 64 \leq l_2 \land h > 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    h > l_1 \land h \geq l_2 \land 85 \leq l_2 \land 64 \leq l_2 \land h > 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    h < l_2 \land 85 < l_2 \land 64 < l_2 \land h > 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    85 \leq l_1 \land h \leq l_1 \land 64 \leq l_1 \land 64 \geq l_2 \land h \leq 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    85 > l_1 \land h \leq l_1 \land 64 \leq l_1 \land 64 \geq l_2 \land h \leq 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    h > l_1 \land 64 \leq l_1 \land 64 \geq l_2 \land h \leq 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    64 > l_1 \land 64 \geq l_2 \land h \leq 85 \land h > 64 \land h \neq 85 \land h \neq 64, \\
    85 \leq l_1 \land h \leq l_1 \land h \geq l_2 \land 64 < l_2 \land h \leq 85 \land h > 64 \land h \neq 85 \land h \neq 64 \} \]
Case Study: LawDB, DB size = 100

Keep pushing tree deeper until partitions have size 1.

<table>
<thead>
<tr>
<th>Tree depth</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>7</td>
</tr>
<tr>
<td>MaxSMT</td>
<td>17</td>
</tr>
<tr>
<td>MARCO</td>
<td>7</td>
</tr>
</tbody>
</table>
Case Study: LawDB, DB size = 100

Keep pushing tree deeper until partitions have size 1.

<table>
<thead>
<tr>
<th></th>
<th>Tree depth</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>7</td>
<td>57s</td>
</tr>
<tr>
<td>MaxSMT</td>
<td>17</td>
<td>21s</td>
</tr>
<tr>
<td>MARCO</td>
<td>7</td>
<td>2m 36s</td>
</tr>
</tbody>
</table>
Case Study: LawDB, DB size = 100

Keep pushing tree deeper until partitions have size 1.

<table>
<thead>
<tr>
<th></th>
<th>Tree depth</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>7</td>
<td>57s</td>
</tr>
<tr>
<td>MaxSMT</td>
<td>17</td>
<td>21s</td>
</tr>
<tr>
<td>MARCO</td>
<td>7</td>
<td>2m 36s</td>
</tr>
</tbody>
</table>
Proposed Experiments

DARPA Space-Time Analysis for Cybersecurity (STAC)
Canonical Side-Channel Vulnerability Benchmark

https://github.com/Apogee-Research/STAC/

7 Applications, 1 to 3 variants each
14 total programs
Challenges and Solutions

Fully Offline Static

Quantify over all $s \in S$

Exponential blowup

Cost model: overly ideal, not realistic

Ignores HW / OS properties
Challenges and Solutions

**Fully Offline Static**
- Quantify over all $s \in S$
- Exponential blowup
- Cost model: overly ideal, not realistic
- Ignores HW / OS properties

**Static + Dynamic**
- Real system has one $s \in S$
Challenges and Solutions

**Fully Offline Static**
- Exponential blowup
- Cost model: overly ideal, not realistic
- Ignores HW / OS properties

**Static + Dynamic**
- Real system has one $s \in S$
Challenges and Solutions

Fully Offline Static

- Exponential blowup
- Cost model: overly ideal, not realistic
- Ignores HW / OS properties

Quantify over all $s \in S$

Static + Dynamic

- Real system has one $s \in S$
- Put program on a real system
- Dynamic cost profiling
Online Attack Synthesis
Online Attack Synthesis Proposed Work
private s = getMaxBytes();

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

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

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

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

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

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

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

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

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

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

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

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

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

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

public int compare(int i){
    if(s <= i)
        some computation; // 1 s
    else
        log.write("too many bits"); // 2s
    return 0;
}
Challenges: Uncertainty Everywhere
Challenges: Uncertainty Everywhere

Attacker Belief?

$s$?
Challenges: Uncertainty Everywhere

Attacker Belief?

\[ s? \]
Challenges: Uncertainty Everywhere

Attacker Belief? $s$?

Input Choice? $i^*$

$\frac{1}{8}$

1 2 3 4 5 6 7 8
Challenges: Uncertainty Everywhere

Attacker Belief?  S?

Input Choice?  i*

Observation noise?  s ≤ i, s > i
Challenges: Uncertainty Everywhere

Attacker Belief? $s$?

Input Choice? $i^* = 5$

Observation noise? $s \leq 5$, $s > 5$
Challenges: Uncertainty Everywhere

Attacker Belief?

Input Choice?

Observation noise?

\[
i^* = 5
\]

\[
s \leq 5 \quad s > 5
\]

\[
t = 4.12
\]
Challenges: Uncertainty Everywhere

Attacker Belief? $s$?

Input Choice? $i^* = 5$

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

$t = 4.12$
Challenges: Uncertainty Everywhere

Attacker Belief?

Input Choice?

Observation noise?

\[ i^* = 5 \]

\[ s \leq 5 \quad s > 5 \]
Challenges: Uncertainty Everywhere

Attacker Belief? s?

Input Choice? $i^* = 5$

Observation noise? $s \leq 5$ $s > 5$
Challenges: Uncertainty Everywhere

Attacker Belief? $s$?

Input Choice? $i^* = 5$

Observation noise?

$$s \leq 5$$

$$s > 5$$

$\frac{1}{8}$

More likely

less likely

1 2 3 4 5 6 7 8

$t = 2.3$
Challenges: Uncertainty Everywhere

Attacker Belief?

\[ s? \]

Input Choice?

\[ i^* = 5 \]

Observation noise?

\[ s \leq 5 \quad s > 5 \]
Challenges: Uncertainty Everywhere

Attacker Belief?

Input Choice?

\[ i^* = 5 \]

Observation noise?

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

\[ s \leq 5 \quad s > 5 \]
Challenges: Uncertainty Everywhere

Attacker Belief?

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

Observation noise?

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

$s \leq 5\quad s > 5$
Challenges: Uncertainty Everywhere

Attacker Belief?  

\[ s? \]

Input Choice?  

\[ i^* = 5 \]

Observation noise?  

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

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

\[ s \leq 5 \]

\[ s > 5 \]
Challenges: Uncertainty Everywhere

Attacker Belief?

Input Choice?

Observation noise?

\[ s? \quad i^* = 5 \quad s \leq 5 \quad s > 5 \]

\[ p(s|o, i^*) \quad p(o|s, i) \]
Challenges: Uncertainty Everywhere

Attacker Belief?  

$\begin{align*}
\text{Input Choice?} & \quad i^* = 5 \\
\text{Observation noise?} & \\
\end{align*}$

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

$p(o|s, i)$  

$p(o|s, i)$
Challenges: Uncertainty Everywhere

Attacker Belief? $s$?

Input Choice? $i^* = 5$

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

$p(s|o, i^*)$  $p(o|s, i^*)$  $p(o|s, i)$
Challenges: Uncertainty Everywhere

Attacker Belief? $s$?

Input Choice? $i^* = 5$

Observation noise? $s \leq 5$ \quad $s > 5$

$p(s|o, i^*)$ \quad $p(o|s, i^*)$ \quad $p(o|s, i)$
Challenges: Uncertainty Everywhere

Attacker Belief?

Input Choice?

Observation noise?

\[
p(s \mid o, i^*) \quad \text{and} \quad p(s \mid o, i^*) \quad \text{and} \quad p(o \mid s, i)
\]
Challenges: Uncertainty Everywhere

Attacker Belief? $s$?

Input Choice? $i^* = 5$

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

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

Bayes Rule
Challenges: Uncertainty Everywhere

Attacker Belief?

Input Choice?

Observation noise?

\[s?\]

\[i^* = 5\]

\[s \leq 5\]  \hspace{1cm}  \[s > 5\]

\[\frac{1}{8}\]

\[1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8\]

\[p(s|o, i^*)\]  \hspace{1cm}  \[p(s|o, i^*)\]  \hspace{1cm}  \[p(o|s, i)\]

Bayes Rule
Challenges: Uncertainty Everywhere

Attacker Belief?  

Input Choice?  
\[ i^* = 5 \]

Observation noise?  

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

Bayes Rule

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

Model Counting

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

\[ s \leq 5 \]

\[ s > 5 \]
Challenges: Uncertainty Everywhere

Attacker Belief?

\[ s? \]

Input Choice?

\[ i^* = 5 \]

Observation noise?

\[ s \leq 5 \quad s > 5 \]

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

Bayes Rule

Weighted Model Counting
Challenges: Uncertainty Everywhere

Attacker Belief?

\[ s? \]

\[ \begin{align*}
\frac{1}{8} & \quad | \quad 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{align*} \]

Input Choice?

\[ i^* = 5 \]

\[ \max \mathcal{H} \]

Weighted Model Counting

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

Bayes Rule

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

Observation noise?

\[ s \leq 5 \]

\[ s > 5 \]
Proposed Approach
Proposed Approach
1. Offline Static Analysis
Proposed Approach

1. Offline Static Analysis

2. Offline Dynamic Analysis
Proposed Approach

1. Offline Static Analysis

2. Offline Dynamic Analysis

3. Online Attack Synthesis
Proposed Approach

1. Offline Static Analysis
2. Offline Dynamic Analysis
3. Online Attack Synthesis
Proposed Approach
Proposed Approach

\[ P(s, i) \]
Source Code
Proposed Approach

\[ P(s, i) \rightarrow \text{Symbolic Execution} \rightarrow \{ \phi_j(s, i) \} \]

path constraints
Proposed Approach

\[ P(s, i) \]

Source Code

\[ \text{Symbolic Execution} \]

\[ \{ \phi_j(s, i) \} \]

path constraints

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

PC models (witnesses)
Proposed Approach

\[ P(s, i) \xrightarrow{\text{Source Code}} \quad \text{Symbolic Execution} \quad \xrightarrow{\{\phi_j(s, i)\}} \quad \{w_j = (s_j, i_j)\} \quad \text{PC models (witnesses)} \]

Idea: each PC characterizes an observable program behavior
Proposed Approach

\[ P(s, i) \rightarrow \text{SYMBOLIC EXECUTION} \rightarrow \{ \phi_j(s, i) \} \rightarrow \{ w_j = (s_j, i_j) \} \]

path constraints
PC models (witnesses)

Idea: each PC characterizes an observable program behavior

\[(s_j, i_j) \models \phi_j \quad (s'_j, i'_j) \models \phi_j\]
Proposed Approach

Source Code $P(s, i) \rightarrow$ Symbolic Execution $\{\phi_j(s, i)\} \rightarrow \{w_j = (s_j, i_j)\}$

Idea: each PC characterizes an observable program behavior

$$ (s_j, i_j) \models \phi_j \quad (s'_j, i'_j) \models \phi_j $$

$$ P(s_j, i_j) \quad P(s'_j, i'_j) $$
Proposed Approach

\[ P(s, i) \]  \rightarrow \text{Symbolic Execution} \rightarrow \{ \phi_j(s, i) \} \rightarrow \{ w_j = (s_j, i_j) \}

Path constraints

PC models (witnesses)

Idea: each PC characterizes an observable program behavior

\[ (s_j, i_j) \models \phi_j \quad (s'_j, i'_j) \models \phi_j \]

\[ P(s_j, i_j) \quad ? \quad P(s'_j, i'_j) \]
Proposed Approach

\[ P(s, i) \rightarrow \text{Symbolic Execution} \rightarrow \{ \phi_j(s, i) \} \rightarrow \{ w_j = (s, i_j) \} \]

path constraints PC models (witnesses)

Idea: each PC characterizes an observable program behavior

\[(s_j, i_j) \models \phi_j \quad (s'_j, i'_j) \models \phi_j\]

\[P(s_j, i_j) \equiv P(s'_j, i'_j)\]

\[\phi_j(s, i)\] characterizes observationally indistinguishable behaviors

\[P(s_j, i_j)\] is a representative of all behaviors in that class
Proposed Approach

\[ P(s, i) \rightarrow \text{SYMBOLIC EXECUTION} \rightarrow \{ \phi_j(s, i) \} \rightarrow \{ w_j = (s_j, i_j) \} \]

- Source Code
- Path constraints
- PC models (witnesses)
Proposed Approach

\[ P(s, i) \rightarrow \text{Source Code} \rightarrow \text{Symbolic Execution} \rightarrow \{ \phi_j(s, i) \} \rightarrow \{ w_j = (s_j, i_j) \} \]

- Path constraints
- PC models (witnesses)
Proposed Approach

\[ P(s, i) \rightarrow \text{Symbolic Execution} \rightarrow \{ \phi_j(s, i) \} \rightarrow \{ w_j = (s_j, i_j) \} \]

2. Offline Dynamic Analysis

3. Online Attack Synthesis
Proposed Approach

$P(s,i)$
Source Code

Symbolic Execution

$\{\phi_j(s,i)\}$
path constraints

$\{w_j = (s_j, i_j)\}$
PC models (witnesses)

2. Offline Dynamic Analysis

3. Online Attack Synthesis
Proposed Approach

\[ P(s, i) \rightarrow \text{SYMBOLIC EXECUTION} \rightarrow \{ \phi_j(s, i) \} \rightarrow \{ w_j = (s_j, i_j) \} \]

- \( P(s, i) \): Source Code
- \( \text{SYMBOLIC EXECUTION} \)
- \( \{ \phi_j(s, i) \} \): path constraints
- \( \{ w_j = (s_j, i_j) \} \): PC models (witnesses)
Proposed Approach

\[ P(s, i) \]

Source Code \[ \rightarrow \] Symbolic Execution

\[ \\{ \phi_j(s, i) \} \] path constraints

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

Network

\[ P(s, i) \]
HW / OS
Proposed Approach

\[ P(s, i) \rightarrow \text{Symbolic Execution} \rightarrow \{ \phi_j(s, i) \} \rightarrow \{ w_j = (s_j, i_j) \} \]

\{ w_j = (s_i, i_j) \}
Proposed Approach

\[ P(s, i) \]

Source Code

\[ \{ \phi_j (s, i) \} \]

path constraints

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

PC models (witnesses)

\{ w_j = (s_i, i_j) \}

\[ P(s, i) \]

HW / OS

Network
Proposed Approach

\[ P(s, i) \]

Source Code

Symbolic Execution

\[ \{ \phi_j(s, i) \} \]

Path constraints

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

PC models (witnesses)

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

Network

\[ P(s, i) \]

HW / OS
Proposed Approach

\[ P(s, i) \]

Source Code

\[ \{\phi_j(s, i)\} \]

path constraints

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

PC models (witnesses)

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

\[ P(s, i) \]

HW / OS

Network

\[ \{w_j = (s_i, i_j)\} \]
Proposed Approach

\[ P(s, i) \rightarrow \text{Symbolic Execution} \rightarrow \{ \phi_j(s, i) \} \rightarrow \{ w_j = (s_j, i_j) \} \]

- **Source Code**
- **Symbolic Execution**
- **Path constraints**
- **PC models (witnesses)**

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

Network

\[ P(s, i) \rightarrow \text{HW / OS} \]
Proposed Approach

\[ P(s, i) \rightarrow \text{SYMBOLIC EXECUTION} \rightarrow \{\phi_j(s, i)\} \rightarrow \{w_j = (s_j, i_j)\} \]

\[ \times 1000 \]

\[ \{w_j = (s_i, i_j)\} \rightarrow \text{Network} \rightarrow \{o, \ldots\} \]

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

\[ P(s, i) \]
Proposed Approach

Idea: characterize effect of noise on each class of program behaviors using the witness for that behavior.
Proposed Approach

\[ P(s, i) \]  
Source Code

\[ \{ \phi_j(s, i) \} \]  
path constraints

\[ \{ w_j = (s_i, i_j) \} \]  
PC models (witnesses)

\[ \times 1000 \]

\[ \{ w_j = (s_i, i_j) \} \]  
HW / OS

\[ P(s, i) \]  
Network

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

\[ P(s, i) \quad \rightarrow \quad \text{SYMBOLIC EXECUTION} \quad \rightarrow \quad \{ \phi_j(s, i) \} \quad \rightarrow \quad \{ w_j = (s_j, i_j) \} \]

- **Source Code**
- **Symbolic Execution**
- **Path Constraints**
- **PC Models (Witnesses)**

\[ \times 1000 \]

\[ \{ w_j = (s_i, i_j) \} \quad \rightarrow \quad P(s, i) \quad \rightarrow \quad \text{Network} \quad \rightarrow \quad \{ p(o|s_j, i_j) \} \]
Proposed Approach

\[ P(s, i) \]

Source Code

**Symbolic Execution**

\[ \{ \phi_j(s, i) \} \]

Path constraints

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

PC models (witnesses)

\[ \times 1000 \]

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

Network

\[ P(s, i) \]

HW / OS

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

3. Online Attack Synthesis
Proposed Approach

\[ P(s, i) \rightarrow \text{Symbolic Execution} \rightarrow \{ \phi_j(s, i) \} \rightarrow \{ w_j = (s_j, i_j) \} \]

Source Code

\[ \times 1000 \]

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

\[ P(s, i) \rightarrow \text{HW / OS} \rightarrow \{ p(o|s_j, i_j) \} \]

\[ \{ \uparrow, \downarrow, \ldots \} p(o|s_j, i_j) \]
Proposed Approach

\[ P(s, i) \]
Source Code

Symbolic Execution

\{ \phi_j(s, i) \}
path constraints

\{ w_j = (s_j, i_j) \}
PC models (witneses)

\times 1000

\{ w_j = (s_i, i_j) \}

Network

\[ P(s, i) \]
HW / OS

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

\{ \text{Belief} \}
\[ p(s) \]
Proposed Approach

\[ P(s, i) \]
Source Code

\[ \{ \phi_j(s, i) \} \]
path constraints

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

\[ \times 1000 \]

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

\[ P(s, i) \]
HW / OS

\[ \{ \mathcal{H}, \mathcal{H}, \ldots \} \]
\[ p(o|s_j, i_j) \]
\[ p(s) \]
Belief

\[ \text{MAX } \mathcal{H} \]
\[ i^* \]

\[ \mathcal{H} \]

\[ \mathcal{H} \]

\[ \mathcal{H} \]

\[ \mathcal{H} \]
Proposed Approach

\[ P(s, i) \rightarrow \text{SYMBOLIC EXECUTION} \rightarrow \{\phi_j(s, i)\} \rightarrow \{w_j = (s_j, i_j)\} \]

\( P(s, i) \)  
Source Code  

\( \times 1000 \)

\( \{w_j = (s_i, i_j)\} \)

\text{Network}

\( P(s, i) \)  
HW / OS  

\( \{\text{Waves}, \ldots\} \)

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

\( \text{Belief} \)

\( p(s) \)

\( \text{Weighted Model Counting} \)

\( \text{Max } \mathcal{H} \rightarrow i^* \)
Proposed Approach

\[ P(s, i) \rightarrow \text{Symbolic Execution} \rightarrow \{\phi_j (s, i)\} \rightarrow \{w_j = (s_j, i_j)\} \]

Path constraints

PC models (witnesses)

\[ \times 1000 \]

\[ \{w_j = (s_i, i_j)\} \rightarrow \text{Network} \rightarrow P(s, i) \rightarrow \{\ldots\} \]

\[ P(s, i) \rightarrow \text{HW / OS} \rightarrow \{\ldots\} \]

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

\[ \{\ldots\} \rightarrow \text{Weighted Model Counting} \rightarrow \text{Max } \mathcal{H} \rightarrow i^* \]

Belief

\[ p(s) \]
Proposed Approach

\[ P(s, i) \] Source Code
\[ \xrightarrow{\text{Symbolic Execution}} \] \{ \phi_j(s, i) \} path constraints
\{ w_j = (s_j, i_j) \} PC models (witnesses)

\[ \times 10^3 \]
\{ w_j = (s_i, i_j) \} HW / OS Network

\{ o \mid s_j, i_j \} \text{Belief}
\[ p(s) \]
\[ p(o \mid s_j, i_j) \] Weighted Model Counting
\[ \text{Max } H \]
\[ i^* \]
\[ P(s, \vec{i}) \] HW / OS Network
Proposed Approach

\[ P(s, i) \]

**Source Code**

\[ \{ \phi_j(s, i) \} \]

**path constraints**

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

**PC models (witnesses)**

\[ \times 1000 \]

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

**Network**

\[ P(s, i) \]

\[ \{ \ldots \} \]

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

\[ p(s) \]

**Weighted Model Counting**

\[ \text{Max } \mathcal{H} \]

\[ i^* \]

**Network**

\[ P(s, \tilde{i}) \]

**Observe** \( o \)
Proposed Approach

\[ P(s, i) \]

Source Code

**Symbolic Execution**

\[ \{ \phi_j(s, i) \} \]

path constraints

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

PC models (witnesses)

\[ P(s, i) \times 1000 \]

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

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

\[ p(o \mid s, \vec{i}) \]

**Belief**

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

**Weighted Model Counting**

\[ \text{MAX } \mathcal{H} \]

\[ i^* \]

**Bayesian Update**

\[ \text{observe } o \]

\[ P(s, \vec{i}) \]
Proposed Approach

\[ P(s, i) \]

Source Code

\[ \text{Symbolic Execution} \]

\[ \{ \phi_j(s, i) \} \]

path constraints

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

PC models (witnesses)

\[ P(s, i) \times 1000 \]

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

\[ \{ \lambda, \lambda, \ldots \} \]

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

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

Weighted Model Counting

\[ \text{Max } \mathcal{H} \]

\[ i^* \]

Bayesian Update

observe \( o \)

\[ P(s, \vec{i}) \]

HW / OS

Network

\[ \{ \lambda, \lambda, \ldots \} \]

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

\[ P(s, i) \]  
Source Code  

**Symbolic Execution**

\[ \{ \phi_j(s, i) \} \]  
path constraints

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

\[
\begin{align*}
\times 1000 & \\
\{ w_j = (s_i, i_j) \} & \rightarrow P(s, i) \\
\end{align*}
\]

\[
\begin{align*}
\{ w_j = (s_i, i_j) \} & \rightarrow \text{Network} \\
\end{align*}
\]

\[
\begin{align*}
P(s, i) & \rightarrow \{ p(o|s_j, i_j) \} \\
\end{align*}
\]

\[
\begin{align*}
\{ p(o|s_j, i_j) \} & \rightarrow \text{Belief} \\
\end{align*}
\]

\[
\begin{align*}
\text{Weighted Model Counting} & \rightarrow \text{MAX } \mathcal{H} \\
\end{align*}
\]

\[
\begin{align*}
\text{MAX } \mathcal{H} & \rightarrow i^* \\
\end{align*}
\]

\[
\begin{align*}
\{ p(s|o, i^*) \} & \rightarrow \text{Bayesian Update} \\
\end{align*}
\]

\[
\begin{align*}
P(s, \vec{i}) & \rightarrow \text{Network} \\
\end{align*}
\]

\[
\begin{align*}
\text{HW / OS} & \times 1000 \\
\end{align*}
\]
Prototype 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
Proposed Experiments

DARPA Space-Time Analysis for Cybersecurity (STAC)
Canonical Side-Channel Vulnerability Benchmark
https://github.com/Apogee-Research/STAC/
7 Applications, 1 to 3 variants each
14 total programs

Compare the two approaches.
Proposed Work Summary

1. Offline Static Analysis

2. Offline Dynamic Analysis

3. Online Attack Synthesis
Publications

- Aydin, **Bang**, Bultan: Automata-Based Model Counting for String Constraints. CAV ’15.


- **Bang**, Aydin, Phan, Pasareanu, Bultan: String Analysis for Side Channels with Segmented Oracles. FSE ’16.

- Phan, **Bang**, Pasareanu, Malacaria, Bultan: Synthesis of Adaptive Side-Channel Attacks. CSF ’17.
Timeline

Fall 2017:
Unified theoretical model for side-channel techniques from my work.
Incorporate feedback from committee.
Improve prototype implementation.

Winter 2018:
Finish implementation.
Finish all experiments.

Spring 2018:
Complete dissertation draft by April.
Defend dissertation in May.
Thanks!

Questions?
Questions?