**Problem**

The OpenSSL 3.0 implementation of the RC4-MD5 ciphersuite incorrectly uses the AAD data as the MAC key. This makes the MAC key trivially predictable. An attacker could exploit this issue by performing a man-in-the-middle attack to modify data being sent from one endpoint to an OpenSSL 3.0 recipient such that the modified data would still pass the MAC integrity check. Note that data sent from an OpenSSL 3.0 endpoint to a non-OpenSSL 3.0 endpoint will always be rejected by the recipient and the connection will fail at that point. Many application protocols require data to be sent from the client to the server first. Therefore, in such a case, only an OpenSSL 3.0 server would be impacted when talking to a non-OpenSSL 3.0 client. If both endpoints are OpenSSL 3.0 then the attacker could modify data being sent in both directions. In this case both clients and servers could be affected, regardless of the application protocol. Note that in the absence of an attacker this bug means that an OpenSSL 3.0 endpoint communicating with a non-OpenSSL 3.0 endpoint will fail to complete the handshake when using this ciphersuite. **The confidentiality of data is not impacted by this issue**, i.e., an attacker cannot decrypt data that has been encrypted using this ciphersuite - **they can only modify it (integrity)**. In order for this attack to work both endpoints must legitimately negotiate the RC4-MD5 ciphersuite. This ciphersuite is not compiled by default in OpenSSL 3.0 and is not available within the default provider or the default ciphersuite list. This ciphersuite will never be used if TLSv1.3 has been negotiated. In order for an OpenSSL 3.0 endpoint to use this ciphersuite the following must have occurred: 1) OpenSSL must have been compiled with the (non-default) compile time option enable-weak-ssl-ciphers 2) OpenSSL must have had the legacy provider explicitly loaded (either through application code or via configuration) 3) The ciphersuite must have been explicitly added to the ciphersuite list 4) The libssl security level must have been set to 0 (default is 1) 5) A version of SSL/TLS below TLSv1.3 must have been negotiated 6) Both endpoints must negotiate the RC4-MD5 ciphersuite in preference to any others that both endpoints have in common Fixed in OpenSSL 3.0.3 (Affected 3.0.0,3.0.1,3.0.2).

A copy&paste error meant that the RC4-MD5 cipher (used in TLS) used the TLS

AAD data as the MAC key.

<https://git.openssl.org/gitweb/?p=openssl.git;a=commitdiff;h=7d56a74a96828985db7354a55227a511615f732b>

**cipher\_rc4\_hmac\_md5.c**

static int rc4\_hmac\_md5\_set\_ctx\_params(void \*vctx, const OSSL\_PARAM params[])

{

PROV\_RC4\_HMAC\_MD5\_CTX \*ctx = (PROV\_RC4\_HMAC\_MD5\_CTX \*)vctx;

const OSSL\_PARAM \*p;

size\_t sz;

if (params == NULL)

return 1;

p = OSSL\_PARAM\_locate\_const(params, OSSL\_CIPHER\_PARAM\_KEYLEN);

if (p != NULL) {

if (!OSSL\_PARAM\_get\_size\_t(p, &sz)) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_FAILED\_TO\_GET\_PARAMETER);

return 0;

}

if (ctx->base.keylen != sz) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_INVALID\_KEY\_LENGTH);

return 0;

}

}

p = OSSL\_PARAM\_locate\_const(params, OSSL\_CIPHER\_PARAM\_IVLEN);

if (p != NULL) {

if (!OSSL\_PARAM\_get\_size\_t(p, &sz)) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_FAILED\_TO\_GET\_PARAMETER);

return 0;

}

if (ctx->base.ivlen != sz) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_INVALID\_IV\_LENGTH);

return 0;

}

}

p = OSSL\_PARAM\_locate\_const(params, OSSL\_CIPHER\_PARAM\_AEAD\_TLS1\_AAD);

if (p != NULL) {

if (p->data\_type != OSSL\_PARAM\_OCTET\_STRING) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_FAILED\_TO\_GET\_PARAMETER);

return 0;

}

sz = GET\_HW(ctx)->tls\_init(&ctx->base, p->data, p->data\_size);

if (sz == 0) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_INVALID\_DATA);

return 0;

}

ctx->tls\_aad\_pad\_sz = sz;

}

p = OSSL\_PARAM\_locate\_const(params, **OSSL\_CIPHER\_PARAM\_AEAD\_TLS1\_AAD**);

if (p != NULL) {

if (p->data\_type != OSSL\_PARAM\_OCTET\_STRING) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_FAILED\_TO\_GET\_PARAMETER);

return 0;

}

GET\_HW(ctx)->init\_mackey(&ctx->base, p->data, p->data\_size);

}

p = OSSL\_PARAM\_locate\_const(params, OSSL\_CIPHER\_PARAM\_TLS\_VERSION);

if (p != NULL) {

if (!OSSL\_PARAM\_get\_uint(p, &ctx->base.tlsversion)) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_FAILED\_TO\_GET\_PARAMETER);

return 0;

}

}

return 1;

}

**evpciph\_aes\_stitched.txt**

Title = RC4-HMAC-MD5 test vectors

Availablein = legacy

Cipher = RC4-HMAC-MD5

Key = d48ecc0a163a06626bd1b7e172dfb5b3

MACKey = 5973581f63768353af37d3f51ec9f6ef

TLSAAD = 90a1b2c3e4f506172803010050

TLSVersion = 0x0301

Plaintext = 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f

Ciphertext = eea8eba927d9b16c640958f922b3ca43b197eea520674aa1d059156dfd4c12249e2890e8f3c72676e20fe4a30848c1cc6c12f4596d6e290b5f84745ac36959645ea4acabc84e748b2fd5e4228a2fe4f8d44460dfb9a0fce1faf00f1fc7159c3c

Operation = ENCRYPT

Availablein = legacy

Cipher = RC4-HMAC-MD5

Key = d48ecc0a163a06626bd1b7e172dfb5b3

MACKey = 5973581f63768353af37d3f51ec9f6ef

TLSAAD = 90a1b2c3e4f506172803010060

TLSVersion = 0x0301

Plaintext = 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f

Ciphertext = eea8eba927d9b16c640958f922b3ca43b197eea520674aa1d059156dfd4c12249e2890e8f3c72676e20fe4a30848c1cc6c12f4596d6e290b5f84745ac36959645ea4acabc84e748b2fd5e4228a2fe4f8d44460dfb9a0fce1faf00f1fc7159c3c

Operation = DECRYPT

**evpciph\_rc4\_stitched.txt**

Ciphertext = eea8eba927d9b16c640958f922b3ca43b197eea520674aa1d059156dfd4c12249e2890e8f3c72676e20fe4a30848c1cc6c12f4596d6e290b5f84745ac36959645ea4acabc84e748b2fd5e4228a2fe4f8d44460dfb9a0fce1faf00f1fc7159c3c

Ciphertext = eea8eba927d9b16c640958f922b3ca43b197eea520674aa1d059156dfd4c12249e2890e8f3c72676e20fe4a30848c1cc6c12f4596d6e290b5f84745ac36959645ea4acabc84e748b2fd5e4228a2fe4f8d44460dfb9a0fce1faf00f1fc7159c3c

**Fix**

**evpciph\_aes\_stitched.txt**

**evpciph\_rc4\_stitched.txt**

Ciphertext = eea8eba927d9b16c640958f922b3ca43b197eea520674aa1d059156dfd4c12249e2890e8f3c72676e20fe4a30848c1cc6c12f4596d6e290b5f84745ac36959645ea4acabc84e748b2fd5e4228a2fe4f8c5792501fca9d8455160d626dc1a9716

Ciphertext = eea8eba927d9b16c640958f922b3ca43b197eea520674aa1d059156dfd4c12249e2890e8f3c72676e20fe4a30848c1cc6c12f4596d6e290b5f84745ac36959645ea4acabc84e748b2fd5e4228a2fe4f8c5792501fca9d8455160d626dc1a9716

**cipher\_rc4\_hmac\_md5.c**

static int rc4\_hmac\_md5\_set\_ctx\_params(void \*vctx, const OSSL\_PARAM params[])

{

PROV\_RC4\_HMAC\_MD5\_CTX \*ctx = (PROV\_RC4\_HMAC\_MD5\_CTX \*)vctx;

const OSSL\_PARAM \*p;

size\_t sz;

if (params == NULL)

return 1;

p = OSSL\_PARAM\_locate\_const(params, OSSL\_CIPHER\_PARAM\_KEYLEN);

if (p != NULL) {

if (!OSSL\_PARAM\_get\_size\_t(p, &sz)) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_FAILED\_TO\_GET\_PARAMETER);

return 0;

}

if (ctx->base.keylen != sz) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_INVALID\_KEY\_LENGTH);

return 0;

}

}

p = OSSL\_PARAM\_locate\_const(params, OSSL\_CIPHER\_PARAM\_IVLEN);

if (p != NULL) {

if (!OSSL\_PARAM\_get\_size\_t(p, &sz)) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_FAILED\_TO\_GET\_PARAMETER);

return 0;

}

if (ctx->base.ivlen != sz) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_INVALID\_IV\_LENGTH);

return 0;

}

}

p = OSSL\_PARAM\_locate\_const(params, OSSL\_CIPHER\_PARAM\_AEAD\_TLS1\_AAD);

if (p != NULL) {

if (p->data\_type != OSSL\_PARAM\_OCTET\_STRING) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_FAILED\_TO\_GET\_PARAMETER);

return 0;

}

sz = GET\_HW(ctx)->tls\_init(&ctx->base, p->data, p->data\_size);

if (sz == 0) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_INVALID\_DATA);

return 0;

}

ctx->tls\_aad\_pad\_sz = sz;

}

p = OSSL\_PARAM\_locate\_const(params, **OSSL\_CIPHER\_PARAM\_AEAD\_MAC\_KEY**);

if (p != NULL) {

if (p->data\_type != OSSL\_PARAM\_OCTET\_STRING) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_FAILED\_TO\_GET\_PARAMETER);

return 0;

}

GET\_HW(ctx)->init\_mackey(&ctx->base, p->data, p->data\_size);

}

p = OSSL\_PARAM\_locate\_const(params, OSSL\_CIPHER\_PARAM\_TLS\_VERSION);

if (p != NULL) {

if (!OSSL\_PARAM\_get\_uint(p, &ctx->base.tlsversion)) {

ERR\_raise(ERR\_LIB\_PROV, PROV\_R\_FAILED\_TO\_GET\_PARAMETER);

return 0;

}

}

return 1;

}