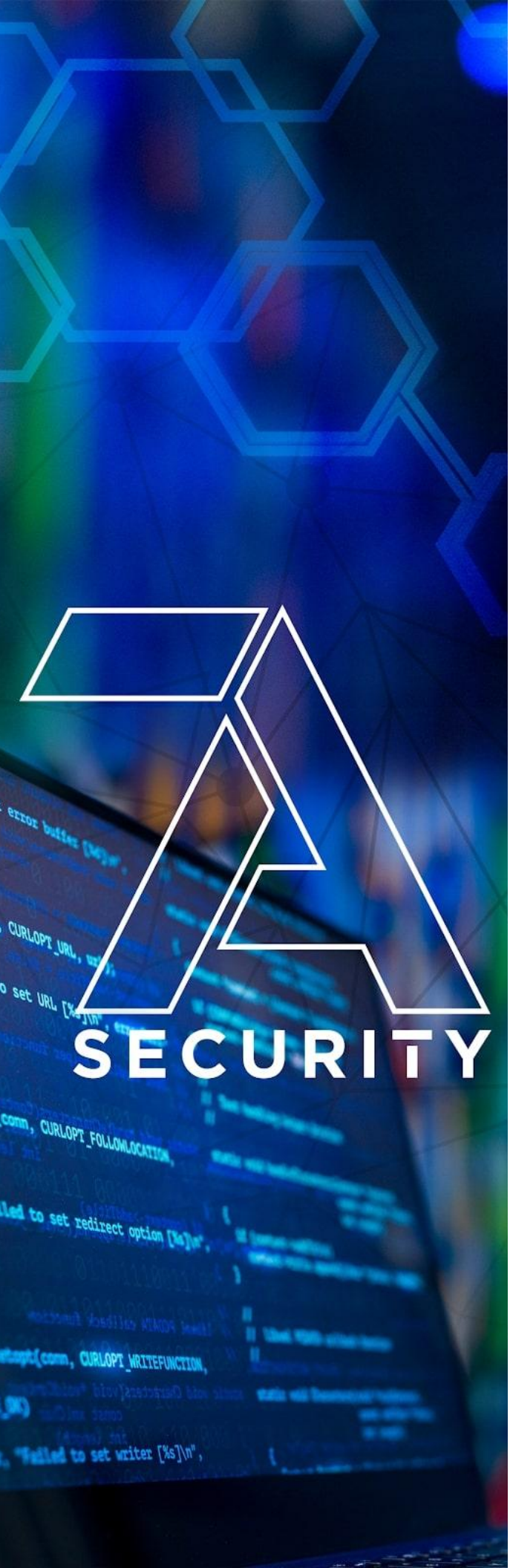

DEfO-2 Test Targets:

DEfO-2 OpenSSL HPKE PR

DEfO-2 Fuzzing

Pentest Report

Client:

DEfO-2

7ASecurity Test Team:

- Abraham Aranguren, MSc.
- Hannes Moesl-Canaval, MSc.
- Alexander Schloegl, MSc.
- Dominik Prodinger, BSc.
- David Gstir, Dipl.-Ing.
- Richard Weinberger
- Lorenz Kofler

7ASecurity

Protect Your Site & Apps

From Attackers

sales@7asecurity.com

7asecurity.com

INDEX

Introduction

"The encrypted ClientHello (ECH) mechanism (draft-spec) is a way to plug a few privacy-holes that remain in the Transport Layer Security (TLS) protocol that's used as the security layer for the web.[...]The DEfO project has developed an implementation of ECH for OpenSSL, and proof-of-concept implementations of various clients and servers that use OpenSSL as a demonstration and for interoperability testing."

From https://defo.ie/

This document outlines the results of a penetration test and *whitebox* security review conducted against the DEfO-2 OpenSSL HPKE PR[1]. The project was solicited by the DEfO-2 team, funded by the Open Technology Fund (OTF), and executed by 7ASecurity in October 2023. The audit team dedicated 49 working days to complete this assignment. Please note that, while this is the first penetration test for this project, the attack surface of the DEfO-2 OpenSSL changes is rather limited and, hence, identification of exploitable security weaknesses was particularly challenging during this assignment.

During this iteration the goal was to review the OpenSSL HPKE PR modifications as thoroughly as possible, to ensure internet users are not negatively impacted by the changes DEfO-2 introduced to eliminate the previously unavoidable privacy leak in OpenSSL[2].

The methodology implemented was *whitebox*: 7ASecurity was provided with access to reference client and server implementations, documentation and source code. A team of 7 senior auditors carried out all tasks required for this engagement, including preparation, delivery, documentation of findings and communication.

A number of necessary arrangements were in place by September 2023, to facilitate a straightforward commencement for 7ASecurity. In order to enable effective collaboration, information to coordinate the test was relayed through email, as well as a shared Element channel. The DEfO-2 team was helpful and responsive at all times, which facilitated the test for 7ASecurity, without introducing any unnecessary delays. 7ASecurity provided regular updates regarding the audit status and its interim findings during the engagement.

This audit split the scope items in the following work packages, which are referenced in the ticket headlines as applicable:

[1] https://github.com/openssl/openssl/pull/17172
[2] https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni

- WP1: Audit of DEfO-2 OpenSSL HPKE PR changes
- WP2: Audit of the OpenSSL master branch areas affected by DEfO-2 PR
- WP3: Code-Fuzzing of OpenSSL HPKE PR changes
- WP4: Network-traffic Fuzzing of OpenSSL HPKE PR changes

The findings of the security audit (WP1-4) can be summarized as follows:

Identified Vulnerabilities	*Hardening Recommendations*	*Total Issues*
0	*10*	*10*

Moving forward, the scope section elaborates on the items under review, while the findings section documents the identified vulnerabilities followed by hardening recommendations with lower exploitation potential. Each finding includes a technical description, a proof-of-concept (PoC) and/or steps to reproduce if required, plus mitigation or fix advice for follow-up actions by the development team.

Finally, the report culminates with a conclusion providing detailed commentary, analysis, and guidance relating to the context, preparation, and general impressions gained throughout this test, as well as a summary of the perceived security posture of the DEfO-2 OpenSSL improvements.

Scope

The following list outlines the items in scope for this project:

- **WP1: Audit of DEfO-2 OpenSSL HPKE PR changes**
  - OpenSSL PR: https://github.com/openssl/openssl/pull/17172
  - Documentation:
    - https://defo.ie
    - https://datatracker.ietf.org/doc/rfc9180/
- **WP2: Audit of the OpenSSL master branch areas affected by DEfO-2 PR**
  - OpenSSL master branch: https://github.com/openssl/openssl
  - Reviewed Commit: 91895e39b10033178e662fc7427a09d7562cf8e1
- **WP3: Code-Fuzzing of OpenSSL HPKE PR changes**
  - As Above
- **WP4: Network-traffic Fuzzing of OpenSSL HPKE PR changes**
  - As Above

Identified Vulnerabilities

This area of the report enumerates findings that were deemed to exhibit greater risk potential. Please note these are offered sequentially as they were uncovered, they are not sorted by significance or impact. Each finding has a unique ID (i.e. *DEF-01-001*) for ease of reference, and offers an estimated severity in brackets alongside the title.

No directly exploitable vulnerabilities could be identified during this assignment.

Hardening Recommendations

This area of the report provides insight into less significant weaknesses that might assist adversaries in certain situations. Issues listed in this section often require another vulnerability to be exploited, need an uncommon level of access, exhibit minor risk potential on their own, and/or fail to follow information security best practices. Nevertheless, it is recommended to resolve as many of these items as possible to improve the overall security posture and protect users in edge-case scenarios.

DEF-01-001 WP1: Missing Return Value Checks may result in Crashes *(Info)*

Note: Although this issue was found in the pull request undergoing the audit, it was later discovered that the weakness is already resolved in the OpenSSL master branch.

During the code audit of the HPKE pull request, it was identified that the pointer returned by the *ossl_HPKE_KEM_INFO_find_id* function is not checked for NULL. Please note the return value is adequately checked in most locations. However, the *OSSL_HPKE_get_recommended_ikmelen* function omits this check entirely. While this inconsistency does not pose a security vulnerability at the time of writing, it should be remediated to conform with related calls to the same function and reduce the potential for weaknesses in the future. For example, dereferencing a null pointer may result in an application crash. This issue can be observed in the following code path:

Affected File:
https://github.com/openssl/openssl/blob/50b3[...]/crypto/hpke/hpke.c#L1431

Affected Code:

```
size_t OSSL_HPKE_get_recommended_ikmelen(OSSL_HPKE_SUITE suite)
{
    const OSSL_HPKE_KEM_INFO *kem_info = NULL;

    if (hpke_suite_check(suite) != 1)
        return 0;
    kem_info = ossl_HPKE_KEM_INFO_find_id(suite.kem_id);
    return kem_info->Nsk;
}
```

It is recommended to consistently check return values throughout the codebase when making function calls. Please note that there may be other instances where return values are not adequately checked. For this reason, the DEfO-2 and OpenSSL teams

are encouraged to utilize automated and manual checks to consistently identify, report and resolve similar weaknesses.

DEF-01-002 WP3: Missing Public Key Size Check in *OSSL_HPKE_keygen* (Info)

Note: Although this issue was found in the pull request undergoing the audit, it was later discovered that the weakness is already resolved in the OpenSSL master branch.

While fuzzing the DEfO-2 OpenSSL HPKE PR changes, it was uncovered that the *OSSL_HPKE_keygen* function may perform buffer out-of-bounds writes for the public key in some scenarios. This could occur depending on the encoding length of the public key, if an insufficiently large buffer is passed. Please note that, since the affected input cannot be tainted by attackers at the time of writing, there are no security implications for this weakness. Nevertheless, this is still a bad practice that might result in potential issues in the future, depending on how people fork and expand the OpenSSL library.

Please note this behavior contradicts the official OpenSSL documentation, which states[3]:

"An error will occur if the input publen is too small."

This issue occurs because the size of the buffer must be at least *Npk* bytes. The reason for this can be found in section *7.1. Key Encapsulation Mechanisms (KEMs)* of *RFC 9180*, which specifies[4] concrete size requirements. Additionally, section *11.1. KEM Identifiers* of the aforementioned RFC defines the *Npk* parameter as:

"Npk: The length in bytes of an encoded public key for the algorithm"

The root cause for this issue can be found in the following code path, which fails to perform a size check before the public key is extracted by the *EVP_PKEY_get_octet_string_param* function:

Affected File:
https://github.com/openssl/openssl/blob/50b3[...]/crypto/hpke/hpke.c#L1317

Affected Code:

```
[...]

if (EVP_PKEY_generate(pctx, &skR) <= 0) {
  ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
  goto err;
}
```

[3] https://github.com/openssl/openssl/blob/master/doc/man3/OSSL_HPKE_CTX_new.pod
[4] https://datatracker.ietf.org/doc/rfc9180/

```
EVP_PKEY_CTX_free(pctx);
pctx = NULL;
if (EVP_PKEY_get_octet_string_param(skR, OSSL_PKEY_PARAM_ENCODED_PUBLIC_KEY, pub,
*publen, publen) != 1) {
  ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
  goto err;
}
*priv = skR;
erv = 1;
[...]
```

This was further confirmed at runtime using the following proof-of-concept:

PoC:

```
int fuzzerCrash0(const uint8_t *buf, size_t len) {
  if (len < 100)
       return 0;
  OSSL_HPKE_SUITE hpke_suite = {0x12, 0x02, 0xffff};
  size_t publen = 132;
  unsigned char *pub = malloc(publen);

  const unsigned char *ikm = NULL;
  size_t ikmlen = 0;
  OSSL_LIB_CTX *libctx = NULL;
  const char *propq = NULL;

  EVP_PKEY *priv;

  OSSL_HPKE_keygen(hpke_suite, pub, &publen, &priv, ikm, ikmlen, libctx,
                           propq);
  if (pub) {
       free(pub);
       pub = NULL;
  }
  if(priv)
       EVP_PKEY_free(priv);

  return 0;
}
```

Which resulted in the following crash:

Crash Output:

```
==31517==ERROR: AddressSanitizer: heap-use-after-free on address 0x602000000112 at pc
0x560f95ef50ec bp 0x7ffda23cae30 sp 0x7ffda23cae28
WRITE of size 1 at 0x602000000112 thread T0
```

```
        #0 0x560f95ef50eb in bn2binpad openssl/crypto/bn/bn_lib.c:605:13
        #1 0x560f95ef53f9 in BN_bn2bin openssl/crypto/bn/bn_lib.c:630:12
        #2 0x560f9604e352 in ossl_ec_GFp_simple_point2oct
openssl/crypto/ec/ecp_oct.c:234:16
        #3 0x560f95ffe663 in EC_POINT_point2oct openssl/crypto/ec/ec_oct.c:88:20
        #4 0x560f95d852f2 in common_get_params
openssl/providers/implementations/keymgmt/ec_kmgmt.c:738:26
        #5 0x560f95d7f6ae in ec_get_params
openssl/providers/implementations/keymgmt/ec_kmgmt.c:761:12
        #6 0x560f9617a0ee in evp_keymgmt_get_params
openssl/crypto/evp/keymgmt_meth.c:394:12
        #7 0x560f95b3bbfe in EVP_PKEY_get_params openssl/crypto/evp/p_lib.c:2369:20
        #8 0x560f95b388a3 in EVP_PKEY_get_octet_string_param
openssl/crypto/evp/p_lib.c:2189:17
        #9 0x560f95b72167 in OSSL_HPKE_keygen openssl/crypto/hpke/hpke.c:1309:9
        #10 0x560f95b1f467 in FuzzerTestOneInput openssl/fuzz/ossl_hpke_keygen.c:100:3
        #11 0x560f95b202a8 in testfile openssl/fuzz/test-corpus.c:55:9
        #12 0x560f95b1fd02 in main openssl/fuzz/test-corpus.c:96:13
        #13 0x7f2747957ccf  (/usr/lib/libc.so.6+0x27ccf) (BuildId:
8bfe03f6bf9b6a6e2591babd0bbc266837d8f658)
        #14 0x7f2747957d89 in __libc_start_main (/usr/lib/libc.so.6+0x27d89) (BuildId:
8bfe03f6bf9b6a6e2591babd0bbc266837d8f658)
        #15 0x560f959e8044 in _start (openssl/fuzz/ossl_hpke_keygen-test+0x169044)
(BuildId: b70fbf5bd74bb0e96676551261c4da39e7bd06bc)
```

The offending out-of-bounds write occurs in the heap, therefore this crash is classified by the *AdressSanitizer (ASAN)* as a *heap-use-after-free* vulnerability. Allocating the buffer on the stack would change the ASAN classification to a stack overflow bug.

It is recommended to check that *publen* fulfills at least the size requirements listed in RFC9180. Additionally, it was observed that the function call uses *publen* as both an input and an output parameter, which should be avoided.

DEF-01-003 WP1/3: Possible Segmentation Fault via Missing Checks *(Info)*

Retest Notes: The DEfO-2 team fixed[5] this issue and 7ASecurity confirmed that the fix is valid.

During the code audit and fuzzing of the HPKE extension within the OpenSSL library, it was discovered that the *OSSL_HPKE_encap* and *OSSL_HPKE_decap* functions fail to validate the *info* input parameter. Specifically, supplying a NULL pointer to *info* and a valid length to *infolen*, the execution results in a segmentation fault. Please note that, since the affected input cannot be tainted by attackers at the time of writing, there are no security implications for this weakness. The root cause for this issue can be confirmed reviewing the following code snippets:

Affected File:
https://github.com/openssl/openssl/blob/50b3[...]/crypto/hpke/hpke.c#L1070

Affected Code:
```
int OSSL_HPKE_encap(OSSL_HPKE_CTX *ctx,
                    unsigned char *enc, size_t *enclen,
                    const unsigned char *pub, size_t publen,
                    const unsigned char *info, size_t infolen)
{
    int erv = 1;

    if (ctx == NULL || enc == NULL || enclen == NULL || *enclen == 0
        || pub == NULL || publen == 0) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_NULL_PARAMETER);
        return 0;
    }
    if (infolen > OSSL_HPKE_MAX_INFOLEN) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_INVALID_ARGUMENT);
        return 0;
    }
    [...]
}
```

Affected File:
https://github.com/openssl/openssl/blob/50b3[...]/crypto/hpke/hpke.c#L1107

Affected Code:
```
int OSSL_HPKE_decap(OSSL_HPKE_CTX *ctx,
                    const unsigned char *enc, size_t enclen,
```

[5] https://github.com/openssl/openssl/pull/22493

```
                    EVP_PKEY *recippriv,
                    const unsigned char *info, size_t infolen)
{
    int erv = 1;

    if (ctx == NULL || enc == NULL || enclen == 0 || recippriv == NULL) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_NULL_PARAMETER);
        return 0;
    }
    if (infolen > OSSL_HPKE_MAX_INFOLEN) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_INVALID_ARGUMENT);
        return 0;
    }

    [...]
}
```

This was further confirmed at runtime using the following proof-of-concept:

PoC:

```
// [...]
OSSL_HPKE_encap(ctx, enc, &enclen, pub, publen, NULL, 20);
```

Which resulted in the following crash:

Crash Output:

```
==95468==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc
0x7f1077f62901 bp 0x7ffdafba0e70 sp 0x7ffdafba0638 T0)
==95468==The signal is caused by a READ memory access.
==95468==Hint: address points to the zero page.
    #0 0x7f1077f62901 in memcpy (/lib/x86_64-linux-gnu/libc.so.6+0xc4901) (BuildId:
69389d485a9793dbe873f0ea2c93e02efaa9aa3d)
    #1 0x557369f81301 in __asan_memcpy
(/workspaces/openssl/fuzz/hpke_roundtrip-test+0x2e4301) (BuildId:
8882758b6dbe6e3cc5d358f0a99e14dff2df2116)
    #2 0x55736a5d58fb in WPACKET_memcpy /workspaces/openssl/crypto/packet.c:470:9
    #3 0x55736a5a6442 in ossl_hpke_labeled_extract
/workspaces/openssl/crypto/hpke/hpke_util.c:324:17
    #4 0x55736a59a851 in hpke_do_middle /workspaces/openssl/crypto/hpke/hpke.c:718:9
    #5 0x55736a597f9a in OSSL_HPKE_encap /workspaces/openssl/crypto/hpke/hpke.c:1088:11
    #6 0x557369fbfb5e in fuzzerCrash0 /workspaces/openssl/fuzz/hpke_roundtrip.c:211:9
    #7 0x557369fbd917 in FuzzerTestOneInput
/workspaces/openssl/fuzz/hpke_roundtrip.c:233:5
    #8 0x557369fc0aac in testfile /workspaces/openssl/fuzz/test-corpus.c:55:9
    #9 0x557369fc054a in main /workspaces/openssl/fuzz/test-corpus.c:96:13
    #10 0x7f1077ec7d8f  (/lib/x86_64-linux-gnu/libc.so.6+0x29d8f) (BuildId:
69389d485a9793dbe873f0ea2c93e02efaa9aa3d)
```

```
    #11 0x7f1077ec7e3f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x29e3f)
(BuildId: 69389d485a9793dbe873f0ea2c93e02efaa9aa3d)
    #12 0x557369eff0a4 in _start
(/workspaces/openssl/fuzz/hpke_roundtrip-test+0x2620a4) (BuildId:
8882758b6dbe6e3cc5d358f0a99e14dff2df2116)
```

It is recommended to ensure the validity of both the *info* and *infolen* parameters before proceeding.

DEF-01-004 WP3: Possible Segmentation Fault via Buffer Overwrite *(Info)*

Retest Notes: The DEfO-2 team fixed[6] this issue and 7ASecurity confirmed that the fix is valid.

While fuzzing the OpenSSL HPKE PR changes, it was discovered that calling *OSSL_HPKE_encap* with return buffers that are too small to store the return values, will lead to a segmentation fault, even if the correct sizes are passed. Please note that, since the affected input cannot be tainted by attackers at the time of writing, there are no security implications for this weakness. Nevertheless, this is still a bad practice that might result in potential issues in the future, depending on how people fork and expand the OpenSSL library.

Please note this behavior contradicts the official OpenSSL documentation for the *OSSL_HPKE_get_public_encap_size* function, which states[7]:

> *"An error will occur if the input enclen is smaller than the value returned from OSSL_HPKE_get_public_encap_size()."*

This issue occurs because the *EVP_PKEY_encapsulate(pctx, NULL, enclen, NULL, &lsslen)* call will overwrite the value stored in *enclen* (which is a *size_t**), setting it to *Nenc=Npk*, the encoded public key length. Hence, the modified value will pass the size check in the subsequent call, leading to an out-of-bounds write and segmentation fault. Furthermore, the same problem exists for *OSSL_HPKE_decap*, where a similar call happens.

The root cause for this issue can be confirmed reviewing the following code paths:

Affected File:
https://github.com/openssl/openssl/blob/50b3[...]/crypto/hpke/hpke.c#L510

[6] https://github.com/openssl/openssl/pull/22493
[7] https://github.com/openssl/openssl/blob/master/doc/man3/OSSL_HPKE_CTX_new.pod

Affected Code:

```
[...]
*p = OSSL_PARAM_construct_end();
if (ctx->mode == OSSL_HPKE_MODE_AUTH
        || ctx->mode == OSSL_HPKE_MODE_PSKAUTH) {
        if (EVP_PKEY_auth_encapsulate_init(pctx, ctx->authpriv,
                                           params) != 1) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
        goto err;
        }
} else {
        if (EVP_PKEY_encapsulate_init(pctx, params) != 1) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
        goto err;
        }
}
if (EVP_PKEY_encapsulate(pctx, NULL, enclen, NULL, &lsslen) != 1) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
        goto err;
}
ctx->shared_secret = OPENSSL_malloc(lsslen);
if (ctx->shared_secret == NULL)
        goto err;
ctx->shared_secretlen = lsslen;
if (EVP_PKEY_encapsulate(pctx, enc, enclen, ctx->shared_secret,
                         &ctx->shared_secretlen) != 1) {

[...]
```

Affected File:
https://github.com/openssl/openssl/blob/50b3[...]/crypto/hpke/hpke.c#L605

Affected Code:

```
[...]
} else {
        if (EVP_PKEY_decapsulate_init(pctx, params) != 1) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
        goto err;
        }
}
if (EVP_PKEY_decapsulate(pctx, NULL, &lsslen, enc, enclen) != 1) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
        goto err;
}
ctx->shared_secret = OPENSSL_malloc(lsslen);
if (ctx->shared_secret == NULL)
        goto err;
if (EVP_PKEY_decapsulate(pctx, ctx->shared_secret, &lsslen,
```

```
                        enc, enclen) != 1) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
        goto err;
}
[...]
```

This was further confirmed at runtime using the following proof-of-concept:

PoC:

```
#include <openssl/err.h>
#include <openssl/evp.h>
#include <openssl/hpke.h>
#include <openssl/types.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>

int main() {
  int mode = 0;
  OSSL_HPKE_SUITE hpke_suite = {0x21, 0x1, 0x2};
  size_t enclen =0x17, publen =0xff, infolen =0x20;

  unsigned char *pub = malloc(publen);
  unsigned char *info = malloc(infolen);
  unsigned char *enc = malloc(enclen);

  EVP_PKEY *priv;
  OSSL_HPKE_keygen(hpke_suite, pub, &publen, &priv, NULL, 0, NULL, NULL);
  OSSL_HPKE_CTX *ctx = OSSL_HPKE_CTX_new(mode, hpke_suite, OSSL_HPKE_ROLE_SENDER, NULL,
NULL);
  OSSL_HPKE_encap(ctx, enc, &enclen, pub, publen, NULL, 0);

  if (pub) {
       free(pub);
       pub = NULL;
  }
  if (info) {
       free(info);
       info= NULL;
  }
  if (enc) {
       free(enc);
       enc = NULL;
  }
  return 0;
}
```

The above snippet was compiled and linked against OpenSSL. Note that without the *AdressSanitizer* employed by the AFL fuzzer only a segmentation fault will be encountered. This resulted in the following crash:

Crash Output:

```
==65334==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x603000012507 at pc
0x55e99665952f bp 0x7ffc7f2fcbf0 sp 0x7ffc7f2fc3b0
WRITE of size 56 at 0x603000012507 thread T0
       #0 0x55e99665952e in __asan_memcpy (openssl/fuzz/ossl_hpke_encap-test+0x25552e)
(BuildId: 10e84c1c628567aebc403d7a17bc06a4b7816d4b)
       #1 0x55e996b2b7fc in dhkem_encap
openssl/providers/implementations/kem/ecx_kem.c:588:5
       #2 0x55e996b297a6 in ecxkem_encapsulate
openssl/providers/implementations/kem/ecx_kem.c:665:20
       #3 0x55e99681e505 in EVP_PKEY_encapsulate openssl/crypto/evp/kem.c:235:12
       #4 0x55e9966b114a in hpke_encap openssl/crypto/hpke/hpke.c:562:9
       #5 0x55e9966afe58 in OSSL_HPKE_encap openssl/crypto/hpke/hpke.c:1076:9
       #6 0x55e9966a4772 in FuzzerTestOneInput openssl/fuzz/ossl_hpke_encap.c:97:9
       #7 0x55e9966a5848 in testfile openssl/fuzz/test-corpus.c:55:9
       #8 0x55e9966a52a2 in main openssl/fuzz/test-corpus.c:96:13
       #9 0x7f88176d8ccf  (/usr/lib/libc.so.6+0x27ccf) (BuildId:
8bfe03f6bf9b6a6e2591babd0bbc266837d8f658)
       #10 0x7f88176d8d89 in __libc_start_main (/usr/lib/libc.so.6+0x27d89) (BuildId:
8bfe03f6bf9b6a6e2591babd0bbc266837d8f658)
       #11 0x55e99656d044 in _start (openssl/fuzz/ossl_hpke_encap-test+0x169044)
(BuildId: 10e84c1c628567aebc403d7a17bc06a4b7816d4b)
```

In order to resolve this issue, the call to *EVP_PKEY_encapsulate* ought to be made in such a way that *enclen* is not overwritten and *enclen* is adequately sized. If the length is insufficient, an error should be returned to the caller.

DEF-01-005 WP1: Potential GREASE Fingerprinting via RNG Bias *(Low)*

Retest Notes: The DEfO-2 team fixed[8] this issue and 7ASecurity confirmed that the fix is valid.

While reviewing the HPKE generation code for GREASE[9] values, it was noticed that the random selection of KEM, AEAD and KDF can be slightly biased, which facilitates implementation fingerprinting. Malicious endpoints could potentially use this information to launch targeted attacks against a specific implementation or individual users.

OpenSSL provides the *OSSL_HPKE_get_grease_value* API function. This generates random HPKE-like structured data, to be used by protocols like TLS, to ensure implementations correctly reject unspecified extension identifiers. For HPKE, this requires generating a valid suite with identifiers for the KEM, AEAD and KDF algorithms.

This is implemented by the *ossl_HPKE_KEM_INFO_find_random*, *ossl_HPKE_AEAD_INFO_find_random* and *ossl_HPKE_KDF_INFO_find_random* utility functions as shown below. All of these generate a random 1-byte value and use the modulo operation to reduce the value range to the total number of valid identifiers. Using modulo this way may introduce a slight bias, if the randomly generated number is not divisible by the total number of value identifiers[10]. Please note such a bias will be noticeable when an attacker is capable of collecting a large amount of output samples.

Since GREASE values for e.g. TLS will usually be sent with every handshake, this seems practical. The importance of adequate randomly-generated GREASE values is also explicitly mentioned in the security considerations section of RFC8701[11].

The HPKE implementation currently contains this modulo bias in a corner case, when it selects a random AEAD function, and ChaCha20 or Poly1305 are not available. This may also occur when more HPKE identifiers are specified and implemented in the future, as the number of total entries increases.

The following snippet shows the random selection code for KEM, AEAD and KDF:

Affected File:
https://github.com/openssl/openssl/blob/50b3[…]/crypto/hpke/hpke_util.c#L190

[8] https://github.com/openssl/openssl/pull/22493
[9] https://datatracker.ietf.org/doc/html/rfc8701
[10] https://research.kudelskisecurity.com/2020/07/28/the-definitive-guide-to-modulo-bias-...
[11] https://datatracker.ietf.org/doc/html/rfc8701#section-7

Affected Code:

```
const OSSL_HPKE_KEM_INFO *ossl_HPKE_KEM_INFO_find_random(OSSL_LIB_CTX *ctx)
{
    unsigned char rval = 0;
    int sz = OSSL_NELEM(hpke_kem_tab);

    if (RAND_bytes_ex(ctx, &rval, sizeof(rval), 0) <= 0)
        return NULL;
    return &hpke_kem_tab[rval % sz];
}
[...]
const OSSL_HPKE_KDF_INFO *ossl_HPKE_KDF_INFO_find_random(OSSL_LIB_CTX *ctx)
{
    unsigned char rval = 0;
    int sz = OSSL_NELEM(hpke_kdf_tab);

    if (RAND_bytes_ex(ctx, &rval, sizeof(rval), 0) <= 0)
        return NULL;
    return &hpke_kdf_tab[rval % sz];
}

[...]
const OSSL_HPKE_AEAD_INFO *ossl_HPKE_AEAD_INFO_find_random(OSSL_LIB_CTX *ctx)
{
    unsigned char rval = 0;
    /* the minus 1 below is so we don't pick the EXPORTONLY codepoint */
    int sz = OSSL_NELEM(hpke_aead_tab) - 1;

    if (RAND_bytes_ex(ctx, &rval, sizeof(rval), 0) <= 0)
        return NULL;
    return &hpke_aead_tab[rval % sz];
}
```

It is suggested to use rejection sampling or a similar strategy to remove this modulo bias such that all HPKE suite identifiers are chosen with the same probability and fingerprinting is not possible.

DEF-01-006 WP1: HPKE API Fails To Reject Weak PSK Values *(Low)*

Retest Notes: The DEfO-2 team fixed[12] this issue and 7ASecurity confirmed that the fix is valid.

During the code audit and implementation review of the OpenSSL user API provided for HPKE, it was discovered that it allows setting weak, too short values for a Pre-Shared Key (PSK). This opens the door for API users to choose easy-to-guess PSK values, and hence, might lead to HPKE sender impersonation in a worst-case scenario.

HPKE supports two modes with a PSK which enable sender authentication. As specified in RFC9180, the PSK must have at least 32 bytes of entropy and should be of length *Nh* (length of KDF output)[13]. The OpenSSL API for setting the PSK is *OSSL_HPKE_CTX_set1_psk*, which does validate the maximum length of a PSK, but fails to check if the PSK has a minimum length above zero. Therefore the API allows users to set short and weak PSK values which must not be allowed per specification.

This issue can be confirmed observing the following code:

Affected File:
https://github.com/openssl/openssl/blob/50b3[…]/crypto/hpke/hpke.c#L875

Affected Code:

```
int OSSL_HPKE_CTX_set1_psk(OSSL_HPKE_CTX *ctx,
                           const char *pskid,
                           const unsigned char *psk, size_t psklen)
{
    if (ctx == NULL || pskid == NULL || psk == NULL || psklen == 0) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_NULL_PARAMETER);
        return 0;
    }
    if (psklen > OSSL_HPKE_MAX_PARMLEN) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_INVALID_ARGUMENT);
        return 0;
    }
    if (strlen(pskid) > OSSL_HPKE_MAX_PARMLEN) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_INVALID_ARGUMENT);
        return 0;
    }
    if (ctx->mode != OSSL_HPKE_MODE_PSK
        && ctx->mode != OSSL_HPKE_MODE_PSKAUTH) {
```

[12] https://github.com/openssl/openssl/pull/22493
[13] https://datatracker.ietf.org/doc/html/rfc9180#section-9.5

```
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_INVALID_ARGUMENT);
        return 0;
    }
    /* free previous values if any */
    OPENSSL_clear_free(ctx->psk, ctx->psklen);
    ctx->psk = OPENSSL_memdup(psk, psklen);
    if (ctx->psk == NULL)
        return 0;
    ctx->psklen = psklen;
    OPENSSL_free(ctx->pskid);
    ctx->pskid = OPENSSL_strdup(pskid);
    if (ctx->pskid == NULL) {
        OPENSSL_clear_free(ctx->psk, ctx->psklen);
        ctx->psk = NULL;
        ctx->psklen = 0;
        return 0;
    }
    return 1;
}
```

It is recommended to add a check for the minimum PSK length to prevent API misuse and the resulting implications of potential sender impersonation by malicious actors.

DEF-01-007 WP1: Unnecessary LabeledExtract in HPKE KeySchedule *(Info)*

Retest Notes: The DEfO-2 team fixed[14] this issue and 7ASecurity confirmed that the fix is valid.

While reviewing the HPKE key schedule code, it was noticed that an additional, *LabeledExtract*[15] operation is performed on the PSK. However, this serves no purpose in the algorithm. It appears that this is leftover code from an earlier draft of the RFC. While this has no security impact on its own, it unnecessarily increases complexity, and hence, might indirectly increase the odds of potential security weaknesses in the future. This can be confirmed reviewing the following section of the implementation:

Affected File:
https://github.com/openssl/openssl/blob/50b3[…]/crypto/hpke/hpke.c#L729

Affected Code:

```
static int hpke_do_middle(OSSL_HPKE_CTX *ctx,
                          const unsigned char *info, size_t infolen)
{
```

[14] https://github.com/openssl/openssl/pull/22493
[15] https://datatracker.ietf.org/doc/html/rfc9180#section-4

```
[...]
if (ossl_hpke_labeled_extract(kctx, ks_context + 1 + halflen, halflen,
                              NULL, 0, OSSL_HPKE_SEC51LABEL,
                              suitebuf, sizeof(suitebuf),
                              OSSL_HPKE_INFOHASH_LABEL,
                              (unsigned char *)info, infolen) != 1) {
    ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
    goto err;
}
ks_contextlen = 1 + 2 * halflen;
/* Extract and Expand variously... */
psk_hashlen = halflen;
if (ossl_hpke_labeled_extract(kctx, psk_hash, psk_hashlen,
                              NULL, 0, OSSL_HPKE_SEC51LABEL,
                              suitebuf, sizeof(suitebuf),
                              OSSL_HPKE_PSK_HASH_LABEL,
                              ctx->psk, ctx->psklen) != 1) {
    ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
    goto err;
}
secretlen = kdf_info->Nh;
if (secretlen > OSSL_HPKE_MAXSIZE) {
    ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
    goto err;
}
if (ossl_hpke_labeled_extract(kctx, secret, secretlen,
                              ctx->shared_secret, ctx->shared_secretlen,
                              OSSL_HPKE_SEC51LABEL,
                              suitebuf, sizeof(suitebuf),
                              OSSL_HPKE_SECRET_LABEL,
                              ctx->psk, ctx->psklen) != 1) {
    ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
    goto err;
}
```

It is suggested to remove this superfluous code to improve readability and maintainability.

DEF-01-008 WP1: HPKE API Allows Invalid *psk_id* & *psk* Combinations *(Low)*

Retest Notes: The DEfO-2 team fixed[16] this issue and 7ASecurity confirmed that the fix is valid.

It was uncovered that the HPKE API fails to reject *psk_id* and *psk* value combinations explicitly prohibited by RFC9180. Specifically, *psk_id* can be set to its default value (empty string), while *psk* can be set to a non-empty string. This makes the API prone to misuse, which might lead to potentially insecure configurations outside of the HPKE specification. The root cause for this issue can be found in the code path below, which shows the *OSSL_HPKE_CTX_set1_psk* function is used to set the *psk* and *psk_id* values for a HPKE context type.

The identified behavior violates section *5.1. Creating the Encryption Context* of RFC9180, which states[17]:

> *"The psk and psk_id fields MUST appear together or not at all. That is, if a non-default value is provided for one of them, then the other MUST be set to a non-default value."*

However, as evident from the code below, there are only checks for *psk_id* and *psk* being NULL and *psklen == 0*, but no check for *strlen(psk_id) == 0*. Thus, a caller may accidentally supply a pointer to a valid buffer containing an empty string.

Affected File:
https://github.com/openssl/openssl/blob/50b3[…]/crypto/hpke/hpke.c#L875

Affected Code:

```
int OSSL_HPKE_CTX_set1_psk(OSSL_HPKE_CTX *ctx,
                           const char *pskid,
                           const unsigned char *psk, size_t psklen)
{
    if (ctx == NULL || pskid == NULL || psk == NULL || psklen == 0) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_NULL_PARAMETER);
        return 0;
    }
    if (psklen > OSSL_HPKE_MAX_PARMLEN) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_INVALID_ARGUMENT);
        return 0;
    }
    if (strlen(pskid) > OSSL_HPKE_MAX_PARMLEN) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_INVALID_ARGUMENT);
```

[16] https://github.com/openssl/openssl/pull/22493
[17] https://datatracker.ietf.org/doc/html/rfc9180#section-5.1

```
        return 0;
    }
    if (ctx->mode != OSSL_HPKE_MODE_PSK
        && ctx->mode != OSSL_HPKE_MODE_PSKAUTH) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_INVALID_ARGUMENT);
        return 0;
    }
    /* free previous values if any */
    OPENSSL_clear_free(ctx->psk, ctx->psklen);
    ctx->psk = OPENSSL_memdup(psk, psklen);
    if (ctx->psk == NULL)
        return 0;
    ctx->psklen = psklen;
    OPENSSL_free(ctx->pskid);
    ctx->pskid = OPENSSL_strdup(pskid);
    if (ctx->pskid == NULL) {
        OPENSSL_clear_free(ctx->psk, ctx->psklen);
        ctx->psk = NULL;
        ctx->psklen = 0;
        return 0;
    }
    return 1;
}
```

It is recommended to implement the specified checks on *psk* and *psk_id* from RFC9180 to eliminate the potential for API misuse.

DEF-01-009 WP1: Possible NULL Dereference in *OSSL_HPKE_export* *(Info)*

Retest Notes: The DEfO-2 team fixed[18] this issue and 7ASecurity confirmed that the fix is valid.

Close inspection of the HPKE export API showed that there is a possibility for a NULL dereference. This may occur when providing *NULL* for the *label* argument, while *labellen* is set to a positive value. However, as this can happen only when the API is used incorrectly, this is simply mentioned as a hardening recommendation similar to DEF-01-003.

The root cause for this issue can be observed in the following code path, where *OSSL_HPKE_export* checks a number of arguments, but fails to verify whether *label* is NULL. Instead, the pointer is relayed to *ossl_hpke_labeled_expand* which calls *WPACKET_memcpy* without further verification:

[18] https://github.com/openssl/openssl/pull/22493

Affected File:
https://github.com/openssl/openssl/blob/50b3[…]/crypto/hpke/hpke.c#L1215

Affected Code:

```
int OSSL_HPKE_export(OSSL_HPKE_CTX *ctx,
                     unsigned char *secret, size_t secretlen,
                     const unsigned char *label, size_t labellen)
{
    int erv = 0;
    EVP_KDF_CTX *kctx = NULL;
    unsigned char suitebuf[6];
    const char *mdname = NULL;
    const OSSL_HPKE_KDF_INFO *kdf_info = NULL;

    if (ctx == NULL) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_NULL_PARAMETER);
        return 0;
    }
    if (labellen > OSSL_HPKE_MAX_PARMLEN) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_INVALID_ARGUMENT);
        return 0;
    }
    [...]
    /* full suiteid as per RFC9180 sec 5.3 */
    suitebuf[0] = ctx->suite.kem_id / 256;
    suitebuf[1] = ctx->suite.kem_id % 256;
    suitebuf[2] = ctx->suite.kdf_id / 256;
    suitebuf[3] = ctx->suite.kdf_id % 256;
    suitebuf[4] = ctx->suite.aead_id / 256;
    suitebuf[5] = ctx->suite.aead_id % 256;
    erv = ossl_hpke_labeled_expand(kctx, secret, secretlen,
                                   ctx->exportersec, ctx->exporterseclen,
                                   OSSL_HPKE_SEC51LABEL,
                                   suitebuf, sizeof(suitebuf),
                                   OSSL_HPKE_EXP_SEC_LABEL,
                                   label, labellen);
    EVP_KDF_CTX_free(kctx);
    if (erv != 1)
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
    return erv;
}
```

Affected File:
https://github.com/openssl/openssl/blob/50b3[…]/crypto/hpke/hpke_util.c#L343

Affected Code:

```
int ossl_hpke_labeled_expand(EVP_KDF_CTX *kctx,
```

```
                            unsigned char *okm, size_t okmlen,
                            const unsigned char *prk, size_t prklen,
                            const char *protocol_label,
                            const unsigned char *suiteid, size_t suiteidlen,
                            const char *label,
                            const unsigned char *info, size_t infolen)
{
    int ret = 0;
    size_t label_hpkev1len = 0;
    size_t protocol_labellen = 0;
    size_t labellen = 0;
    size_t labeled_infolen = 0;
    unsigned char *labeled_info = NULL;
    WPACKET pkt;

    label_hpkev1len = strlen(LABEL_HPKEV1);
    protocol_labellen = strlen(protocol_label);
    labellen = strlen(label);
    labeled_infolen = 2 + okmlen + prklen + label_hpkev1len
        + protocol_labellen + suiteidlen + labellen + infolen;
    labeled_info = OPENSSL_malloc(labeled_infolen);
    if (labeled_info == NULL)
        return 0;

    /* labeled_info = concat(okmlen, "HPKE-v1", suiteid, label, info) */
    if (!WPACKET_init_static_len(&pkt, labeled_info, labeled_infolen, 0)
            || !WPACKET_put_bytes_u16(&pkt, okmlen)
            || !WPACKET_memcpy(&pkt, LABEL_HPKEV1, label_hpkev1len)
            || !WPACKET_memcpy(&pkt, protocol_label, protocol_labellen)
            || !WPACKET_memcpy(&pkt, suiteid, suiteidlen)
            || !WPACKET_memcpy(&pkt, label, labellen)
            || !WPACKET_memcpy(&pkt, info, infolen)
            || !WPACKET_get_total_written(&pkt, &labeled_infolen)
            || !WPACKET_finish(&pkt)) {
        ERR_raise(ERR_LIB_PROV, PROV_R_OUTPUT_BUFFER_TOO_SMALL);
        goto end;
    }

    ret = ossl_hpke_kdf_expand(kctx, okm, okmlen,
                               prk, prklen, labeled_info, labeled_infolen);
end:
    WPACKET_cleanup(&pkt);
    OPENSSL_free(labeled_info);
    return ret;
}
```

Please note that, while it appears the same issue applies for the *secret* parameter, this is not the case, as that is mitigated deeper down the call-chain, by the KDF code itself.

It is suggested to check the export function inputs for being *NULL* and return an error in such a case.

DEF-01-010 WP1: Hardcoded Buffer Size in *hpke_aead_enc* *(Info)*

During code audit of *hpke_aead_enc*, it was discovered that the buffer size for the authentication tag is hardcoded to 16 bytes. This is acceptable for the currently specified AEAD ciphers, as they all have a tag size of 16 bytes. However, as the code uses the exposed tag size of AEAD ciphers as the buffer size, this might become a potential weakness in the future, when ciphers with different tag lengths are added. For example, future additions of AEAD ciphers with a larger tag length might introduce a buffer overflow vulnerability.

The root cause for this issue can be found in the following code path, which displays the hardcoded buffer size for *tag* in *hpke_aead_enc* and that *hctx->aead_info->taglen* is used as size of the buffer. This is problematic, since there is no assertion for *hctx->aead_info->taglen* always being 16:

Affected File:
https://github.com/openssl/openssl/blob/50b3[…]/crypto/hpke/hpke.c#L220

Affected Code:

```
static int hpke_aead_enc(OSSL_HPKE_CTX *hctx, const unsigned char *iv,
                         const unsigned char *aad, size_t aadlen,
                         const unsigned char *pt, size_t ptlen,
                         unsigned char *ct, size_t *ctlen)
{
    int erv = 0;
    EVP_CIPHER_CTX *ctx = NULL;
    int len;
    size_t taglen = 0;
    unsigned char tag[16];

    taglen = hctx->aead_info->taglen;
    if (*ctlen <= taglen || ptlen > *ctlen - taglen) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_PASSED_INVALID_ARGUMENT);
        return 0;
    }
    [...]
    *ctlen += len;
    /* Get tag. Not a duplicate so needs to be added to the ciphertext */
    if (EVP_CIPHER_CTX_ctrl(ctx, EVP_CTRL_AEAD_GET_TAG, taglen, tag) != 1) {
        ERR_raise(ERR_LIB_CRYPTO, ERR_R_INTERNAL_ERROR);
```

```
        goto err;
    }
    memcpy(ct + *ctlen, tag, taglen);
    *ctlen += taglen;
    erv = 1;

err:
    if (erv != 1)
        OPENSSL_cleanse(ct, *ctlen);
    EVP_CIPHER_CTX_free(ctx);
    return erv;
}
```

It is suggested to use a buffer size matching the selected AEAD cipher. While this will currently always be 16 bytes, modifications to future AEAD ciphers will not lead to any unforeseen buffer overflows.

Conclusion

The DEfO-2 OpenSSL PR modifications provided a number of positive impressions during this assignment that must be mentioned here:

- 7ASecurity was unable to identify any directly exploitable vulnerability on the DEfO-2 implementation. This unusual result is particularly remarkable for a first security audit.
- Similarly, the code audit of the HPKE codebase failed to spot any significant issue.
- Other noteworthy code review impressions from the test team include the clarity and simplicity of the implementation. Which not only made the investigation process straightforward, but also made evident that the development team is well-aware of best-practices in terms of secure programming.
- Furthermore, the code implements RFC9180 very concisely, which makes it robust and avoids undefined protocol behavior, while reducing the attack surface substantially.
- The coding style could perhaps be best defined as elegant, clean and defensive, which fits a cryptographic library like OpenSSL.

The security of the DEfO-2 enhancements may be improved further with a focus on the following areas:

- **Input Validation:** In order to enhance security, developers should implement rigorous input validation checks to ensure that user-supplied data meets the expected criteria, thereby preventing malicious input from compromising the system. Additionally, meticulously validating return values from functions and methods is crucial, as it ensures the appropriate execution of operations and allows developers to handle errors effectively.
- **Fuzzing Test Cases:** In terms of dynamic testing, several fuzz testing harnesses were developed by the audit team during this assignment. While testing is essential for any project, its importance grows with the scale of the endeavor. Testing is, therefore, essential for the overall security of the OpenSSL project, especially since it is implemented in C, which is a language often prone to memory corruption vulnerabilities. Therefore, integrating fuzzing harnesses for the HPKE feature into the codebase would be beneficial.
- **Automated CI/CD Tests:** More unit tests could to be deployed to ensure similar weaknesses are not re-introduced in the future. This could be accomplished by integrating automated tests in the OpenSSL and/or DEfO-2 CI/CD pipelines. Some examples to consider in this regard would be the fuzzers, *CodeQL* and *semgrep* rules.

It is advised to address all issues identified in this report, including informational and low severity tickets where possible. This will not just strengthen the security posture of the implementation significantly, but also reduce the number of tickets in future audits.

Once all issues in this report are addressed and verified, a more thorough review, ideally including another code audit, is highly recommended to ensure adequate security coverage of the platform. This will be particularly crucial following the incorporation of the *Encrypted Client Hello* (ECH) as an application of HPKE into the codebase.

It is suggested to test the implementation regularly, at least when substantial changes are going to be deployed, to make sure new features do not introduce undesired security vulnerabilities. This proven strategy will reduce the number of security issues consistently and make OpenSSL highly resilient against online attacks over time.

7ASecurity would like to take this opportunity to sincerely thank Stephen Farrell and the rest of the DEfO-2 team, for their exemplary assistance and support throughout this audit. Last but not least, appreciation must be extended to the *Open Technology Fund (OTF)* for sponsoring this project.