Revisiting SSL/TLS Implementations
New Bleichenbacher Side Channels and Attacks

Christopher Meyer, Juraj Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, Erik Tews
TLS
... is Like Intelligent Glas
TLS

... because Privacy and Confidentiality Matters....

[ Source http://www.cafe-reichard.de ]
Oracle Attacks Are Back Again
Exploiting SSL/TLS Stacks

[ Source http://thebruinclub.files.wordpress.com/ ]
Bleichenbacher’s Attack
Restoring the Key Material
Bleichenbacher’s Attack
Restoring the Key Material

- Preconditions
  - RSA-Based Key Exchange
  - Previously Recorded SSL/TLS Session
  - Connection to Target Server Possible
Bleichenbacher’s Attack
Restoring the Key Material
Handshake Protocol Recap
Handshake Protocol Recap
Handshake Protocol
Content of a ClientKeyExchange Message (RSA Case)

Encrypted Payload when using RSA for Key Exchange

PMS = PreMasterSecret
Key Derivation
Why is the PMS so Sensitive?

![Diagram showing key derivation process]

- Known
- Unknown

- Initial Channel specific values:
  - Client random value
  - Server random value

- Session specific value:
  - PreMasterSecret
  - PRF

- "master secret"

- MasterSecret

- Server random value

- "key expansion"

- PRF

- Client random value

- Key Block

- Channel specific values
Handshake Protocol
Content of a *ClientKeyExchange* Message (RSA Case)

```
00 02 Padding != 00 00 03 01 Random
```

Encrypted Payload when using RSA for Key Exchange

Structure Violations Cause Different Response Messages!
Bleichenbacher’s Attack
Restoring the Key Material

Modified Message

TLS Handshake

Response

Structure OK?
YES: Continue with Handshake
NO: Send Error Message

Message Structure OK/NOT OK

Adjust Interval of Possible Values
-> Modify Message

...
Bleichenbacher’s Attack
Adjusting the ClientKeyExchange Payload
Bleichenbacher’s Attack
Adjusting the ClientKeyExchange Payload

- RSA Homomorphic-Property

\[ enc_{n, e}(m_1) \times enc_{n, e}(m_2) \]

\[ = enc_{n, e}(m_1 \times m_2) \]
Bleichenbacher’s Attack
Adjusting the ClientKeyExchange Payload

- RSA Homomorphic-Property

\[ \text{enc}_{n, e}(m_1) \cdot \text{enc}_{n, e}(m_2) \]

\[ = \text{enc}_{n, e}(m_1 \cdot m_2) \]

- Explanation:

\[ \text{enc}_{n, e}(m_1) \cdot \text{enc}_{n, e}(m_2) \]

\[ \equiv m_1^e \cdot m_2^e \mod n \equiv (m_1 \cdot m_2)^e \mod n \]

\[ = \text{enc}_{n, e}(m_1 \cdot m_2) \]
Bleichenbacher’s Attack
Adjusting the ClientKeyExchange Payload

- s: Value Chosen by Attacker
- c: Eavesdropped, Encrypted Content of a ClientKeyExchange Msg
- m: (Unknown) Plainext of c

\[ \text{enc}_{n, e}(s) * c \equiv s^e * c \mod n \]
\[ \equiv (s * m)^e \mod n \]
\[ = \text{enc}_{n, e}(s * m) \]
Bleichenbacher’s Attack
Restoring the Key Material

TLS 1.0 Defines Countermeasures
Bleichenbacher’s Attack
Restoring the Key Material

TLS 1.0 Defines Countermeasures

Create a random $\text{PMS}_R$ if there’s anything wrong with the one we received and continue with $\text{PMS}_R$
Bleichenbacher’s Attack
New Side-Channels
Bleichenbacher’s Attack
New Side-Channels

- Force Different Error Messages
Bleichenbacher’s Attack
New Side-Channels

- Force Different Error Messages
  - Implementation Bug
Bleichenbacher’s Attack
New Side-Channels

- Force Different Error Messages
  - Implementation Bug

```java
1 for (int i = 0; i < secLen; i++) {
2     pad1[i] ^= secret[i + secOff];
3     pad2[i] ^= secret[i + secOff];
4 }
```

Listing 8.1: Internal loop causing an Exception in special cases – Source: com.sun.crypto.provider.TlsPrfGenerator
Bleichenbacher’s Attack
New Side-Channels

- Force Different Error Messages
  - Implementation Bug

Listing 8.1: Internal loop causing an Exception in special cases – Source:
com.sun.crypto.provider.TlsPrfGenerator

```java
1 for (int i = 0; i < secLen; i++) {
2     pad1[i] ^= secret[i + secOff];
3     pad2[i] ^= secret[i + secOff];
4 }
```

ArrayIndexOutOfBoundsException causes an
INTERNAL_ERROR alert instead of HANDSHAKE_FAILURE
Bleichenbacher’s Attack
JSSE – Different Error Messages

Root Cause of Bug: com.sun.crypto.provider.TlsPrfGenerator.expand(..)
Real-World Bleichenbacher

~ 12 hours and ~177k queries later....
Java Total Breach of RSA-based TLS

Internal Flaws Resurrected Bleichenbacher’s Attack

Pre Master Secret

03 01 is protocol version of TLS 1.0
Bleichenbacher’s Attack
New Side-Channels

- Force Different Error Messages
  - Implementation Bug
- Force Different Processing Time
Measurement Setup
How to Measure Different Processing Times
Bleichenbacher’s Attack
New Side-Channels

- Force Different Error Messages
  - Implementation Bug
- Force Different Processing Time
  - Random Number Generation
Bleichenbacher’s Attack
New Side-Channels

- Force Different Error Messages
  - Implementation Bug
- Force Different Processing Time
  - Random Number Generation

*Only create a random PMS_R if there’s anything wrong with the received one and continue with PMS_R*

VS.

*Always create a random PMS_R, but only continue with PMS_R if there’s anything wrong with the received one*
Bleichenbacher’s Attack
OpenSSL – Different Processing Times

Figure 8.4.: Timing measurement results for OpenSSL 0.98. The valid secret refers to a TLS compliant ciphertext with correct PreMasterSecret. The invalid secret refers to a non-PKCS#1 v1.5 compliant ciphertext. In the non-PKCS#1 v1.5 compliant structure the first byte (which should be 00) was altered to 08 to provoke a random number generation on server-side.

~ 1,5 -2 µs
Bleichenbacher’s Attack
New Side-Channels

- Force Different Error Messages
  - Implementation Bug
- Force Different Processing Time
  - Random Number Generation

No attack possible (so far) timing differences very small and oracle due to strict PKCS#1 checks very weak
Bleichenbacher’s Attack
New Side-Channels

- Force Different Error Messages
  - Implementation Bug
- Force Different Processing Time
  - Random Number Generation
  - Exceptions
Bleichenbacher’s Attack
JSSE – Exceptions Cause Different Processing Time

- Unpadding Function Used to Remove PKCS#1 Overhead

```java
private byte[] unpadV15 (byte[] padded) throws BadPaddingException {
    if (not PKCS compliant) {
        throw new BadPaddingException();
    } else {
        return unpadded text;
    }
}
```
Bleichenbacher’s Attack

JSSE – Exceptions Cause Different Processing Time

Figure 8.5.: Timing measurement results for Java 1.7 (JSSE). The valid secret refers to a PKCS#1 compliant ciphertext. The invalid secret refers to a non-PKCS#1 compliant ciphertext. In the non-PKCS#1 compliant structure the first byte (which should be 0x00) was altered to 0x08 to provoke an BadPaddingException on the server.

~ 20 µs
Bleichenbacher’s Attack

JSSE – Exceptions Cause Different Processing Time

- Optimized Algorithm (Bardou et al.)
- 2048 bit
- Strong Oracle (no TLS Version Check)
- Optimized PKCS#1 v1.5 Structure to Speed up Algorithm

~ 19.5 hours and ~18,6k queries
Bleichenbacher’s Attack
New Side-Channels

- Force Different Error Messages
  - Implementation Bug
- Force Different Processing Time
  - Random Number Generation
  - Exceptions
- Special Behaviour
Bleichenbacher’s Attack
Cavium – Misbehaving Hardware Accelerators

- Hardware did not check the first byte’s value

```
00 02 Padding ! = 00 00 03 01 Random
01
02
03
...
```

48 Bytes PMS
Bleichenbacher’s Attack
Cavium – Misbehaving Hardware Accelerators

Figure 11: Timing measurement results for our IBM Datapower. The valid secret refers to a message, which starts with 0x??02, where 0x?? indicates an arbitrary byte. The invalid secret refers to a message starting with different bytes.
Bleichenbacher’s Attack
Cavium – Misbehaving Hardware Accelerators

- Missing First Byte Check Requires Algorithm Changes (Oracle is Much Weaker)
- 2048 bit
  ~ 40 hours and ~7371 queries
## Result Summary

<table>
<thead>
<tr>
<th>TLS impl.</th>
<th>Side channel</th>
<th>Queries &amp; Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenSSL</td>
<td>timing</td>
<td>$O(2^{40})$</td>
</tr>
<tr>
<td>JSSE</td>
<td>error message</td>
<td>177,000</td>
</tr>
<tr>
<td>JSSE</td>
<td>timing</td>
<td>18,600</td>
</tr>
<tr>
<td>Cavium</td>
<td>timing</td>
<td>7371</td>
</tr>
</tbody>
</table>
Revisiting SSL/TLS Implementations - New Bleichenbacher Side Channels and Attacks

Chris Meyer

christopher.meyer@rub.de
@armoredbarista
armoredbarista.blogspot.com