Guided Differential Testing of Certificate Validation in SSL/TLS Implementations

Yuting Chen
Shanghai Jiao Tong University
chenyt@cs.sjtu.edu.cn

This work has been presented at ESEC/FSE-2015.
Background

• SSL (Secure Sockets Layer)/TLS (Transport Layer Security)
How does SSL/TLS work?

1. Customer
2. Server
3. Hello, let's set up a secure SSL session
   - Hello, here is my certificate
   - Also checks that:
     - Certificate is valid
     - Signed by someone user trusts
4. Here is a one time, encryption key for our session
   (encrypted using Server's public key)
5. Server decrypts session key using its private key and establishes a secure session

Does a client validate a certificate correctly?
Defects in SSL/TLS Implementations

• SSL/TLS implementations do contain defects
  o They follow a complicated, ad-hoc process described in several RFCs
    • RFC 2246, 2527, 2818, 4346, 5246, 5280, 6101, 6125...
  o Developers must define their respective validation policies for handling ambiguous descriptions
  o Developers can also make minor mistakes
Implementation/Application

- OpenSSL
- PolarSSL
- Gnutls
- NSS
- CyaSSL
- MatrixSSL
- Chrome
- Internet Explorer
- Firefox
- Opera
- ...

Any behavior differences among SSL/TLS implementations can be taken as potential defects
Outline

- Motivation & problem description
- Approach
  - MCMC sampling
  - Differential testing
- Evaluation
- Conclusion
Previous Work

• Frankencert [Brubaker S&P2014]

2014 IEEE Symposium on Security and Privacy

Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations

Chad Brubaker * †  Suman Jana †  Baishakhi Ray †  Sarfraz Khurshid †  Vitaly Shmatikov †
*Google
†The University of Texas at Austin
‡University of California, Davis
Previous Work (cont.)

- 14 different SSL/TLS implementations
- 208 discrepancies due to 15 root causes
- Multiple bugs
  - Accepting fake and unauthorized intermediate Certificate Authorities (CAs) attacker can impersonate any website!
  - Accepting certificates not authorized for use in SSL or not valid for server authentication
  - …
Motivation

8 Million Test Certificates!
8000000

208 discrepancies/15 root causes
Mucert

• Idea: Certificate fuzzing + Differential testing

• OpenSSL
• PolarSSL
• Gnutls
• NSS
• CyaSSL
• MatrixSSL
• Chrome
• Internet Explorer
• Firefox
• Opera
• …
Certificate fuzzing

• To diversify the easily accessible X.509 certificates
  - Given certificates $\text{Cert} = \{\text{cert}_0, \text{cert}_1, \ldots, \text{cert}_n\}$, construct $\text{Cert}' = \{\text{cert}'_0, \text{cert}'_1, \ldots, \text{cert}'_n\}$ whose certificates are as diverse as possible

• Certificate mutation

37 mutators (mutative operators) are defined for mutating single certificates or certificate chains, see details in the paper
Problem Description

• Goal
  o A variety of certificates can be obtained and used for differential testing

• Solution
  o The process of certificate fuzzing needs to be guided
Outline

• Motivation & problem description
• Approach
  o MCMC sampling
  o Differential testing
• Evaluation
• Conclusion
Diversification

An MCMC process

Fitness function: to improve code coverage as high as possible
Markov chain Monte Carlo (MCMC)

• What is MCMC?
  o In statistics, MCMC methods are a class of algorithms for sampling from a probability distribution based on constructing a Markov chain that has the desired distribution as its equilibrium distribution (Src: wiki)
  o Random walk Monte Carlo methods
    • Metropolis–Hastings algorithm
    • Gibbs sampling
    • Slice sampling
    • Multiple-try Metropolis
    • Reversible-jump
    • ...

Algorithm 1: MCMC-guided algorithm to optimize certificates

Input: certificate corpus CertStore, n, k
Output: test suite of n certificates
1: Select n random certificates from CertStore and add to Cert
2: highest_cov ← Cov(Cert)
3: OptimizedCerts ← {Cert}
4: repeat
5: cert ← random.choice(Cert)
6: mutator ← random.choice(Mutator)
7: cert′ ← apply(mutator, cert)
8: Cov′ ← Cov(Cert′)
9: if highest_cov < Cov(Cert′) then
10: highest_cov ← Cov(Cert′)
11: OptimizedCerts ← {Cert′}
12: else if highest_cov == Cov(Cert′) then
13: OptimizedCerts ← OptimizedCerts U {Cert′}
14: Accept Cert′ according to A(Cert → Cert′)
15: if accepted then
16: Cert′ ← Cert′
17: until highest_cov has not been increased for k steps
18: OptimizedCert ← random.choice(OptimizedCerts)
19: return OptimizedCert
MCMC-Guided Algorithm

- Cov(Cert) \ll Cov(OptimizedCert) \leq T
- An MCMC sampling process
  - It generates a sequence of samples, each of which can be the test suite
  - The desired distribution:

\[ P(Cert) = \frac{1}{Z} \exp(-\beta(Cov(Cert) - \perp)) \]
### Differential Testing

- **Distinct discrepancy**
  - A discrepancy appears if a sequence is not all zeros or all ones
  - Two discrepancies can be classified into one category if their encoded results are equal
Outline

• Motivation & problem description
• Approach
  o MCMC sampling
  o Differential testing
• Evaluation
• Conclusion
Evaluated Methods

• Mucert
• Frankencert [Brubaker 2014]
• Randmut: a random mutation algorithm
• Greedymut: a greedy mutation algorithm

• Two heuristics
  o SOSH: Stmt. Cov. Optimized Search
  o BOSH: Branch Cov. Optimized Search
Results on Certificate Generation

1K mucerts achieve up to 25% higher normalized coverage than 1K frankencerts; 200 mucerts achieve higher coverage than 100K frankencerts.
Results on Discrepancy Analysis

1K mucerts trigger up to 14.5X as many discrepancies and 5.6X as many distinct discrepancies as 1K frankencerts; 1K mucerts trigger up to 2.2X as many distinct discrepancies as 100K frankencerts
Mucerts achieve 28.7% precision and 3.0% diversity; all distinct discrepancies found by frankencerts and randmut/greedymut certificates are also found by mucerts.
Correlation between Coverage and Discrepancies

Mucert can stochastically diversify a test suite even when it does not further increase test coverage
Bug Reports and Developer Feedback

- Certificate validation discrepancies are prevalent

- Developers do have many doubts and concerns when implementing certificate validation code

<table>
<thead>
<tr>
<th></th>
<th>OpenSSL</th>
<th>PolarSSL</th>
<th>Gnutils</th>
<th>NSS</th>
<th>CyaSSL</th>
<th>MatrixSSL</th>
<th>Firefox</th>
<th>Chrome</th>
<th>IE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accepted</td>
<td>120</td>
<td>344</td>
<td>10</td>
<td>9</td>
<td>262</td>
<td>299</td>
<td>308</td>
<td>2</td>
<td>109</td>
</tr>
<tr>
<td>Not parsable</td>
<td>/</td>
<td>12</td>
<td>12</td>
<td>38</td>
<td>/</td>
<td>23</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>Rejected</td>
<td>237</td>
<td>1</td>
<td>335</td>
<td>310</td>
<td>95</td>
<td>35</td>
<td>49</td>
<td>355</td>
<td>248</td>
</tr>
</tbody>
</table>
Bug Reports and Developer Feedback (cont.)

• A lot of certificates can violate RFCs, but developers cannot reject all of them unless they are certainly security related.

• A certificate has two instances of the SubjectKeyIdentifier.
  o This finding has triggered an open question on IETF PKIX: Is it necessary to match AKI/SKI certificate extensions during certificate validation?
  o “Suggested replacement text for RFC5280, section 4.2.1.1,” [link](http://www.ietf.org/mail-archive/web/pkix/current/msg33248.html)

• Other issues
Outline

- Motivation & problem description
- Approach
  - MCMC sampling
  - Differential testing
- Evaluation
- Conclusion
Conclusion

- Problem formulation
- MCMC-guided certificate mutation
- Implementation and evaluation
- Community feedback and impact

- Mucert is available at: http://stap.sjtu.edu.cn/~chenyt/mucert.html
- More details are given in Yuting Chen, Zhendong Su: Guided differential testing of certificate validation in SSL/TLS implementations. ESEC/SIGSOFT FSE 2015: 793-804