CVE-2017-1000385 vulnerability

Kaisa Jürmann

Supervisor: Meelis Roos

Tartu 2018
Table of Contents

Introduction...............................................................................................................................................3
  History of Bleichenbacher attack........................................................................................................3
Description of vulnerability.....................................................................................................................4
Vulnerable versions and fix......................................................................................................................4
  Simple workaround.............................................................................................................................4
  Actual fix.............................................................................................................................................5
References................................................................................................................................................7
Introduction

General-purpose programming language Erlang is used to build big expandable real-time systems, where most important factor is high availability. It it used in e-commerce, banking and instant messaging. Erlang's runtime system has built-in support for concurrency, distribution and fault tolerance. OTP is a set of Erlang libraries and design principles which help to develop the system. OTP includes its own debugging tools, distributed database and applications to interact with other languages.

The Erlang SSL application implements the SSL/TLS protocol. To authenticate, the client and server perform a TLS handshake procedure, where they agree on protocol version and cryptographic algorithms and before transmitting any data.

In the beginning of December 2017, Hanno Böck and Juraj Somorovsky and Craig Young published an article called "Return Of Bleichenbacher's Oracle Threat". In the article they revealed that by using some slight variations the so-called Bleichenbacher attack can still be used against many HTTPS hosts. This includes the vulnerability CVE-2017-1000385.

History of Bleichenbacher attack

Adaptive-chosen ciphertext attack is a form of chosen-ciphertext attack where attacker can gather information by collecting the decryptions of chosen ciphertexts. From this information the attacker tries to recover the hidden secret key used for decryption.

In 1998, Daniel Bleichenbacher discovered that the error messages given by SSL servers for errors in the PKCS #1 v1.5 padding allowed an adaptive-chosen ciphertext attack. This attack fully breaks the confidentiality of TLS when used with RSA encryption. It is described in detail in the paper “Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1” by Daniel Bleichenbacher.
Description of vulnerability

An Erlang TLS server configured with cipher suites using RSA may be vulnerable to an adaptive-chosen ciphertext attack (Bleichenbacher attack). The Erlang OTP TLS server answers with different TLS alerts to different error types in the RSA PKCS #1 1.5 padding. This may culminate in plain-text recovery of encrypted messages or a man-in-the-middle attack (sign messages with the server's private key), even if the attacker has not gained access to the server's private key.

Exploiting this to perform a man-in-the-middle attack requires the attacker to complete the initial attack, which may involve thousands of server requests, during the handshake stage (within the period of the handshake timeout) of the targeted session. The limitations of time and bandwidth make this attack undoubtedly more difficult to successfully execute.

RSA key exchange is enabled by default, but with low priority in the server order. For it to be actually chosen, it pretty much needs to be the only shared (also supported by the client) cipher suite. Captured TLS sessions encrypted with ephemeral cipher suites (DHE or ECDHE) are not at risk due to this specific vulnerability.

Vulnerable versions and fix

Simple workaround:

There also a simple workaround if default cipher suite configuration is used. It is possible to configure the server to not use vulnerable suites with the ciphers:

```
{ciphers, [Suite || Suite <- ssl:cipher_suites(),
             element(1,Suite) /= rsa]}
```

Then the code will look something like this:

```
ssl:listen(Port, [{ciphers, [Suite || Suite <-
                   ssl:cipher_suites(), element(1,S) /= rsa] |
                  Options}]).
```
**Actual fix:**

This vulnerability has known fix and already implemented in most versions.

Vulnerable and fixed packages:

<table>
<thead>
<tr>
<th>Source Package</th>
<th>Release</th>
<th>Version</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erlang (PTS)</td>
<td>wheezy</td>
<td>1:15.b.1- dfsg- 4+ deb7u1</td>
<td>vulnerable</td>
</tr>
<tr>
<td></td>
<td>wheezy (security)</td>
<td>1:15.b.1- dfsg- 4+ deb7u2</td>
<td>fixed</td>
</tr>
<tr>
<td></td>
<td>jessie</td>
<td>1:17.3- dfsg- 4+ deb8u1</td>
<td>vulnerable</td>
</tr>
<tr>
<td></td>
<td>jessie (security)</td>
<td>1:17.3- dfsg- 4+ deb8u2</td>
<td>fixed</td>
</tr>
<tr>
<td></td>
<td>stretch (security), stretch</td>
<td>1:19.2.1+ dfsg- 2+ deb9u1</td>
<td>fixed</td>
</tr>
<tr>
<td></td>
<td>buster</td>
<td>1:20.3.3+ dfsg- 1</td>
<td>fixed</td>
</tr>
<tr>
<td></td>
<td>sid</td>
<td>1:20.3.4+ dfsg- 1</td>
<td>fixed</td>
</tr>
</tbody>
</table>

Steps if RSA is being used for key agreement and authentication:

- client generates a 48-byte premaster secret
- encrypting it (using the public key from the server’s certificate)
- sending the result as an encrypted premaster secret message

This structure is an alternative of the ClientKeyExchange message and is not a message in itself.

Implementation of this vulnerability fix is that TLS serves should not generate an alert if processing an RSA-encrypted premaster secret message fails, or the version number is not as it was expected. Instead it must always provide some kind of premaster secret (it can be randomly generated then) to finish the handshake. The handshake can fail later.

The real failure cause may be logged when it’s useful, but it must be kept secure to avoid leaking this information to an attacker.

Below is the commit with the specific fix that was created for OTP-20.1.7 on Nov 7th 2017. For other versions, the implementation is similar.
handle_client_hello(
    #client_hello(client_version = ClientVersion) = Hello,
    State = prepare_flight(
        State0, Stateconnection_states = ConnectionStates,
        negotiated_version = Version,
        client_hello_version = ClientVersion,
        hashsign_algorithm = HashSign,
        session = Session,
        negotiated_protocol = Protocol)
)

server_certify_and_key_exchange(
    State0, Connection) ->
    request_client_cert(State2, Connection).
    certify_client_key_exchange(
        #encrypted_premaster_secret(premastersecret = EncPMS),
        State = Prestate,
        client_hello_version = #Major, Minor = Version)
    Prestate =
    Prestate2 =
    PremasterSecret =
    try ssl_handshake:
        PremasterSecret =
        EncPMS, Key of
        Secret when erlang:byte_size(Secret) == TNUM_OF_PREMASTERSECRET_BYTES ->
            case Secret of
                <<?BYTE(Major), ?BYTE(Minor), _Binary>> -> % Correct
                Secret;
                <<?BYTE(_), ?BYTE(_), Rest/binary>> -> % Version mismatch
                <<?BYTE(Major), ?BYTE(Minor), Rest/binary>>
                end;
                _ when erlang:byte_size(Secret) /= TNUM_OF_PREMASTERSECRET_BYTES
                ->
                make_premaster_secret(Version, rsa)
                catch
                #alert(description = ?DECRYPT_ERROR) ->
                make_premaster_secret(Version, rsa)
            end,
    TNUM_OF_PREMASTERSECRET_BYTES
    calculate_master_secret(PremasterSecret, State, Connection, certify, cipher);

session_cache_cb :: atom(),
crl_db :: term(),
negotiated_version :: ssl_record:ssl_version() | 'undefined',
client_hello_version :: ssl_record:ssl_version() | 'undefined',
client_certificate_requested = false :: boolean(),
key_algorithm :: ssl_cipher:key_algor(),
hashsign_algorithm = (undefined, undefined),
gen_handshake(ssl_connection, hello, internal, {
    common_client_hello, Type, ServerHelloExt,
    State = Stateconnection_states = ConnectionStates,
    negotiated_version = Version,
    client_hello_version = ClientVersion,
    hashsign_algorithm = HashSign,
    session = Session,
    negotiated_protocol = Protocol})
References

https://security-tracker.debian.org/tracker/CVE-2017-1000385
https://www.erlang.org/
https://github.com/erlang/otp
https://robotattack.org/
https://en.wikipedia.org/wiki/Chosen-ciphertext_attack
http://archiv.infsec.ethz.ch/education/fs08/secsem/bleichenbacher98.pdf