# TLS-Attacker

## Systematic Fuzzing and Testing of TLS Libraries

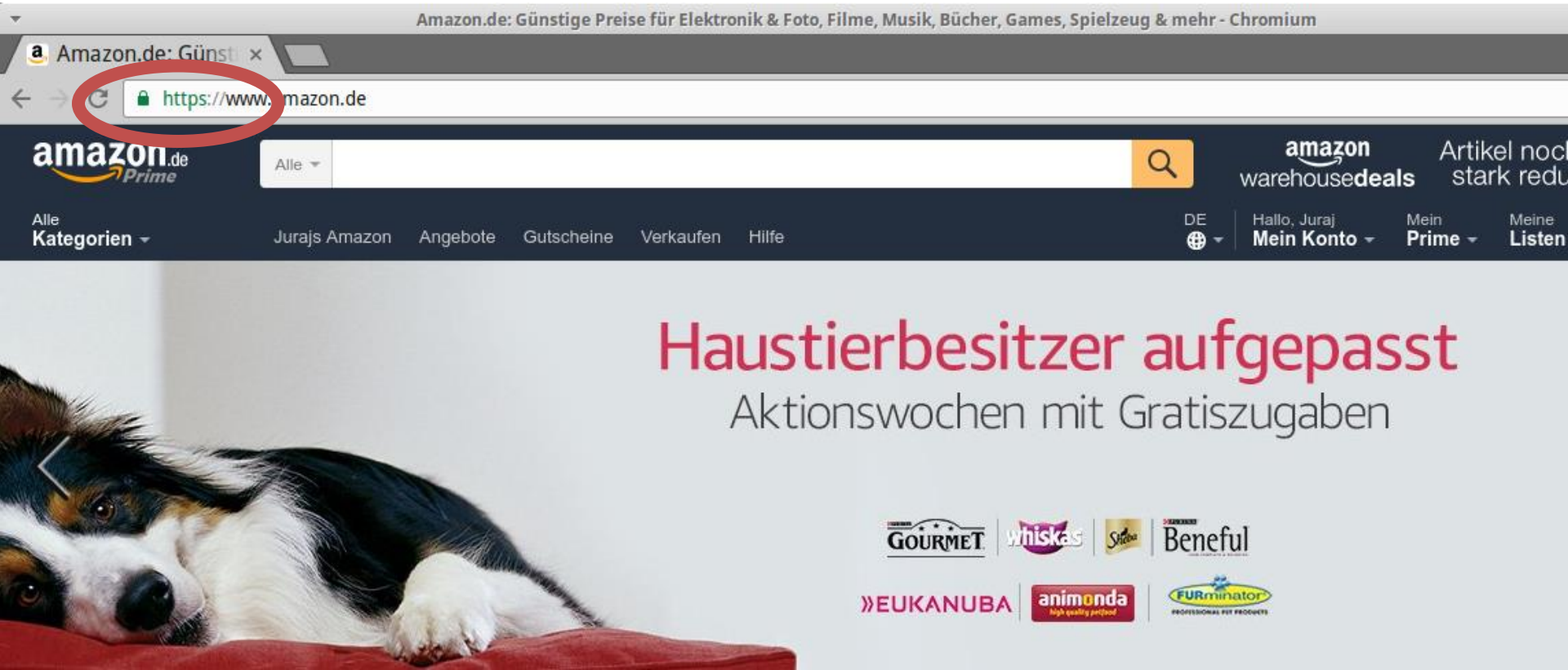
**Juraj Somorovsky**

RUHR UNIVERSITÄT BOCHUM RUB

HACKMANIT

# Transport Layer Security

- The most important crypto protocol
- HTTP, SMTP, IMAP …



# TLS History

Secure Sockets Layer (SSL), SSLv2

SSLv3

1995

Wagner, Schneier: Analysis of SSLv3

Bleichenbacher's attack

Transport Layer Security

2000

Padding oracle attack

2005

TLS 1.1

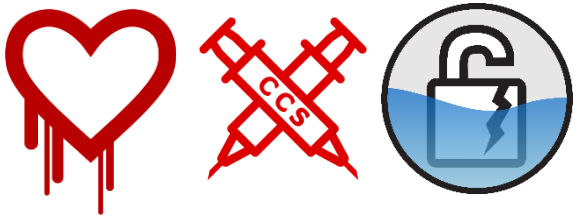
TLS 1.2

2010

BEAST, CRIME, BREACH, Lucky 13

TLS 1.3

2015



# Questions

- How can we test these attacks?
- Can we find such attacks automatically?

# Approach [SP2-17]

1. Collect TLS libraries
2.
3. Profit

# Approach [SP2-17]

1. Collect TLS libraries
2.
3. Profit

# Approach [SP2-17]

1. Collect TLS libraries
2. 
3. Profit

hackerone About Product

who succeed in this challenge deserve

## Bounty Qualification

The project maintainers have final decis
respect their decision, and we ask that y
determined by the project.

- **Critical**: $5,000+
- **High**: $2,500
- **Moderate**: $1,000
- **Low**: $500

# Approach [SP2-17]

1. Collect TLS libraries
2. 
3. Profit

# Contributions

- Flexible TLS framework
- Fuzzing, testing, writing attacks …
- High impact vulnerability in OpenSSL
- Additional vulnerabilities in Botan, MatrixSSL…
- https://github.com/RUB-NDS/TLS-Attacker

RUB-NDS / TLS-Attacker

Watch 37 | Star 277 | Fork 50

Code | Issues 2 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs

TLS-Attacker is a Java-based framework for analyzing TLS libraries. It is developed by the Ruhr University Bochum (http://nds.rub.de/) and the Hackmanit GmbH (http://hackmanit.de/).

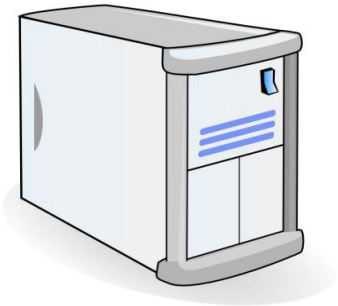
547 commits | 1 branch | 3 releases | 8 contributors | Apache-2.0

Branch: master | New pull request | Find file | Clone or download

# Overview

1. TLS Protocol
2. Framework Prerequisites
3. TLS-Attacker Design
4. Results
5. Conclusions

# TLS RSA Handshake



ClientHello →

← ServerHello
Certificate
ServerHelloDone

ClientKeyExchange
ChangeCipherSpec
(Client-) Finished →

← ChangeCipherSpec
(Server-) Finished

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

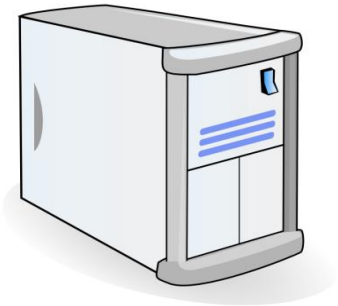
Application →

← Application

# TLS is complex …

- Different versions
- Crypto primitives: RSA, EC, AES, 3DES, RC4, Chacha, Poly1305, New Hope
- Extensions
- Protocol flows

# TLS is complex …



ClientHello →

← ServerHello
Certificate
ServerKeyExchange
ServerHelloDone

Certificate
ClientKeyExchange
CertificateVerify
ChangeCipherSpec
(Client-) Finished →

← ChangeCipherSpec
(Server-) Finished

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Heartbeat →

← Heartbeat

# Recent Attacks on TLS

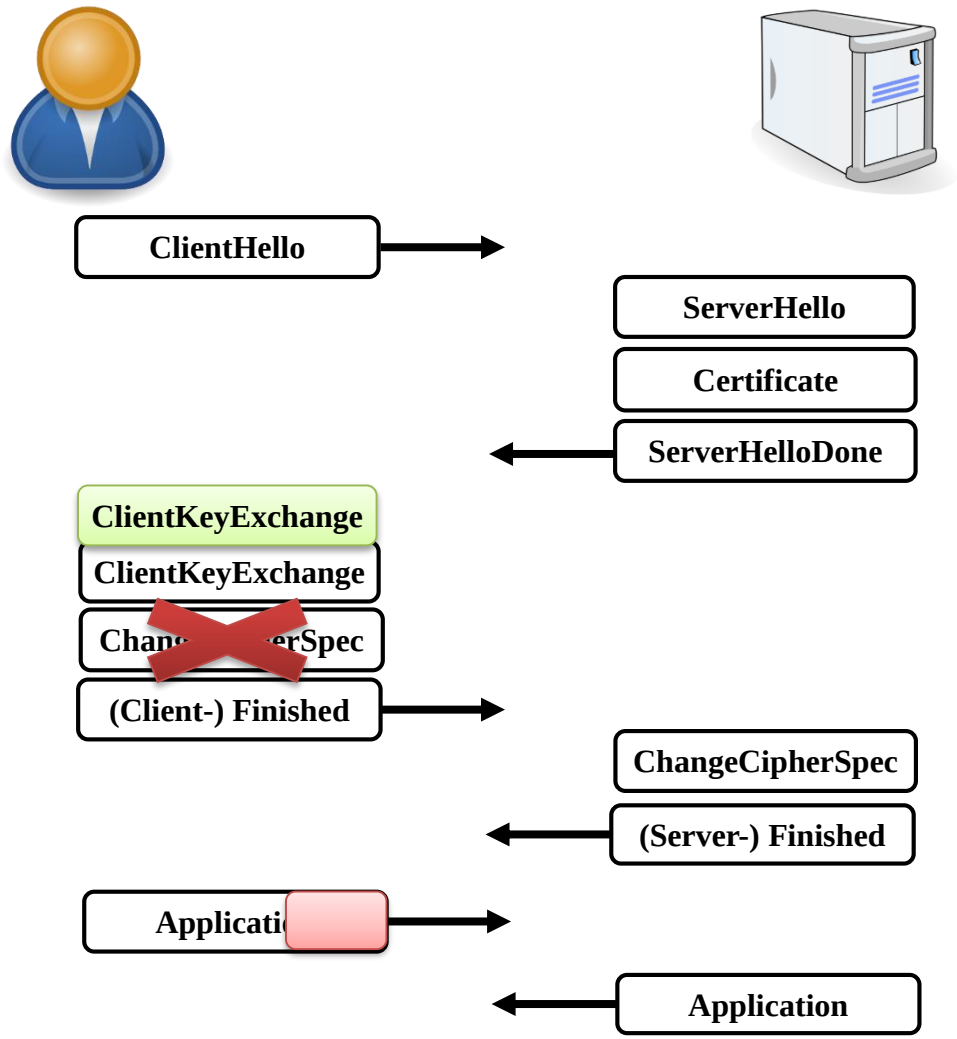
- Not only crypto attacks …
- Attacks on TLS state machines
  - FREAK
  - Early CCS
- Buffer overflows / overreads
  - Heartbleed
  - CVE-2016-6307 (High) -> CVE-2016-6309 (Critical)
- Tool for flexible protocol executions needed

# Overview

1. TLS Protocol
2. Framework Prerequisites
3. TLS-Attacker Design
4. Results
5. Conclusions

# Framework Prerequisites

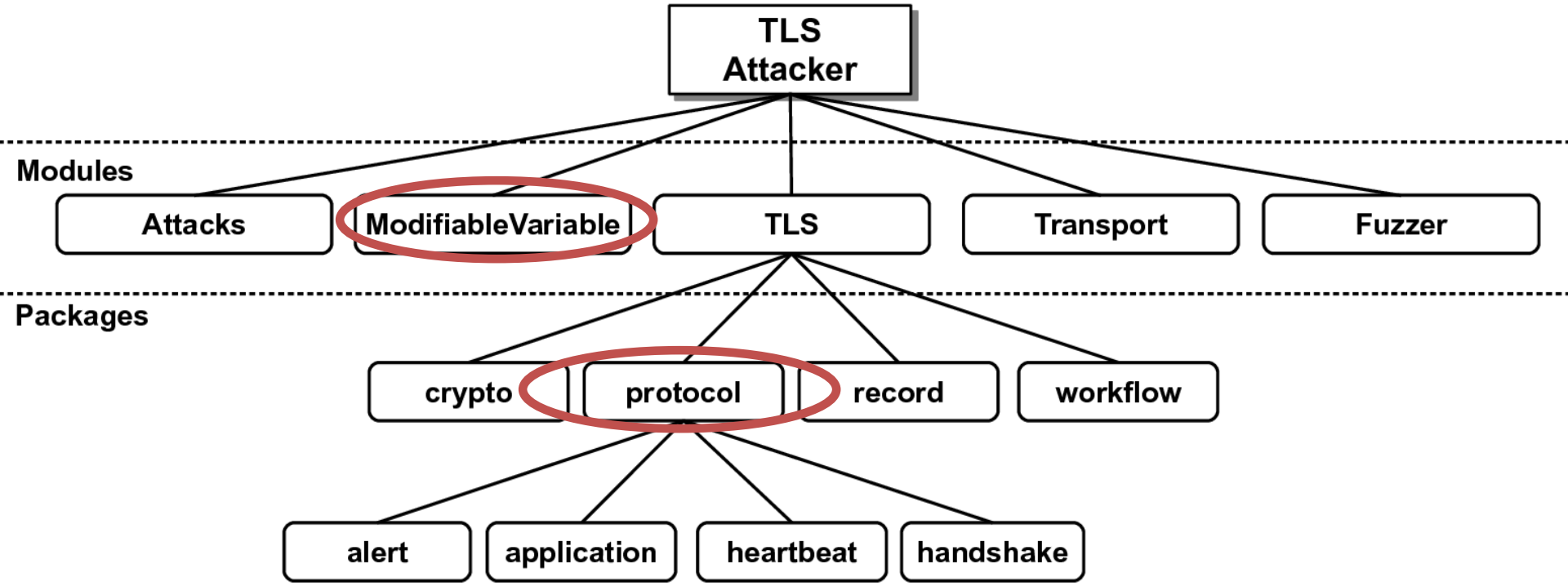
- Flexible protocol flow definition
- Message modifications
- Invalid behavior detection
- Protocol flow reproduction



# Overview

1. TLS Protocol
2. Framework Prerequisites
3. TLS-Attacker Design
4. Results
5. Conclusions

# High-Level Overview



# Modifiable variables

- Define basic data types (integer, byte, arrays) with modifications

- Example:

```
ModifiableInteger i = new ModifiableInteger();
i.setValue( 30 );
i.setModification(new AddModification( 20 ));
System.out.println(i.getValue());  // 50
```

- Further modifications: xor, shuffle, delete, …

# Protocol messages

- ClientHello

| ClientHelloMessage |
| --- |
| **cipherSuites:** ModifiableByteArray<br>**cipherSuiteLength:** ModifiableInteger<br>**…** |
| **getCipherSuites()**<br>**getCipherSuiteLength()** |

- Stored in a message list
- Serializable in XML

# Defining a protocol flow

```
<protocolMessages>
   <ClientHello>
      <supportedCipherSuites>
        <CipherSuite>TLS_RSA_WITH_AES_128_CBC_SHA</CipherSuite>
      </supportedCipherSuites>
   </ClientHello>
   <ServerHello/>
   <Certificate/>
   <ServerHelloDone/>
   <RSAClientKeyExchange/>
   <RSAClientKeyExchange/>
   <ChangeCipherSpec/>
   <Finished/>
   <ChangeCipherSpec/>
   <Finished/>
   <Application/>
</protocolMessages>
```

# Defining a protocol flow

```
<protocolMessages>
   <ClientHello>
      <supportedCipherSuites>
        <CipherSuite>TLS_RSA_WITH_AES_128_CBC_SHA</CipherSuite>
      </supportedCipherSuites>
   </ClientHello>
   <ServerHello/>
   <Certificate/>
   <ServerHelloDone/>
   <RSAClientKeyExchange/>

   <ChangeCipherSpec/>
   <Finished/>
   <ChangeCipherSpec/>
   <Finished/>
   <Heartbeat/>
</protocolMessages>
```

```
<Heartbeat>
  <payloadLength>
    <integerAddModification>
                20000
    </integerAddModification>
  </payloadLength>
</Heartbeat>
```

# TLS-Attacker used for…

- Attacks
- Fuzzing (only server, sorry)
- Test suite

**Demo**

# Overview

1. TLS Protocol
2. Framework Prerequisites
3. TLS-Attacker Design
4. Results
5. Conclusions

# Results

- Padding oracle attack
  - OpenSSL (CVE-2016-2107)
  - Botan 1.11.21 (CVE-2015-7824)
  - MatrixSSL 3.8.2
- Bleichenbacher attack
  - MatrixSSL 3.8.2
- Missing length checks
  - GnuTLS 3.4.9
  - OpenSSL 1.0.1
- Out-of-bound reads / writes
  - OpenSSL-1.1.0-pre1 (stack overflow)
  - Botan 1.11.28 (Out-of-bound read)

# Overview

1. TLS Protocol
2. Framework Prerequisites
3. TLS-Attacker Design
4. Results
5. Conclusions

# Conclusions and future work

- Maintaining a crypto library is hard
- Systematic fuzzing and evaluation needed
- TLS-Attacker
  - For researchers, pentesters
  - For developers
  - Integrated in Botan and MatrixSSL
- Development / fuzzing improvements needed
  - TLS client-side tests
  - Better fuzzing strategies

# Questions

More security research talks?

4.5. – 5.5. 2017

Non-profit security conference