**Problem**

It was found that wolfssl before 3.15.7 is vulnerable to a new variant of the Bleichenbacher attack to perform downgrade attacks against TLS. This may lead to leakage of sensible data.

<https://eyalro.net/project/cat.html>

Talk on the article : <https://www.youtube.com/watch?v=aC5_XtM-lZI&t=70s&ab_channel=RealWorldCrypto>

Article : <https://eprint.iacr.org/2018/1173.pdf>

Fix : <https://github.com/wolfSSL/wolfssl/pull/1950>

**rsa.c**

static int RsaUnPad(const byte \*pkcsBlock, unsigned int pkcsBlockLen, byte \*\*output, byte padValue) {

**word32 maxOutputLen = (pkcsBlockLen > 10) ? (pkcsBlockLen - 10) : 0;**

word32 invalid = 0;

word32 i = 1;

word32 outputLen;

if (output == NULL || pkcsBlockLen == 0) {

return BAD\_FUNC\_ARG;

}

if (pkcsBlock[0] != 0x0) { /\* skip past zero \*/

invalid = 1;

}

pkcsBlock++; pkcsBlockLen--;

/\* Require block type padValue \*/

invalid = (pkcsBlock[0] != padValue) || invalid;

/\* verify the padding until we find the separator \*/

if (padValue == RSA\_BLOCK\_TYPE\_1) {

while (**i<pkcsBlockLen** && pkcsBlock[i++] == 0xFF) {/\* Null body \*/}

} else {

while (i<pkcsBlockLen && pkcsBlock[i++]) {/\* Null body \*/}

}

if (!(**i==pkcsBlockLen** || pkcsBlock[i-1]==0)) {

WOLFSSL\_MSG("RsaUnPad error, bad formatting");

return RSA\_PAD\_E;

}

outputLen = pkcsBlockLen - i;

invalid = (outputLen > maxOutputLen) || invalid;

if (invalid) {

WOLFSSL\_MSG("RsaUnPad error, invalid formatting");

return RSA\_PAD\_E;

}

\*output = (byte \*)(pkcsBlock + i);

return outputLen;

}

**Fix**

**rsa.c**

static int RsaUnPad(const byte \*pkcsBlock, unsigned int pkcsBlockLen, byte \*\*output, byte padValue) {

int ret;

word32 i;

byte invalid = 0;

if (output == NULL || pkcsBlockLen == 0) {

return BAD\_FUNC\_ARG;

}

if (padValue == RSA\_BLOCK\_TYPE\_1) {

/\* **First byte must be 0x00 and Second byte, block type, 0x01** \*/

if (pkcsBlock[0] != 0 || pkcsBlock[1] != RSA\_BLOCK\_TYPE\_1) {

WOLFSSL\_MSG("RsaUnPad error, invalid formatting");

return RSA\_PAD\_E;

}

/\* check the padding until we find the separator \*/

for (i = 2; i < pkcsBlockLen && pkcsBlock[i++] == 0xFF; ) { }

**/\*** **Minimum of 11 bytes of pre-message data and must have separator. \*/**

if (**i < RSA\_MIN\_PAD\_SZ** || pkcsBlock[i-1] != 0) {

WOLFSSL\_MSG("RsaUnPad error, bad formatting");

return RSA\_PAD\_E;

}

**\*output = (byte \*)(pkcsBlock + i);**

ret = pkcsBlockLen - i;

} else {

word32 j;

byte pastSep = 0;

/\* Decrypted with private key - unpad must be constant time. \*/

for (i = 0, j = 2; j < pkcsBlockLen; j++) {

/\* Update i if not passed the separator and at separator. \*/

i |= (~pastSep) & ctMaskEq(pkcsBlock[j], 0x00) & (j + 1);

pastSep |= ctMaskEq(pkcsBlock[j], 0x00);

}

/\* **Minimum of 11 bytes of pre-message data - including leading 0x00**. \*/

invalid |= ctMaskLT(i, **RSA\_MIN\_PAD\_SZ**);

/\* Must have seen separator. \*/

invalid |= ~pastSep;

/\* First byte must be 0x00. \*/

invalid |= ctMaskNotEq(pkcsBlock[0], 0x00);

/\* Check against expected block type: padValue \*/

invalid |= ctMaskNotEq(pkcsBlock[1], padValue);

**\*output = (byte \*)(pkcsBlock + i);**

ret = ((int)~invalid) & (pkcsBlockLen - i);

}

return ret;

}