Proving the Correctness of Amazon’s s2n TLS Library

Aaron Tomb <atomb@galois.com>
Galois, Inc.

May 11, 2018
International Cryptographic Modules Conference
The s2n TLS Library

- Amazon’s open source implementation of TLS
  - "designed to be simple, small, fast, and with security as a priority"
- We want a high degree of assurance in its correctness
  - Widely used libraries have had serious bugs in the past
- And to retain that assurance as it evolves
  - Many implementation choices, so code changes even when the specification doesn’t
- This talk: using automated reasoning check correctness
  - Integrated with Travis CI to re-check every change
Cryptographic Assurance via Testing

- Typical approach: run on **test vectors**
  - From NIST or other authority
  - One part of FIPS certification
- But bugs occur anyway!
- Some **undefined behavior**: Heartbleed
  - Buffer over-read in OpenSSL
  - A **generic** misuse of language
- Some **incorrect behavior**: OpenSSL
  - ECC modular reduction
    - Incorrect modular reduction
    - Requires knowing **correct** results
Generic vs. Application-Specific Bugs

- Heartbleed is a generic bug:
  - Undefined behavior incorrect regardless of the specification
  - Most code doesn’t have an unambiguous specification
  - Most static analysis tools focus here, for generality

- ECC bug is application-specific:
  - Well-defined, but doesn’t match specification
  - Crypto code often does have a specification!
  - This talk focuses here

- If we have an executable specification, we can test against it!
  - With many more tests that a static set of test vectors

- But for realistic programs, infinite inputs (or effectively so)
Exhaustive Testing via Automated Reasoning

- But exhaustive testing is possible (given a full specification)
  - Enabled by automated reasoning tools: SAT and SMT
  - Specification and production implementation can be translated to logic
  - Automated provers can show that they produce the same results for all possible inputs
- Specification must be machine-readable
Cryptol and SAW

- Open source tools for software analysis
  - Especially cryptographic software verification
- Cryptol allows us to write **concise, unambiguous** specifications
  - Specifications exist for many common algorithms
  - They closely resemble specification documents
- The Software Analysis Workbench (SAW) can compare Cryptol specifications and code
  - Extracts **models** from programs
  - Transforms and **proves** properties of models
  - Supports languages compiled to JVM and LLVM (with more in development)
  - Uses SAT & SMT in the background
HMAC

\[ \text{HMAC}(K, \text{text}) = H((K_0 \oplus \text{opad}) \| H((K_0 \oplus \text{ipad}) \| \text{text})) \]

where

\[ K_0 = \text{kinit } K \]
\[ \text{ipad} = \text{repeat } 0x36 \]
\[ \text{opad} = \text{repeat } 0x5C \]

- Keyed-Hash Message Authentication Code
  - Specified in FIPS 198-1
- NIST document and Cryptol match closely
  - But Cryptol is less ambiguous
- Specification is much simpler!
  - 5 LOC in Cryptol vs. ~200 LOC in C
HMAC: Cryptographic Strength

- HMAC proved secure (by hand) (Bellare, et al., 1996, 2006)
- Later, machine-checked using the Foundational Cryptography Framework (FCF) (Beringer et al., 2015)
- We connected FCF to Cryptol
- Result: the s2n HMAC implementation:
  - is equivalent to the NIST specification, and
  - produces output indistinguishable from random.
DRBG

- Deterministic Random Bit Generator
  - Specified in NIST SP 800-90A
  - Instantiated in CTR mode with AES-128
- Specification is similar in structure to implementation
  - 81 lines of Cryptol vs. 187 lines of C
  - Closer than HMAC because DRBG specified imperatively
  - Specification is shorter…
  - …and could be used to verify multiple implementations!
- Security of the specification is an open question
  - Though machine-checked proof exists for HMAC_DRBG
TLS Handshake

- Coordinates cipher choice, key exchange
  - Specified in RFC 5246
  - Famous bugs here, e.g., SKIP-TLS
- Specification is very different from code
  - s2n implements a **subset** of the RFC
  - RFC spec: 200 lines
  - Subset spec: 242 lines
  - C code: 463 lines
  - Tables are most of spec and C
- Discovered a bug through verification
  - Not security-critical, but incorrect
Continuous Integration

- All of this re-runs on **every commit**, using Travis CI
- Runs **faster** than the concrete tests: 3-15 minutes per proof

```
<table>
<thead>
<tr>
<th>Build Jobs</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ # 1730.16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Takeaway

- Given a specification, **exhaustive** testing is often possible
  - Especially for **cryptographic** code
  - Enabled by **SAT** and **SMT**
- We have done this for parts of the open source s2n TLS library
  - HMAC, DRBG, TLS Handshake Protocol
  - More likely in the future
- Re-checked **on every commit**, using Travis CI
  - Implementations change frequently, even when specifications are stable
Acknowledgements

Contributors include Aaron Tomb, Adam Foltzer, Adam Wick, Andrey Chudnov, Andy Gill, Benjamin Barenblat, Ben Jones, Brian Huffman, Brian Ledger, David Lazar, Dylan McNamee, Eddy Westbrook, Edward Yang, Eric Mertens, Eric Mullen, Fergus Henderson, Iavor Diatchki, Jeff Lewis, Jim Teisher, Joe Hendrix, Joe Hurd, Joe Kiniry, Joel Stanley, Joey Dodds, John Launchbury, John Matthews, Jonathan Daugherty, Kenneth Foner, Kyle Carter, Ledah Casburn, Lee Pike, Levent Erkök, Magnus Carlsson, Mark Shields, Mark Tullsen, Matt Sottile, Nathan Collins, Philip Weaver, Robert Dockins, Sally Browning, Sam Anklesaria, Sigbjørn Finne, Stephen Magill, Thomas Nordin, Trevor Elliott, and Tristan Ravitch.
Resources

- Contact me:
  - Aaron Tomb <atomb@galois.com>
- SAW is open source, BSD3 licensed, on GitHub:
  - http://saw.galois.com
  - https://github.com/GaloisInc/saw-script
- Cryptol open source, BSD3 licensed, on GitHub:
  - http://cryptol.net
  - https://github.com/GaloisInc/cryptol
- HMAC verification blog post: