“They’re not that hard to mitigate”: What Cryptographic Library Developers Think About Timing Attacks

Jan Jancar¹, Marcel Fourné², Daniel De Almeida Braga³, Mohamed Sabt³, Peter Schwabe², Gilles Barthe², Pierre-Alain Fouque³ and Yasemin Acar²,⁴
Timing attacks

- **When?** 25+ years old
Timing attacks

- **When?** 25+ years old
- **What?** Duration of operation leaks information on secrets
Timing attacks

- **When?** 25+ years old
- **What?** Duration of operation leaks information on secrets
- **Why?** Branches or memory accesses on secret-derived values
Timing attacks

- **When?** 25+ years old
- **What?** Duration of operation leaks information on secrets
- **Why?** Branches or memory accesses on secret-derived values
- They are still around

“They’re not that hard to mitigate”: What Cryptographic Library Developers Think About Timing Attacks
## Timing attacks

### Resistance and tools for verification

- **Constant-time code practice**
- **Tools to verify constant-timleness**

<table>
<thead>
<tr>
<th>Tool</th>
<th>Target</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABPV13</td>
<td>C</td>
<td>Formal</td>
</tr>
<tr>
<td>Binsec/Rel</td>
<td>Binary</td>
<td>Symbolic</td>
</tr>
<tr>
<td>Blazer</td>
<td>Java</td>
<td>Formal</td>
</tr>
<tr>
<td>BPT17</td>
<td>C</td>
<td>Formal</td>
</tr>
<tr>
<td>CacheAudit</td>
<td>Binary</td>
<td>Symbolic</td>
</tr>
<tr>
<td>CacheD</td>
<td>Trace</td>
<td>Symbolic</td>
</tr>
<tr>
<td>COCO-CHANNEL</td>
<td>Java</td>
<td>Symbolic</td>
</tr>
<tr>
<td>ctgrind</td>
<td>Binary</td>
<td>Dynamic</td>
</tr>
<tr>
<td>ct-fuzz</td>
<td>LLVM</td>
<td>Dynamic</td>
</tr>
<tr>
<td>ct-verif</td>
<td>LLVM</td>
<td>Formal</td>
</tr>
<tr>
<td>CT-WASM</td>
<td>WASM</td>
<td>Formal</td>
</tr>
<tr>
<td>DATA</td>
<td>Binary</td>
<td>Dynamic</td>
</tr>
<tr>
<td>dudect</td>
<td>Binary</td>
<td>Statistics</td>
</tr>
<tr>
<td>FaCT</td>
<td>DSL</td>
<td>Formal</td>
</tr>
<tr>
<td>FlowTracker</td>
<td>LLVM</td>
<td>Formal</td>
</tr>
<tr>
<td>haybale-pitchfork</td>
<td>LLVM</td>
<td>Symbolic</td>
</tr>
<tr>
<td>KMO12</td>
<td>Binary</td>
<td>Formal</td>
</tr>
<tr>
<td>MemSan</td>
<td>LLVM</td>
<td>Dynamic</td>
</tr>
<tr>
<td>MicroWalk</td>
<td>Binary</td>
<td>Dynamic</td>
</tr>
<tr>
<td>SC-Eliminator</td>
<td>LLVM</td>
<td>Formal</td>
</tr>
<tr>
<td>SideTrail</td>
<td>LLVM</td>
<td>Formal</td>
</tr>
<tr>
<td>Themis</td>
<td>Java</td>
<td>Formal</td>
</tr>
<tr>
<td>timecop</td>
<td>Binary</td>
<td>Dynamic</td>
</tr>
<tr>
<td>tis-ct</td>
<td>C</td>
<td>Symbolic</td>
</tr>
<tr>
<td>VirtualCert</td>
<td>x86</td>
<td>Formal</td>
</tr>
</tbody>
</table>
Why are timing attacks still around?

Are timing attacks part of threat models of libraries?

How do libraries protect against timing attacks?

Are developers aware of the tools?

Why are the tools not being used effectively?
Why are timing attacks still around?

- Let’s ask the crypto library developers!
- They are the ones that would fix them

Are timing attacks part of threat models of libraries?

How do libraries protect against timing attacks?

Are developers aware of the tools?

Why are the tools not being used effectively?
Ran a survey

Sample

- Targeted open-source cryptographic libraries
- Most-active contributors
  - number of commits
- Invited 201 developers from 36 libraries
  - 44 valid responses
  - 27 libraries
- Thanks to our participants!

libraries

OpenSSL, LibreSSL, Amazon s2n, libgcrypt, RustCrypto, libsecp256k1

developers

11 core developers, 19 maintainers, 11 committers,
Ran a survey

Content

1. Participant background

Asking about

- Background in cryptography
- Experience developing cryptographic code
- Academic / Industry background
Ran a survey

Content

1. Participant background
2. Library / Primitive properties and decisions
3. Tool awareness
4. Tool use
5. Hypothetical tool use
   5a. Dynamic instrumentation
   5b. Statistical runtime tests
   5c. Formal analysis
6. Miscellaneous

Asked about

- role in library
- Library design decisions
- Library threat model
- Timing attack protections in library
- Testing of timing attack resistance of library
Ran a survey

Content

1. Participant background
2. Library / Primitive properties and decisions
3. Tool awareness
4. Tool use
5. Hypothetical tool use
   5a. Dynamic instrumentation
   5b. Statistical runtime tests
   5c. Formal analysis
6. Miscellaneous

Asked about

- awareness of tools
- How learned about them

“They’re not that hard to mitigate”: What Cryptographic Library Developers Think About Timing Attacks
Ran a survey

Content

- 1. Participant background
- 2. Library / Primitive properties and decisions
- 3. Tool awareness
- 4. Tool use
- 5. Hypothetical tool use
  - 5a. Dynamic instrumentation
  - 5b. Statistical runtime tests
  - 5c. Formal analysis
- 6. Miscellaneous

Asked about

- Experience with using tools
Presented properties of three groups of tools

- Groups of tools
  - Dynamic instrumentation
  - Statistical runtime tests
  - Formal analysis

- Properties
  - Requirements on code
  - Guarantees on the results

- Asked about likeliness of use and reasoning
Ran a survey

Content

1. Participant background
   ▼
2. Library / Primitive properties and decisions
   ▼
3. Tool awareness
   ▼
4. Tool use
   ▼
5. Hypothetical tool use
   ▼
   5a. Dynamic instrumentation
   ▼
   5b. Statistical runtime tests
   ▼
   5c. Formal analysis
   ▼
6. Miscellaneous

“They’re not that hard to mitigate”: What Cryptographic Library Developers Think About Timing Attacks
Developers know about timing attacks...

- 100% knew about timing attacks
- Opinions varied

“They’re not that hard to mitigate, at least with the compilers I’m using right now.”

“It was totally obvious for everybody right from the start that protection against timing attacks is necessary.”

“For many cases there aren’t enough real world attacks to justify spending time on preventing timing leaks.”
…and consider them a threat…

- Threat models of libraries
  - Included timing attacks: 23
  - Did not include: 2
- Libraries differentiate between local and remote attacks
  - Include remote: 20
  - Include local: 16

“We worry mostly about timing now. These can vary, remote observation is obviously a bigger issue, local observation cannot be discounted either.”
...and consider them a threat...

- Threat models of libraries
  - Included timing attacks: 23
  - Did not include: 2

- Libraries differentiate between local and remote attacks
  - Include remote: 20
  - Include local: 16

- Reasoning varied
  - Timing attacks relevance, loss of reputation, target environment of library

“We worry mostly about timing now. These can vary, remote observation is obviously a bigger issue, local observation cannot be discounted either.”

“Yes. They [timing attacks] are a concern for some users. And it is never fun to be the bug that a new research paper is talking about exploiting :)”
...that is worth protecting against.

- Claimed resistance against timing attacks
  - Yes, fully: 13
  - Partially: 10
  - No: 3

“It’s just how you write cryptographic code, every other way is the wrong approach (unless in very specific circumstances or if no constant-time algorithm is known).”
…that is worth protecting against.

- Claimed resistance against timing attacks
  - Yes, fully: 13
  - Partially: 10
  - No: 3

- Various protection techniques
  - Constant-time code practice: 21
  - Constant-time algorithm: 9
  - Blinding, slicing, assembly, hardware features, random delays

“It’s just how you write cryptographic code, every other way is the wrong approach (unless in very specific circumstances or if no constant-time algorithm is known).”

“Conditional branches and lookups are avoided on secrets. Assembly code and common tricks are used to prevent compiler optimizations.”
Most heard about the tools...

- 33% heard of at least one
- Well-known tools:
  - ct-grind: 27%
  - ct-verif: 17%
  - MemSan: 8%
- Most of the tools were unknown

“We independently came up with this approach and were using it [before we] knew ctgrind existed.”
Most heard about the tools...  

- 33% heard of at least one
- Well-known tools:
  - ct-grind: 27%
  - ct-verif: 17%
  - MemSan: 8%
- Most of the tools were unknown

...but haven’t tried using them,

- Only 19% tried to use
- Why not?
  - Lack of time: 26
  - Inability to ignore issues: 8
  - Tool not maintained: 5
  - Tool not available: 4

“We independently came up with this approach and were using it [before we] knew ctgrind existed.”
...and are unlikely to use some of them.

<table>
<thead>
<tr>
<th>Likeliness of use</th>
<th>Formal analysis</th>
<th>Statistical runtime tests</th>
<th>Dynamic instrumentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very unlikely</td>
<td>8</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Somewhat unlikely</td>
<td>11</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Neutral</td>
<td>12</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Somewhat likely</td>
<td>5</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Very likely</td>
<td></td>
<td>5</td>
<td>12</td>
</tr>
</tbody>
</table>

- **Formal analysis**
  - Perceived as too much effort: 22
- **Dynamic & Statistical tools**
  - Acceptable trade-off between effort and guarantees: 10

“I’m very interested in these sorts of tools, but so far it seems formal analysis tools (at least where we’ve tried to apply it to correctness) are **not really usable by mere mortals yet**.”
There is a leaky pipeline of developers using tools.

- 25% Don’t know about tools
- 31.8% Haven’t tried to use tools
- 4.5% Don’t use tools
- 100% Don’t know about tools

44 Developers

17 Developers

“They’re not that hard to mitigate”: What Cryptographic Library Developers Think About Timing Attacks
Recommendations

**Tool developers**
- Make tools usable
  - Available
  - Easy to install
  - Documentation, examples
- Promote tools at appropriate venues
Recommendations

Tool developers
- Make tools usable
  - Available
  - Easy to install
  - Documentation, examples
- Promote tools at appropriate venues

Crypto developers
- Use the tools, automate, include in CI
- Eliminate all timing leaks
- Mark secrets in code
**Recommendations**

**Tool developers**
- Make tools usable
  - Available
  - Easy to install
  - Documentation, examples
- Promote tools at appropriate venues

**Crypto developers**
- Use the tools, automate, include in CI
- Eliminate all timing leaks
- Mark secrets in code

**Compiler writers**
- Support secret types
  - Do not introduce timing leaks
- Give more control to developers
  - To stop introduction of timing leaks

"They're not that hard to mitigate": What Cryptographic Library Developers Think About Timing Attacks
Recommendations

**Tool developers**
- Make tools usable
  - Available
  - Easy to install
  - Documentation, examples
- Promote tools at appropriate venues

**Crypto developers**
- Use the tools, automate, include in CI
- Eliminate all timing leaks
- Mark secrets in code

**Compiler writers**
- Support secret types
  - Do not introduce timing leaks
- Give more control to developers
  - To stop introduction of timing leaks

**Standardization bodies**
- Encourage submitters to use tools
- Require constant-time code

“They’re not that hard to mitigate”: What Cryptographic Library Developers Think About Timing Attacks
“They’re not that hard to mitigate”: What Cryptographic Library Developers Think About Timing Attacks

Jan Jancar, Marcel Fourné, Daniel De Almeida Braga, Mohamed Sabt, Peter Schwabe, Gilles Barthe, Pierre-Alain Fouque and Yasemin Acar

Developers

- Know and care about timing attacks
- Do not know most tools for verifying constant-timeness
- Do not use tools, mostly due to lack of time

Questions?

J08nY
bit.ly/3riKHWB