More Secure Software Systems
by Formal Verification, Property-Based Testing, Secure Compilation, and Dynamic Monitoring

Cătălin Hrițcu
Inria Paris, Prosecco team
Software [in]security is a big problem

e.g. vulnerabilities in TLS (Prosecco)

Tracking the FREAK Attack

On Tuesday, March 3, 2015, researchers announced a new SSL/TLS vulnerability called the FREAK attack. It allows an attacker to intercept HTTPS traffic. The attack, which affects a wide range of devices, uses a weak encryption method to trick the client into using a weaker encryption method. The FREAK attack is discovered by researchers at the University of Michigan, including Zakir Durumeric. For more details, see this Washington Post article.

The Logjam Attack

The Logjam Attack is a new type of SSL/TLS vulnerability that allows attackers to decrypt HTTPS traffic. The attack is discovered by researchers at the University of Michigan. For more details, see this Washington Post article.

The BEAST Wins Again: Why TLS Keeps Failing to Protect HTTP

FREAK Attack Threatens SSL Clients

Documents
- PDF of slides
- summary of briefing
- Paper: Virtually Host
- Paper: Triple Handshakes

Exploit videos
- Chrome
- Firefox
- Internet Explorer

msm1267 writes:
For the nth time in the last couple of years, security experts are warning about a new Internet-scale vulnerability, this time in some popular SSL clients. The flaw allows attackers to force clients to downscale to weakened ciphers and break their supposedly encrypted communications through a man-in-the-middle attack.
Formal verification can help

• ... find bugs & prove security

• ProVerif & CryptoVerif
  – Prosecco tools for automatically analyzing the security of crypto protocol models
  – successful for finding logical flaws early in protocol design phase
Formal verification can help

- find bugs & prove security
- ProVerif & CryptoVerif

- Prosecco tools for automatically analyzing the security of crypto protocol models
- successful for finding logical flaws early in protocol design phase
Formal verification can help

• ... find bugs & prove security

• **ProVerif** & **CryptoVerif**
  – Prosecco tools for *automatically* analyzing the security of crypto protocol *models*
  – successful for finding logical flaws early in protocol design phase

• Just that **models are very abstract**
  – previous proofs of TLS models *missed* implementation attacks

• Verified models are cool
  – **but verified implementations are much coolear**
Verifying implementations with

- **F** is a new programming language
- ... putting together:
  - **impure functional programming** in ML
    - extracts to OCaml and F#, interoperates
  - the **automation** of SMT-based verification systems
    - like in Why3, Frama-C, Boogie, VCC, Dafny
  - the **expressive power** of interactive proof assistants based on dependent types
    - like in Coq, Agda, or Lean
miTLS*

- Formally verified reference implementation of TLS 1.2 in F* (working towards TLS 1.3)
- Written from scratch focusing on verification
The limits of formal verification

• **scalability**
  – state of the art for verifying correctness and security of systems is 10.000-20.000 LOC (and 500.000 LOP)

• **legacy code** (e.g. OpenSSL)
  – vs nice fresh reference implementations (e.g. miTLS*)

• **effort of failed proofs** (automatic or interactive)
  – finding bugs by failed proof attempts very costly
  – can find very interesting bugs by testing
SMACKTest: testing TLS state machine

Live state machine attack testing.

Run tests against your browser

SmackTest can connect your browser to a FlexTLS instance and model various SMACKTLS traces that will try to trick your TLS instance into adopting an insecure state. Start

Run tests against your server

SmackTest can create a FlexTLS instance that can evaluate SMACKTLS tests against a server and return detailed trace results. Start

Downloads

- USENIX paper (WOOT 2015): PDF
- USENIX slides (WOOT 2015): PDF
- FlexTLS source code: TAR
SMACKTest: testing TLS state machine

Live state machine attack testing.

If the test does not begin, click here to launch it manually, then return to this tab to inspect results.

298: Test incomplete. Click for detailed log.

297: Test incomplete. Click for detailed log.

296: Test failed. Click for detailed log.

295: Test succeeded. Click for detailed log.

294: Test incomplete. Click for detailed log.

293: Test incomplete. Click for detailed log.

292: Test incomplete. Click for detailed log.

291: Test incomplete. Click for detailed log.

290: Test incomplete. Click for detailed log.

289: Test succeeded. Click for detailed log.
Dependable property-based testing

• Beyond just finding bugs, confidence by testing
• Integrating testing and formal verification
  – QuickChick: property-based testing for Coq (soon F* too)
    • i.e. putting the “property” back in property-based testing
• Systematically measuring testing quality
  – Polarized mutation testing
    • i.e. property-based mutation
• Making testing more thorough and cost-effective
  – Luck: a domain-specific language for data generators
    • i.e. property-based generation
Back to miTLS*

Problem 1: insecure languages

15,000 LOC
F*

50,000 LOC
OCaml

400,000 LOC
C
ASM

Problem 2: insecure interaction

compiled F* ↔ compiled OCaml ↔ compiled C ↔ compiled ASM

OK we can thoroughly test this
OK we can verify this

Ooops
Secure compilation

1. Secure language semantics (e.g. memory safe C)
2. Secure language interaction (dynamic isolation, call discipline, type checking, immutability, uniqueness, ...)

But, at what cost? In software, 10x? 100x? 1000x?

Micro-policies
- new tagged hardware architecture
- associates large metadata tag to each word
- efficiently propagates and checks tags; hw caching
- dynamic monitoring: software defined, very flexible, fine-grained (words, instructions), fast ...
- ... average 10% runtime overhead for complex policies!
More Secure Software Systems

• Formal Verification
• Property-Based Testing
• Secure Compilation
• Dynamic Monitoring
• ... they can all play a role!

Thank you!