Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Audit-Report ExpressVPN Lightway Protocol 03.2021

Cure53, Dr.-Ing. M. Heiderich, Dipl.-Inf. Markus Vervier, Dipl.-Inf. Eric Sesterhenn, Dipl.-Inf. Luis Merino, Dipl.-Inf. Djamal Touazi

Index

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Introduction

"You don't need to know how a VPN works to use a VPN. But if you've ever checked the settings on your ExpressVPN app, you'll see a tab that lets you choose a protocol. Protocols are methods by which your device connects to ExpressVPN's secure servers. Find out how protocols differ and how to choose the best protocol for you."

From https://www.expressvpn.com/what-is-vpn/protocols

This report presents the findings of a security assessment featuring the ExpressVPN Lightway protocol and its related sources. Carried out by Cure53 in March 2021, the project entailed a penetration test and a source code audit. Note that the Lightway protocol is referred to by its internal project codename, Helium, within the source code.

The work was requested by ExpressVPN International Ltd. in early February 2021 and then promptly scheduled. Based on the required skills and expertise, a team of five senior testers were assigned to this project's preparation, execution and finalization. They executed the examination in the second half of March 2021, namely in CW11 and CW12. Further commenting on the resources, a total of twenty-two person-days were invested into reaching the coverage expected for this project.

In order to best address the goals set for this assignment, the work was divided into two separate work packages (WPs). In WP1, Cure53 performed source-code assisted penetration tests against primary scope items delineated as *xv_helium_cli*, *xv_helium_server* and *xv_libhelium* (note that *libhelium* is known publicly as Lightway Core). Those were examined in full. For WP2, source-code assisted penetration tests also took place, albeit focused on secondary targets. The latter scope, with items audited in parts only relevant for the WP1 items, encompassed *libdnet* and *libuv.* The project was completed with white-box methodology. Cure53 was given access to all relevant sources as well as various binaries compiled for this security assessment.

Before the assessment, all preparations were finalized in early March 2021, namely in CW10, so as to ensure that Cure53 can have a smooth start. The scope was excellently prepared by the ExpressVPN team and all necessary info was present prior to the start of the project. No road bumps were encountered during the testing and auditing processes. A dedicated and shared Slack channel was leveraged for communications, effectively gluing ExpressVPN and Cure53 workspaces together. All discussions were helpful, although not many questions were needed because of the stellar preparatory work.

Still, Cure53 offered frequent status updates about the test and the spotted findings. Live-reporting was requested and performed by Cure53 for one finding, namely the

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Denial-of-Service (DoS) believed to *potentially* signify an RCE as well (see EXP-04-014). More generally, the testing team acquired a very good coverage over the WP1-WP2 scope items. Fourteen security-relevant discoveries were made: five were classified as security vulnerabilities and nine represented general weaknesses with lower exploitation potential. Note that three findings have given *High* severity ratings given their potential impact on the ExpressVPN users. Among these, two constituted DoS problems and the last one concerned a known WolfSSL vulnerability, namely *CVE-2021-3336*. While one of the findings was borderline *Critical*, the available testing time prevented Cure53 from developing a working exploit.

In the following sections, the report will first shed light on the scope and key test parameters, as well as the structure and content of the WPs. Next, all findings will be discussed in grouped vulnerability and miscellaneous categories, then following a chronological order in each of the arrays. Alongside technical descriptions, PoC and mitigation advice are supplied when applicable. Finally, the report will close with broader conclusions about this March 2021 project. Cure53 elaborates on the general impressions and reiterates the verdict based on the testing team's observations and collected evidence. Tailored hardening recommendations for the ExpressVPN complex are also incorporated into the final section.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Scope

- **Code audits & assessments of ExpressVPN Lightway protocol & sources**
  - **WP1**: Source-code assisted penetration tests of primary scope items
    - The following items were in primary scope for this exercise and got audited fully:
      - *xv_helium_cli*
      - *xv_helium_server*
      - *xv_libhelium* (Lightway Core)
  - **WP2**: Source-code assisted penetration tests against secondary scope items
    - The following items were in secondary scope; parts to be audited are those relevant to and utilized by items in WP1:
      - *libdnet*
      - *libuv*
  - **The following items were out-of-scope:**
    - WolfSSL Library and Reference implementation
    - ExpressVPN Client Applications & ExpressVPN setup files provided
    - Information leaks in Debug builds
    - Denial-of-Service in the Lightway client caused by OOM bugs
  - **A detailed scope document was shared by ExpressVPN**
  - **Binaries for testing were shared with Cure53**
  - **All relevant sources were shared with Cure53**

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *EXP-04-001*) for the purpose of facilitating any future follow-up correspondence.

EXP-04-007 Server: User-enumeration due to login timing differences *(Info)*

Note: *This issue was discussed with the ExpressVPN team and it was shown that existing mechanisms on username generation are sufficiently secure - enumeration is unlikely given the fact that usernames are created from random bytes with sufficient entropy. The issue was therefore changed from medium severity to informational severity.*

User-authentication retrieves the password hash of the user from the database and compares it with the hashed password the user supplied. If a username does not exist in the database, no hashing is performed. The authentication returns faster in comparison to the case where the username exists but the supplied password is invalid. This timing difference could be abused by an attacker to verify whether a username is valid or not. Modern password hashing algorithms are designed to require some time to compute, which makes this attack easier in case they are used.

Affected file:
xv_helium_server/lua/he_auth.lua

Affected code:

```
function auth_user(username, password)
  load_db()
  stmt:bind_values(username)

  retval = false

  for row in stmt:nrows() do
                retval = crypt.check(password, row.encrypted_credentials)
        end

  unload_db()

        -- No user found?
        return retval
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
end
```

It is recommended to use a secure key derivation function such as *ARGON2* on the supplied passwords across all cases. This will mitigate the possibility of successful timing attacks.

EXP-04-010 Server: Amplification DoS due to packet-resending *(Medium)*

***Note**: This issue was discussed with the ExpressVPN team and it was found that an existing firewall rule was deployed in production systems. The rule was reviewed by Cure53 and deemed to be sufficient to mitigate the problem.*

When the Helium server receives a client connection containing the DTLS Client Hello, it will respond to that client with an answer. In case there is no immediate reply, two more answers are sent to that initial packet. The packet coming from the client is 144 bytes, making the answers 228 bytes in total. Since the protocol used is UDP, an attacker is able to send packets with a spoofed IP address and request a connection for a third-party. The server will then send more bytes to that third-party than the attacker sent (~1.5 times the amount). This lets attackers abuse the Helium server for UDP-based DoS attacks and increase the bandwidth available to them by ~50%.

Affected file:
xv_helium_cli/lib/libhelium/src/he/wolf.c

Affected code:

```
// Call the write callback if set
if(client->outside_write_cb) {
  client->outside_write_cb(client, client->write_buffer, sz +
sizeof(he_wire_hdr_t),
                           client->data);

  // If we're not yet connected, be aggressive and send two more packets. If
aggressive mode
  // is set, always be aggressive and send two more.
  if(client->state != HE_STATE_ONLINE || client->use_aggressive_mode) {
    client->outside_write_cb(client, client->write_buffer, sz +
sizeof(he_wire_hdr_t), client->data);

  client->outside_write_cb(client, client->write_buffer, sz +
sizeof(he_wire_hdr_t), client->data);
  }
}
```

It is proposed to add rate-limiting for packets accepted per IP address to limit the impact of the packet amplification. This could be implemented in code or via a firewall. Another

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

option would be to send only one additional packet when the aggressive mode is enabled.

EXP-04-011 Server: DoS via half-open connections *(Medium)*

***Note**: This issue was discussed with the ExpressVPN team and it was found that an existing firewall rule was deployed in production systems. The rule was reviewed by Cure53 and deemed to be sufficient to mitigate the problem.*

When a new connection is created, the IP address and UDP port (in case of UDP connections) are added to a hashmap which is then used to track the connection for that client. If the client sends no further packets, the connection is removed and freed via a timer that calls *he_conn_nudge()*. If a client sends enough packets with spoofed IP addresses, it is possible to cause this hashmap to grow and cause memory pressure on the system.

If a low number of packets sent by the attacker has a session ID set, the server will perform lookups in the hashmap for that session. The latter causes CPU pressure as well. While it is not going to prevent the server from working, it might increase the time required to process other packets.

Affected file:
xv_helium_server/src/he_flow_in.c

Affected code:
```
// If we still haven't found the connection but also have not rejected it then
create a fresh
// connection
if(!conn) {
  // Creating a new connection can fail. Check that it is successful before
continuing
  if((conn = he_create_new_connection(server, addr, ipcombo)) == NULL) {
    return;
  }
}
```

It is advised to protect against this by rate-limiting the incoming packets. This could be implemented in the code or via a firewall on the network level. This would reduce the memory and CPU pressure an attacker could cause on the system. Another option would be to run *he_conn_nudge()* more frequently and use a shorter timeout to avoid too many connections in the hashmap.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

EXP-04-012 Server: DoS via spoofed packets bouncing between servers *(High)*

Note: *While this issue was technically deemed to be out of scope for the library audit, the issue was nevertheless be addressed by the ExpressVPN team and the fix was verified by Cure53 in June 2021.*

The Helium server receives a packet with a valid Helium header, however, when they are missing payload and a session ID of *HE_PACKET_SESSION_REJECT* (-1), the server responds with exactly the same packet. In case an attacker spoofs this packet with a sender IP that is the same as the IP of the server whilst the UDP port matches the server as well, the server will send answers to these packets to itself. This causes an infinite loop, requesting all available CPU resources.

For most systems, the kernel will drop such spoofed packets. The DoS is still feasible by spoofing such packets so they seem to be from another Helium server, causing the packets to be bouncing between the two servers infinitely.

Affected file:
xv_helium_server/src/he_flow_in.c

PoC:
```
# hexdump -C data

00000000  48 65 01 00 00 00 00 00  ff ff ff ff ff ff ff ff  |He..............|
00000010

# /usr/sbin/hping3 -V -2 -a 10.0.2.15 -s 19655 -p 19655 -c 1 -E data -d 16
10.0.2.15
```

Affected code:
```
he_return_code_t he_internal_flow_outside_packet_received(he_conn_t *conn,
uint8_t *packet,
                                                          size_t length) {
  // Note that he_internal_plugins_egress is in wolf.c:he_wolf_dtls_write
  he_internal_plugins_ingress(conn->plugins, packet, length);

  // Return if packet is definitely too small
  if(length < sizeof(he_wire_hdr_t)) {
    return HE_ERR_PACKET_TOO_SMALL;
  }

  // Check for Helium's header
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
  he_wire_hdr_t *hdr = (he_wire_hdr_t *)packet;

  if(hdr->he[0] != 'H' || hdr->he[1] != 'e') {
    // Not helium data, drop it
    return HE_ERR_NOT_HE_PACKET;
  }

  // Kill the connection if the server has rejected our session (i.e. server
restarted)
  if(!memcmp(&HE_PACKET_SESSION_REJECT, &hdr->session, sizeof(uint64_t))) {
    return HE_ERR_REJECTED_SESSION;
  }
```

It is advised to not send answers to packets with a session of *HE_PACKET_SESSION_REJECT*.

EXP-04-013 Server: DoS via improper closing of TCP connection *(High)*

Note: *This issue was fixed by the ExpressVPN team and the fix was then verified successfully by Cure53 in June 2021. A diff was inspected to verify the fix as working as expected.*

When the server is run with the "*Streaming Mode*" setting on *true* and a client tries to connect but abruptly ceases the connection, the server processes seem to fail. They handle the issue incorrectly as subsequent connections will trigger a heap-use-after-free, hence leading to temporal memory unsafety issues. The screenshot shown below from ASAN demonstrates the issue.

```
=================================================================
==26==ERROR: AddressSanitizer: heap-use-after-free on address 0x614000000b68 at
pc 0x00000053b06f bp 0x7ffcfc30eb00 sp 0x7ffcfc30eaf8
WRITE of size 8 at 0x614000000b68 thread T0
       #0 0x53b06e in uv__stream_init src/unix/stream.c:90
       #1 0x5441d8 in uv_tcp_init_ex src/unix/tcp.c:125
       #2 0x544335 in uv_tcp_init src/unix/tcp.c:144
       #3 0x434ddb in on_new_streaming_connection
(/xv_helium_server/build/release/helium-server.out+0x434ddb)
       #4 0x53c4c7 in uv__server_io src/unix/stream.c:570
       #5 0x551d28 in uv__io_poll src/unix/linux-core.c:462
       #6 0x513ab9 in uv_run src/unix/core.c:385
       #7 0x429f86 in main (/xv_helium_server/build/release/helium-
server.out+0x429f86)
       #8 0x7f58a97ee09a in __libc_start_main ../csu/libc-start.c:308
       #9 0x41b439 in _start (/xv_helium_server/build/release/helium-
server.out+0x41b439)
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
0x614000000b68 is located 296 bytes inside of 424-byte region
[0x614000000a40,0x614000000be8)
freed by thread T0 here:
       #0 0x7f58a9c1cfb0 in __interceptor_free
../../../../src/libsanitizer/asan/asan_malloc_linux.cc:66
       #1 0x426438 in he_internal_free_connection
(/xv_helium_server/build/release/helium-server.out+0x426438)
       #2 0x434a6c in on_tcp_close (/xv_helium_server/build/release/helium-
server.out+0x434a6c)
       #3 0x513636 in uv__finish_close src/unix/core.c:303
       #4 0x5136c8 in uv__run_closing_handles src/unix/core.c:317
       #5 0x513add in uv_run src/unix/core.c:395
       #6 0x429f86 in main (/xv_helium_server/build/release/helium-
server.out+0x429f86)
       #7 0x7f58a97ee09a in __libc_start_main ../csu/libc-start.c:308


previously allocated by thread T0 here:
       #0 0x7f58a9c1d518 in __interceptor_calloc
../../../../src/libsanitizer/asan/asan_malloc_linux.cc:95
       #1 0x42d8df in internal_create_connection
(/xv_helium_server/build/release/helium-server.out+0x42d8df)
       #2 0x42e3c5 in he_create_new_connection_streaming
(/xv_helium_server/build/release/helium-server.out+0x42e3c5)
       #3 0x434e02 in on_new_streaming_connection
(/xv_helium_server/build/release/helium-server.out+0x434e02)
       #4 0x53c4c7 in uv__server_io src/unix/stream.c:570
       #5 0x551d28 in uv__io_poll src/unix/linux-core.c:462
       #6 0x513ab9 in uv_run src/unix/core.c:385
       #7 0x429f86 in main (/xv_helium_server/build/release/helium-
server.out+0x429f86)
       #8 0x7f58a97ee09a in __libc_start_main ../csu/libc-start.c:308

SUMMARY: AddressSanitizer: heap-use-after-free src/unix/stream.c:90 in
uv__stream_init
Shadow bytes around the buggy address:
  0x0c287fff8110: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c287fff8120: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c287fff8130: fd fd fd fd fd fd fd fd fd fd fd fd fa fa fa fa
  0x0c287fff8140: fa fa fa fa fa fa fa fa fd fd fd fd fd fd fd fd
  0x0c287fff8150: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
=>0x0c287fff8160: fd fd fd fd fd fd fd fd fd fd fd fd fd[fd]fd fd
  0x0c287fff8170: fd fd fd fd fd fd fd fd fd fd fd fd fd fa fa fa
  0x0c287fff8180: fa fa fa fa fa fa fa fa fd fd fd fd fd fd fd fd
  0x0c287fff8190: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c287fff81a0: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c287fff81b0: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
==26==ABORTING
```

This situation can, for example, occur when the client is unable to create a TUN device due to incorrect privilege. This leads to tearing down the connection quickly after initiating it.

Affected file:
xv_helium_server/src/uv_callbacks.c

Affected code:

UV_tcp_init call inside the *on new_streaming_connection* callback is the call that triggers the heap's use after free condition in the server code.

```
void on_new_streaming_connection(uv_stream_t *server, int status) {
  he_server_t *he_server = (he_server_t *)server->data;

  if(status < 0) {
       zlogf_time(ZLOG_INFO_LOG_MSG, "New connection error %s\n",
uv_strerror(status));
       zlog_flush_buffer();
       return;
  }

  uv_tcp_t *client = (uv_tcp_t *)calloc(1, sizeof(uv_tcp_t));
  HE_CHECK_WITH_MSG(client != NULL, "Unable to allocate new tcp_client\n");
  uv_tcp_init(he_server->loop, client);
  he_server_connection_t *conn;
  if(uv_accept(server, (uv_stream_t *)client) == 0 &&
       (conn = he_create_new_connection_streaming(he_server)) != NULL) {
       client->data = conn;
       conn->tcp_client = (uv_stream_t *)client;
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
        uv_read_start((uv_stream_t *)client, alloc_buffer_tcp, on_tcp_read);
    } else {
        zlogf_time(ZLOG_INFO_LOG_MSG, "Unable to accept %s\n",
uv_strerror(status));
        uv_close((uv_handle_t *)client, NULL);
        free(client);
    }
}
```

Steps to reproduce:

1. On the server configuration file, issue the following setting:
   `"streaming = true"`
2. On the client-side, set the connection protocol to TCP:
   `"protocol": "tcp"`
3. Run the client in an unprivileged user that cannot create a TUN device. This will lead the client to abruptly close the connection:

```
{"time":"[...]","log_level":"INFO","message":"Authenticating...","context
":{"state":"HE_STATE_AUTHENTICATING"}}
{"time":"[...]","log_level":"INFO","message":"Attempting to create tun
device '<automatic>'"}
{"time":"[...]","log_level":"ERROR","message":"failed to TUNSETIFF:
Operation not permitted"}
{"time":"[...]","log_level":"ERROR","message":"Unable to create tun
device"}
{"time":"[...]","log_level":"INFO","message":"Lightway DISCONNECTED
(disconnect_and_stop).","context":
{"state":"HE_STATE_DISCONNECTED","reason":"HE_ERR_CALLBACK_FAILED"}}
{"time":"[...]","log_level":"INFO","message":"Lightway STOPPED"}
{"time":"[...]","log_level":"INFO","message":"Lightway
DISCONNECTING...","context":{"state":"HE_STATE_DISCONNECTING"}}
{"time":"[...]","log_level":"INFO","message":"Lightway FINISHED"}
```

After that rerun, the client will a few times cause the memory corruption. Triggering it will lead to a server crash.

It is recommended to rework the handling of TCP stream connection. The tests show that *libuv* seems not to properly isolate connection context which leads to the reuse of object pointers of a previous connection context. It is also recommended to add ASAN to the current test-suite as well as *valgrind.* This will help catch memory violations more effectively.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

EXP-04-001 libdnet: Buffer overflow in *arp-ioctl.c* via *sscanf()* *(Low)*

Note: *This issue was fixed by the ExpressVPN team and the fix was then verified successfully by Cure53 in June 2021. A diff was inspected to verify the fix as working as expected.*

A malformed *sscanf()* in *arp_loop()* results in several out-of-bounds writes when the *proc* filesystem returns malicious data. These might be abused to corrupt the memory of the process. Since the malicious data needs to be provided by the Kernel-controlled *procfs,* an attack via this vector is not likely.

Affected file:
xv_helium_cli/lib/libdnet/src/arp-ioctl.c

Affected code:

```
char buf[BUFSIZ], ipbuf[100], macbuf[100], maskbuf[100], devbuf[100];
int i, type, flags, ret;

if ((fp = fopen(PROC_ARP_FILE, "r")) == NULL)
        return (-1);

ret = 0;
while (fgets(buf, sizeof(buf), fp) != NULL) {
        i = sscanf(buf, "%s 0x%x 0x%x %100s %100s %100s\n",
            ipbuf, &type, &flags, macbuf, maskbuf, devbuf);
```

A fix for this issue can be found at the following URL:
https://github.com/ofalk/libdnet/commit/661c72866a6522009612accc5403d7527bf9052c

The changes proposed below will mitigate the issue:

```
diff --git a/src/arp-ioctl.c b/src/arp-ioctl.c
index bc3e66c..ca47ce9 100644
--- a/src/arp-ioctl.c
+++ b/src/arp-ioctl.c
@@ -210,7 +210,7 @@ arp_loop(arp_t *a, arp_handler callback, void *arg)
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
 {
        FILE *fp;
        struct arp_entry entry;
-       char buf[BUFSIZ], ipbuf[100], macbuf[100], maskbuf[100], devbuf[100];
+       char buf[BUFSIZ], ipbuf[101], macbuf[101], maskbuf[101], devbuf[101];
        int i, type, flags, ret;

        if ((fp = fopen(PROC_ARP_FILE, "r")) == NULL)
diff --git a/src/route-linux.c b/src/route-linux.c
index bbd95da..45b5c2e 100644
--- a/src/route-linux.c
+++ b/src/route-linux.c
@@ -306,7 +306,7 @@ route_loop(route_t *r, route_handler callback, void *arg)
        int ret = 0;

        if ((fp = fopen(PROC_ROUTE_FILE, "r")) != NULL) {
-               char ifbuf[16];
+               char ifbuf[17];
                int i, iflags, refcnt, use, metric, mss, win, irtt;
                uint32_t mask;
```

EXP-04-002 Unity: Integer overflow in *unity_memory.c*'s *unity_calloc()* (*Low*)

Note: *This issue was deemed to be out of scope for the library audit since Unity is a testing framework which is not called in binaries shipped in production - no action will be taken here.*

The function *unity_calloc()* contains an integer overflow that might lead to less memory being allocated than what is requested by the caller. This can result in memory corruption errors, in turn letting attackers execute arbitrary code. The current code only calls this function with static parameters, thus making an attack infeasible. At the same time, this situation might change in the future.

Affected file:
xv_helium_cli/lib/Unity/extras/memory/src/unity_memory.c

Affected code:

```
void* unity_calloc(size_t num, size_t size)
{
    void* mem = unity_malloc(num * size);
    if (mem == NULL) return NULL;
    memset(mem, 0, num * size);
    return mem;
}
```

It is recommended to test if the calculation overflows, for example by using *__builtin_mul_overflow()*.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

EXP-04-003 Unity: Integer overflow in *unity_memory.c*'s *unity_malloc()* *(Low)*

Note: *This issue was deemed to be out of scope for the library audit since Unity is a testing framework which is not called in binaries shipped in production - no action will be taken here.*

The function *unity_malloc()* contains an integer overflow that might lead to less memory being allocated than what has been requested by the caller. This can result in memory corruption errors, which might, in turn, make it possible for the attackers to execute arbitrary code. The current code only calls this function with static parameters, therefore making exploitation of the issue infeasible. Still, this might change in the future.

Affected file:
xv_helium_cli/lib/Unity/extras/memory/src/unity_memory.c

Affected code:

```
void* unity_malloc(size_t size)
{
    char* mem;
    Guard* guard;
    size_t total_size;

    total_size = sizeof(Guard) + unity_size_round_up(size + sizeof(end));
```

It is recommended to test if the calculation overflows, for example with *__builtin_add_overflow()*.

EXP-04-004 WolfSSL: *CVE-2021-3336* is a known vulnerability in WolfSSL *(High)*

Note: *While this issue was technically deemed to be out of scope for the library audit, the issue has nevertheless been addressed by the ExpressVPN team.*

libHelium relies on WolfSSL for handling secure transport of data via TLS / DTLS. The current version is affected by a security vulnerability indexed with *CVE-2021-3336*[1]. It is related to TLS1.3 certificate validation which is the preferred and default protocol used by *libHelium*. This makes the server directly affected by this *CVE*.

Affected files:

- *xv_helium_cli/lib/libhelium/windows_64.yml*
- *xv_helium_cli/lib/libhelium/windows_32.yml*

[1] https://nvd.nist.gov/vuln/detail/CVE-2021-3336

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

- *xv_helium_cli/lib/libhelium/unix.yml*
- *xv_helium_server/xv_libhelium/windows_64.yml*
- *xv_helium_server/xv_libhelium/windows_32.yml*
- *xv_helium_server/xv_libhelium/unix.yml*

It is recommended to upgrade to the latest version of WolfSSL that has no known vulnerabilities, namely https://github.com/wolfSSL/wolfssl/releases

EXP-04-005 Libuv: Out-of-bounds read in UTF8 parsing *(Low)*

Note: *This issue was deemed to be out of scope for the library audit, the affected method is not being called by anything in scope. No action will be taken here.*

An out-of-bound read can occur when *uv__idna_toascii()* is used to convert strings to ASCII. The pointer *p* is read and increased without checking whether it is beyond *pe*, with the latter holding a pointer to the end of the buffer. This can lead to information disclosures or crashes. This function can be triggered via *uv_getaddrinfo()*.

Affected file:
xv_helium_cli/lib/libuv/src/idna.c

Affected code:

```
static unsigned uv__utf8_decode1_slow(const char** p,
                                      const char* pe,
                                      unsigned a) {
  unsigned b;
  unsigned c;
  unsigned d;
  unsigned min;

  if (a > 0xF7)
    return -1;

  switch (*p - pe) {
  default:
    if (a > 0xEF) {
      min = 0x10000;
      a = a & 7;
      b = (unsigned char) *(*p)++;
      c = (unsigned char) *(*p)++;
      d = (unsigned char) *(*p)++;
      break;
    }
    /* Fall through. */
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

It is recommended to test if the read is beyond *pe*, for example by changing the code to the following:

```
static unsigned uv__utf8_decode1_slow(const char** p,
                                      const char* pe,
                                      unsigned a) {
  unsigned b;
  unsigned c;
  unsigned d;
  unsigned min;

  if (a > 0xF7)
    return -1;

  switch (*p - pe) {
  default:
    if (a > 0xEF) {
      if (p + 3 > pe)
        return -1;
      min = 0x10000;
      a = a & 7;
      b = (unsigned char) *(*p)++;
      c = (unsigned char) *(*p)++;
      d = (unsigned char) *(*p)++;
      break;
    }
    /* Fall through. */
```

The same applies to other cases in the switch statement.

EXP-04-006 Libuv: More memory allocated than required *(Info)*

Note: *This issue was fixed by the ExpressVPN team and the fix was then verified successfully by Cure53 in June 2021. A diff was inspected to verify the fix as working as expected.*

The function *alloc_buffer()* is used as a callback for *libuv* to provide memory. Since it ignores the *size* parameter, this might cause breakage in case *libuv* changes and relies on a certain amount of memory being allocated. Currently the function allocates 128kB while *libuv* only requests 64kB. This wastes memory and might cause memory pressure on low-memory systems.

Affected file:
xv_helium_server/srcxv_helium_server/src/uv_callbacks.c

Affected code:

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
void alloc_buffer(uv_handle_t *handle, size_t suggested_size, uv_buf_t *buf) {
  // Allocate the buffer
  buf->base = jecalloc(1, HE_SERVER_BUFFER_SIZE);
  HE_CHECK_WITH_MSG(buf->base != NULL, "Unable to allocate buffer for incoming
data\n");
  // Set the size
  buf->len = HE_SERVER_BUFFER_SIZE;
}
```

It is recommended to implement this function similar to *alloc_buffer_tcp()*, which honors the *size* parameter or simply uses *alloc_buffer_tcp()* instead of calling *alloc_buffer()*.

EXP-04-008 lua-crypt: No enforcement of hash algorithm *(Info)*

Note: *The ExpressVPN team states that hashes provided upstream are guaranteed to use sha512-crypt. No additional action will be taken here.*

The function *auth_user(username, password)* is used to authenticate users with no check being done on the specified hashing algorithm. This fosters usage of old and insecure hashing algorithms like MD5, DES, or others that are not considered secure anymore.

Affected file:
xv_helium_server/srcxv_helium_server/lua/he_auth.lua

Affected code:

```
function auth_user(username, password)
  load_db()
  stmt:bind_values(username)

  retval = false

  for row in stmt:nrows() do
        retval = crypt.check(password, row.encrypted_credentials)
    end

  unload_db()
```

It is recommended to check the prefix[2] of the hash before the comparison to prevent the usage of weak algorithms. It is also advised to migrate from *Crypt* to a more secure key derivation function (KDF) such as *Argon2* or *PBKDF2* if the computed time should be seen as acceptable.

[2] https://github.com/jprjr/lua-crypt/blob/feb4acc58355d9a8862c182768ae5962df67b9f5/crypt.lua#L20

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

EXP-04-009 Libhelium: Version numbers not checked *(Info)*

Note: *This issue was fixed by the ExpressVPN team and the fix was then verified successfully by Cure53 in June 2021. A diff was inspected to verify the fix as working as expected.*

The Helium protocol sends various information with the packets, such as the "*He*" header and version information. The client code currently sets the version information to 1.0 but this is not checked on the receiving end. In case this number gets increased and the protocol changes, this might lead to issues.

Affected file:
xv_helium_server/xv_libhelium/src/he/flow.c

It is recommended to check the version numbers supplied and reject packets with a version that is not 1.0.

EXP-04-014 Server: Incorrect pointer checked after allocation *(Info)*

Note: *This issue was fixed by the ExpressVPN team and the fix was then verified successfully by Cure53 in June 2021. A diff was inspected to verify the fix as working as expected.*

The function *he_internal_schedule_client_activity()* allocates several buffers and checks if the allocations succeeded. The pointer *req* is checked after allocating memory for *ca_line*, instead of checking *ca_line*. Error allocations for *ca_line* will not be detected, which might lead to crashes in low memory situations.

Affected file:
xv_helium_server/src/client_activities.c

Affected code:

```
char *ca_line = jecalloc(1, 1024);
HE_CHECK_WITH_MSG(req, "Unable to allocate new output buffer");
```

It is recommended to check *ca_line* instead of *req* by calling *HE_CHECK_WITH_MSG(ca_line, "<Put relevant message here>");*.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Conclusions

The outcomes of this Cure53 assessment, which tackled the ExpressVPN Lightway protocol and its connected sources, are generally positive. After spending twenty-two person-days on the examinations delineated for WP1 and WP2 in March 2021, the Cure53 testing team revealed fourteen shortcomings that should be tackled to improve the complex. To comment on the progress of the project in detail, it needs to be underlined that the tests and audits of the ExpressVPN Lightway protocol and sources moved forward at an excellent pace, with no major roadblocks in the way of the team. Some specific remarks on the complex and flaws ensue.

The internal state machine for example is distributed over multiple files and partly used by client and server, which makes it hard to examine. This applies particularly to checking the state machine of the protocol and auditing various state changes. The use of a whole plugin framework for just one plugin seems to be a bit too much in terms of unnecessary complexity, meaning that the code could be simplified by integrating the plugin into the core code. In the same vein, the addition of Lua for just a database lookup seems excessive as well. A lot of overhead and third-party code could be removed if that database lookup and password check were instead implemented in C. It was noticed that the client and server use different *libuv* versions which might cause incompatibilities or differentials, ultimately leading to bugs.

It should be ensured that all third-party components are updated to their latest versions in both the client and on the server, with the important aim of avoiding compatibility issues and known bugs in the utilized libraries. The codebase observed on Lightway Core follows consistent coding patterns and exhibits - in the testers' view - a high quality. Although the use of callbacks in various areas makes the code hard to follow in some areas, this cannot be avoided due to the use of *libuv.* It needs to be underscored that one bug pattern exists and envelops DoS issues. In the given scenario of a VPN protocol. This is clearly suboptimal and ill-advised.

At the same time, at least in this audit iteration, no *Critical* severity bugs could be spotted. It cannot be disregarded, however, that one issue came very close, as mentioned in EXP-04-013. To conclude, the scope of the ExpressVPN Lightway protocol assessed by Cure53 in this project makes a relatively robust impression. This holds despite the number of findings listed in this report. It is crucial to observe that the fixes are rather trivial to implement.

Given that not many vulnerabilities were found, it is expected that the implementation should be good for production use once the issues are addressed. It is recommended to frequently look at the code in case significant updates are committed. Ongoing security

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

engagements will make it more feasible to maintain the desirable state of privacy and security posture.

Cure53 would like to thank Walter, Dan G., Pete M., David W.F. and Aaron E. from the ExpressVPN team for their excellent project coordination, support and assistance, both before and during this assignment.