



More Secure Software Systems

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

Cătălin Hrițcu

Inria Paris, Prosecco team

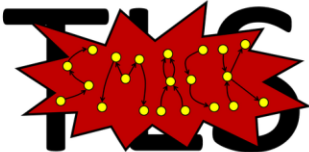
Software [in]security is a big problem

e.g. vulnerabilities in TLS (Prosecco)

MI TLS 3SHAKE SMACK VHC

SMACK Introduction Threat Model SKIP-TLS Attack FREAK Attack

SMACK: State Machine Attacks



Implementations of the Transport Layer Security (TLS) protocol handle a variety of protocol versions and modes and key exchange methods, which we describe in a different message sequence number (MSN) server. We address the problem of a state machine that can correctly handle all modes.

Triple Handshakes Compromise TLS Security: Breaking and Fixing Authentication

March 4, 2014

Introduction	TLS Weaknesses	Triple Handshake
Countermeasures	Disclosure	Other

Slides from the TLS WG session at IETF89 and our proposed fixes.

Our research paper with more details on the attacks (see Sections 2 and 3) is available here.

THE WALL STREET JOURNAL

Home World U.S. Politics Economy Business Tech Markets Opinion Arts Life Real Estate

TECH

New Computer Bug Exposes Broad Security Flaws

Fix for Logjam bug could make more than 20,000 websites unreachable



The Logjam Attack

Warning! Your web browser is vulnerable to Logjam and can be tricked into using weak encryption. Update your browser.

Diffie-Hellman key exchange is a popular cryptographic algorithm that allows Internet protocols to establish a shared key and negotiate a secure connection. It is fundamental to many protocols including HTTP, SMTPS, and protocols that rely on TLS.

We have uncovered several weaknesses in how Diffie-Hellman key exchange has been deployed:

Tracking the FREAK Attack

Good News! Your browser appears to be safe from 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 connections between web browsers and servers and downgrade them to weak encryption. We are tracking the impact of the attack.

The FREAK attack was discovered by researchers at the University of Michigan, including Zakir Durumeric. The disclosure was coordinated by the researchers. The team can be contacted at freak@umich.edu.

For additional details about the attack, see this Washington Post article, and this Ars Technica article.

ars technica

MAIN MENU MY STORIES: 24 FORUMS SUBSCRIBE JOBS ARS CONSORTIUM

RISK ASSESSMENT / SECURITY & HACKTIVISM

"FREAK" flaw in Android and Apple devices cripples HTTPS crypto protection

Bug forces millions of sites to use easily breakable key once thought to be dead.

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

Documents

- [PDF of slides](#)
- [summary of briefing](#)
- [Paper: Virtual Host](#)
- [Paper: Triple Handshake](#)

Exploit videos

Disclaimer: The goal of this video is to demonstrate a vulnerability in TLS. We are happy to acknowledge the work of the researchers who discovered this vulnerability and Mozilla. We are also

FREAK Attack Threatens SSL Clients

Posted by [Soulskill](#) on Tuesday March 03, 2015 @04:29PM from the another-day-another-vuln dept.

[msm1267](#) writes:

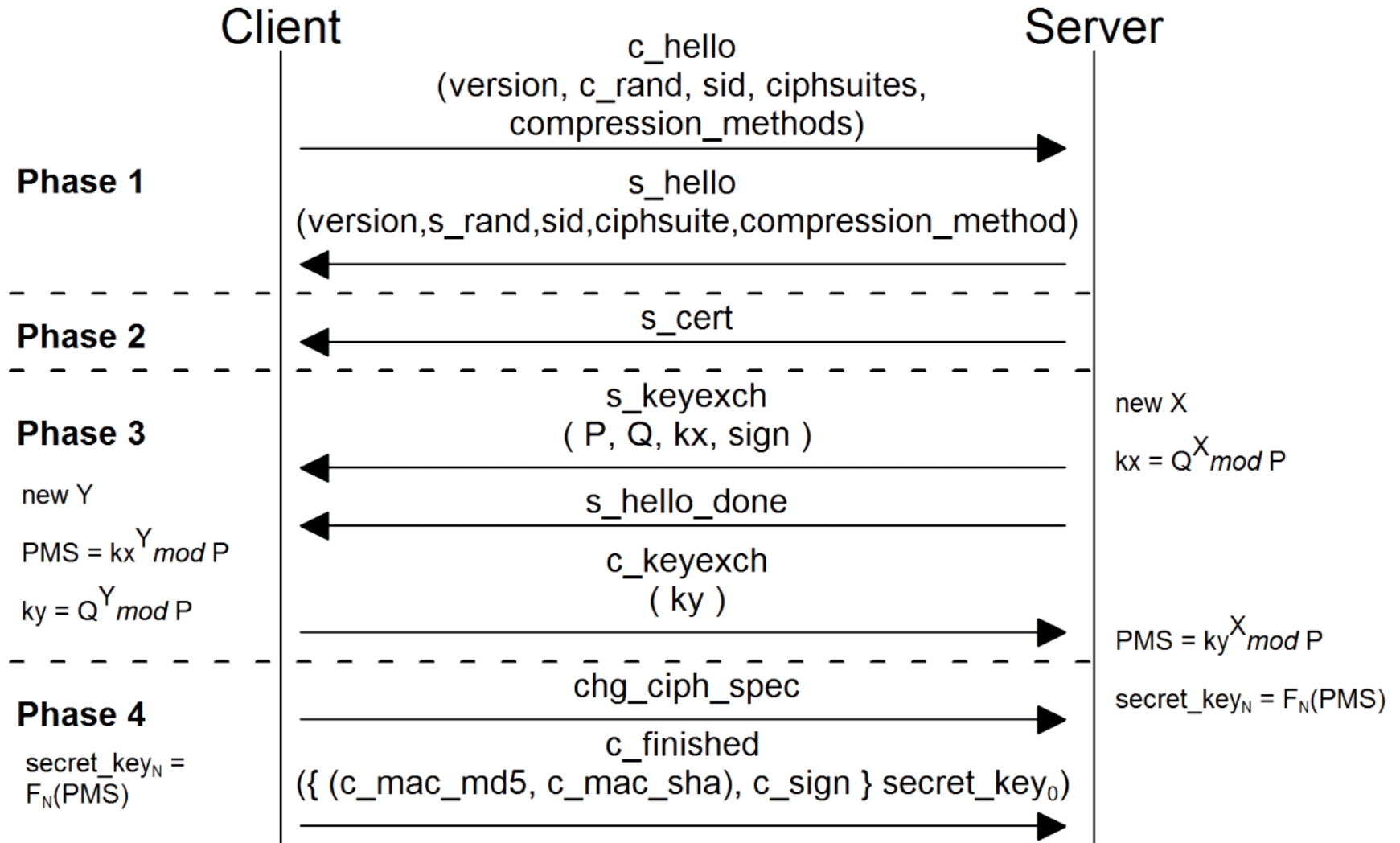
For the nth time in the last couple of years, security experts are warning about a [Internet-scale vulnerability, this time in some popular SSL clients](#). The flaw allows an [attacker to force clients to downgrade to weakened ciphers](#) and break their supposedly encrypted communications through a 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



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 cooler**

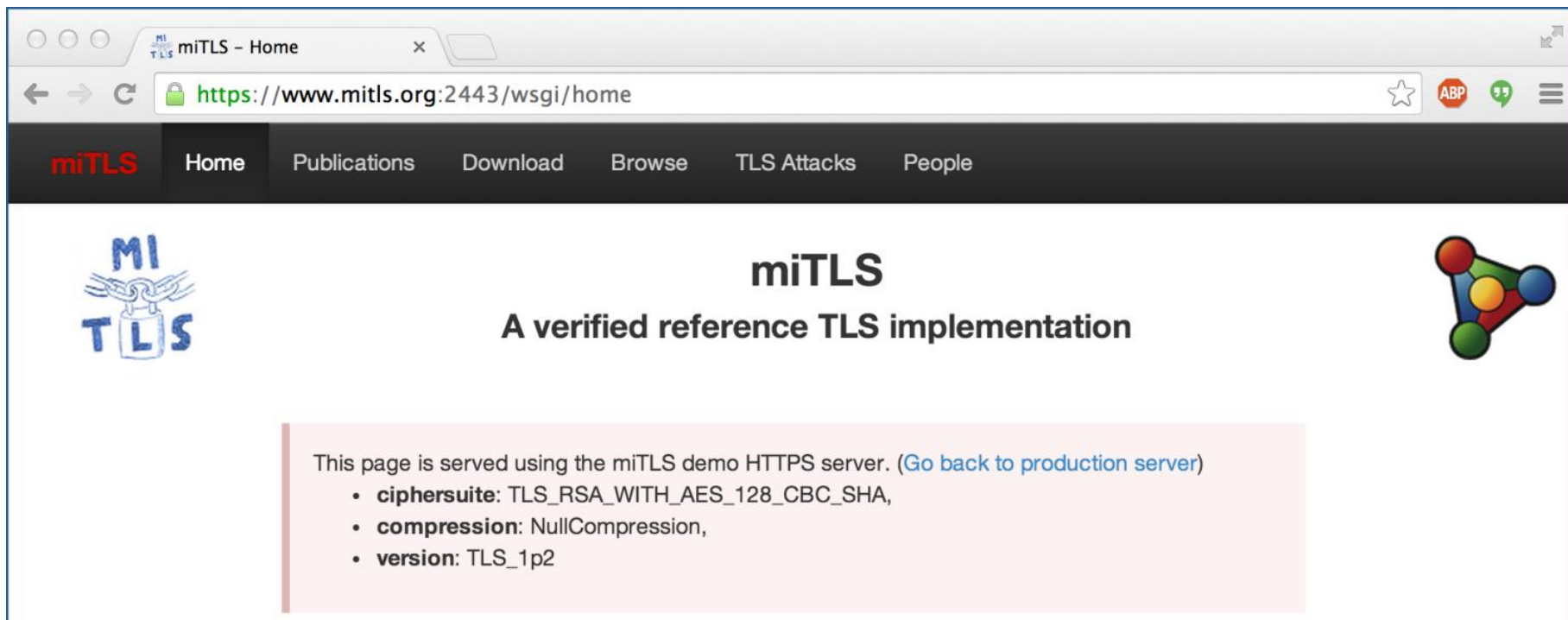
Verifying implementations with



- **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 screenshot shows a web browser window with the address bar displaying `https://www.mitls.org:2443/wsgi/home`. The page features a navigation menu with links for Home, Publications, Download, Browse, TLS Attacks, and People. The main content area includes the miTLS logo on the left, the title "miTLS" in the center, and the subtitle "A verified reference TLS implementation" below it. On the right side, there is a colorful geometric logo. A light pink box at the bottom of the page contains the following text:

This page is served using the miTLS demo HTTPS server. ([Go back to production server](#))

- **ciphersuite:** TLS_RSA_WITH_AES_128_CBC_SHA,
- **compression:** NullCompression,
- **version:** TLS_1p2

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.



Run tests against your server

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

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.

ClientHello

ServerHello

ServerCertificate

ServerKeyExchange

Authenticate Client

ServerCertificateRequest

ServerHelloDone

ClientCertificate

ClientKeyExchange

ClientCertificateVerify

ClientCCS

ClientFinished

ServerNewSessionTicket

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.

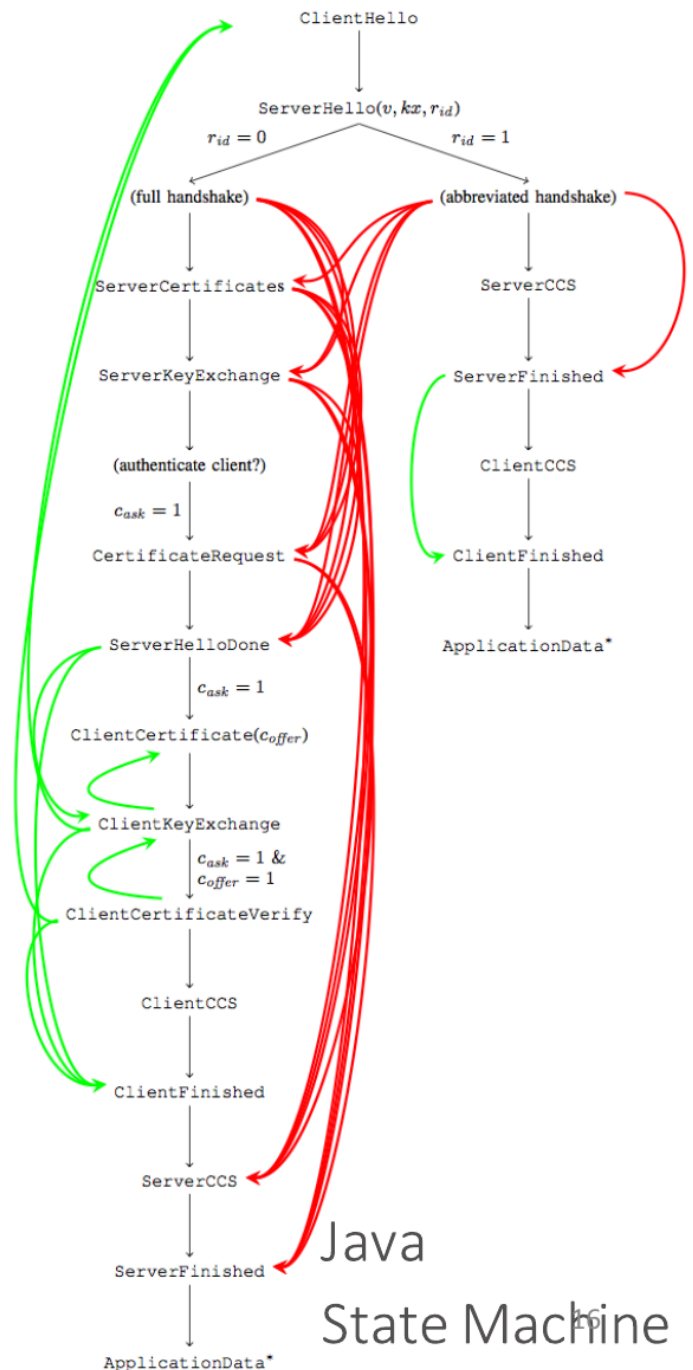
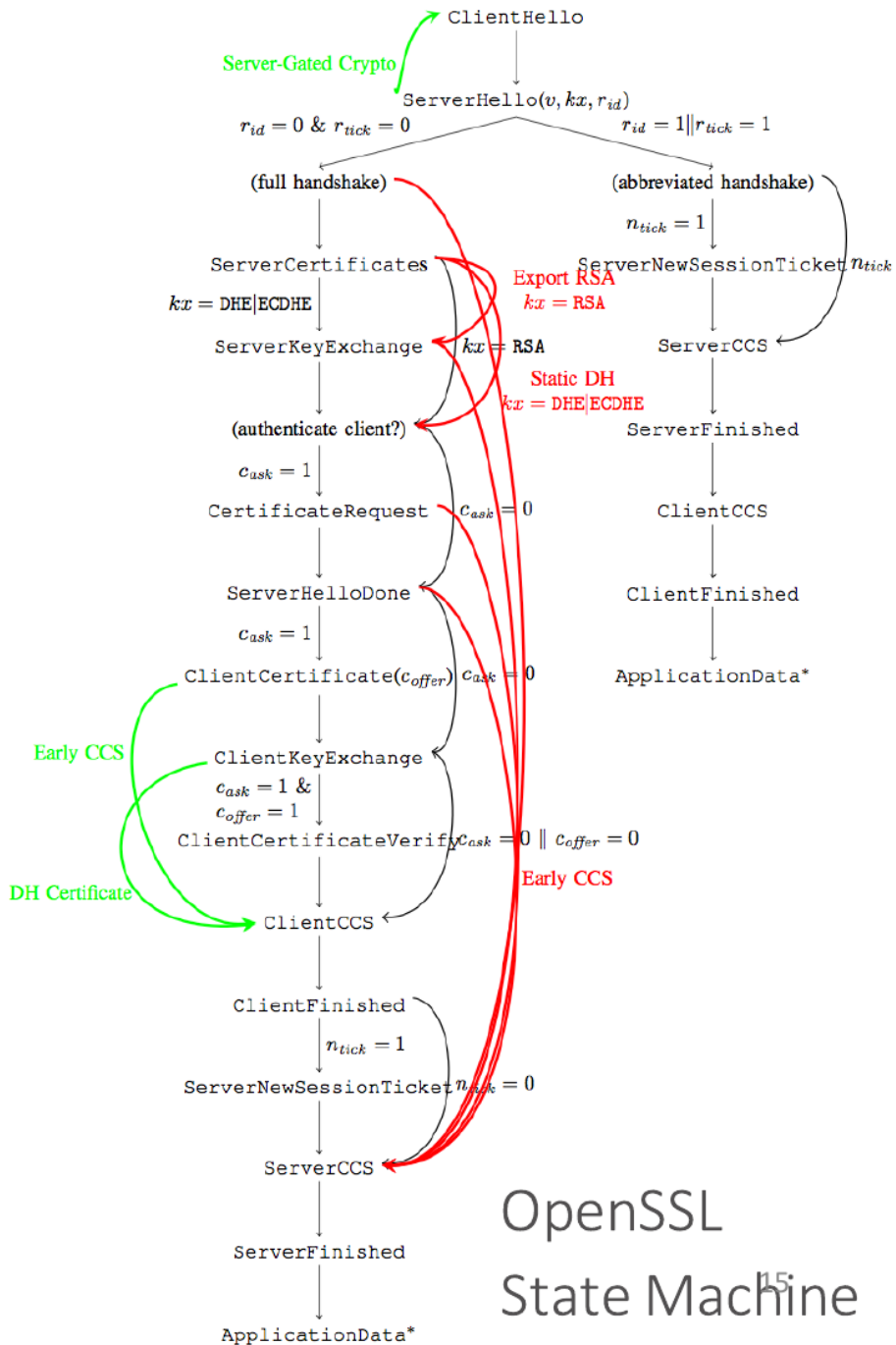
293: Test incomplete. Click for detailed log.

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.



OpenSSL
State Machine

Java
State Machine

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*

Ooops



OK we can thoroughly test this

OK we can verify this

15.000 LOC

50.000 LOC

F*

OCaml

Problem 1: insecure languages

400.000 LOC

C

ASM

compiled F*

compiled OCaml

compiled C

compiled ASM

Problem 2: insecure interaction

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!