IDS

Every call will have a unique number

Null(2);
Log(...);
Null(-2);
Null(3);
Log(...)
Null(-3);
Null(5);
Log(...);
Null(-5);

Efficient constant sensitive ID
Null calls eliminate all ambiguity about stack and current function

Null call log

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-2</td>
<td>3</td>
</tr>
</tbody>
</table>
Null call trace

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-2</td>
<td>3</td>
<td>-3</td>
<td>5</td>
</tr>
</tbody>
</table>
If we add

![Diagram Showing Sequence of Events]

This will shrink trace log

![Diagram Showing Event Log Reduction]

**Mimicry attack**

Attacker mimics correct behavior
Performance of this system: low overhead

**Account all the outgoing edges**

Node

The best is have one brounch, so the attacker has no other choices.

**Sandboxes**

Goal: give untrusted programs limited rights (least privilege)

**IDS**

```
+-----+      +-----+
| APP |      | Model|
|-----|      +-----+
| Derived |      | Syscalls |
|-----|      | check |
```

**Sandboxing**

```
+-----+      +-----+
| APP |      | Policy |
|-----|      +-----+
| User |      | Syscalls |
|-----|      | Allow/not allowed |
```