Thwarting Cache Side-Channel Attacks Through Dynamic Software Diversity

Stephen Crane, Andrei Homescu, Stefan Brunthaler, Per Larsen, Michael Franz
University of California, Irvine

NDSS 2015
Side-Channel Attacks

The attacker:

- Observes dynamic side-effects of computation — timing, cache footprint, power consumption, ...
- Derives secret information from side-channel observations

Input 1

Side-Channel Observation

Input 2

Side-Channel Observation

Input 3

Side-Channel Observation
• Ideal defense decouples all side-channel observations from input
  – Usually requires manual programmer effort or custom hardware for each possible side channel
Side-Channel Attacks

THE PROBLEM

- Ideal defense decouples all side-channel observations from input
  - Usually requires manual programmer effort or custom hardware for each possible side channel
Manual Side-Channel Mitigation

THE PROBLEM

- Secret input
  - If
    - Then 1s
    - Else 10s
  - Process
Automated Software Diversity Approach

Multiple functionally equivalent copies which vary in implementation details

Techniques:
- NOP insertion
- Function reordering
- Register randomization
- Instruction substitution
Diversity
Diversity
Control-Flow Diversity
Each loop iteration results in different side-channel observations, even with the same input.
Side-Channel Variation

Each loop iteration results in different side-channel observations, even with the same input.
Side-Channel Variation

Each loop iteration results in different side-channel observations, even with the same input.
dispatch to replica $i$

roll random $n$-sided die

$i \in \{1...n\}$

dispatch to replica $i$

$n$ replicas
Optimized Asynchronous Update

Implementation

Table Rewriter Thread

```c
for (;;) {
    for each entry E {
        E = choose_rand_variant();
    }
    sleep();
}
```

Dispatch Table

<table>
<thead>
<tr>
<th>Number</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>0x34523232</td>
</tr>
<tr>
<td>100</td>
<td>0xABABCD</td>
</tr>
<tr>
<td>101</td>
<td>0x32345982</td>
</tr>
<tr>
<td>199</td>
<td>0x12345678</td>
</tr>
<tr>
<td>200</td>
<td>0x23451234</td>
</tr>
<tr>
<td>201</td>
<td>0xABBB1234</td>
</tr>
</tbody>
</table>
AES Cache Side-Channel Attack

- Practical attack on the libgcrypt AES implementation
  - Targets L2 caching of AES S-box table lookups
- Modern hardware
  - Intel Core 2 Quad Q9300, 2.5Ghz
- Two types of cache side channels [1]:
  - EVICT+TIME: Overall timing
  - PRIME+PROBE: Cache usage

EVICT+TIME Attack

EVALUATION

Encrypt

Evict cache set $c$

Encrypt

Measure encryption time

Cache Contents
EVICT+TIME Attack

EVALUATION

Encrypt

Evict cache set $c$

Encrypt

Measure encryption time

Cache Contents
EVICT+TIME Attack

EVALUATION

Encrypt

Evict cache set c

Encrypt

Measure encryption time

Cache Contents
EVICT+TIME Attack

EVALUATION

Encrypt

Evict cache set $c$

Encrypt

Measure encryption time

Cache Contents
PRIME+PROBE Attack

EVALUATION

Load cache

Encrypt

Time cache (re)loads

Cache Contents
PRIME+PROBE Attack Evaluation

- Load cache
- Encrypt
- Time cache (re)loads

Cache Contents
PRIME+PROBE Attack

EVALUATION

Load cache

Encrypt

Time cache (re)loads

Cache Contents
PRIME+PROBE Attack Evaluation

Cache Contents

Load cache
Encrypt
Time cache (re)loads

slow load
fast load
• Attacks observe cache usage

• We alter cache behavior by randomly adding memory loads

• Tested two memory load variants: static & dynamic
  – both overwrite AES S-box cache lines

randomly add memory loads
Dashed red line indicates the expected success of an attacker with no side-channel information.
Dashed red line indicates the expected success of an attacker with no side-channel information.
1.5x – 2.0x for practical configurations
Conclusion

- **Generic technique for dynamic runtime diversity**

- Dynamic control-flow diversity significantly reduces side-channel leakage
  - reasonable overhead
  - no developer effort
Questions?

Stephen Crane
sjcrane@uci.edu