CVE-2024-5535: `SSL_select_next_proto` buffer overread

June 27, 2024

celebrating a decade of publishing your heap over the internet

Since 2011, a bug has existed in OpenSSL that means innocuous code like:

```
require('tls').connect({port: 443, NPNProtocols: new Uint8Array()}, function(c) {})
```

or (equivalently, in Python):

```
import ssl, socket
assert ssl.HAS_NPN

ctx = ssl.create_default_context()
ctx.set_npn_protocols([])
ctx.load_verify_locations('root.crt')
sock = socket.create_connection(('127.0.0.1', 443))
sock = ctx.wrap_socket(sock, server_hostname='localhost')
sock.write('hello')
```

Silently sends up to 255 bytes of the client's heap to the server. The server must support NPN, and the heap data is encrypted in transit.

This is confirmed to affect Python <= 3.9 and Node <= 9. It may equally affect any program that calls `SSL_select_next_proto` with a `client` buffer that is not a valid list of protocols (this includes an empty buffer).

Meeting those constraints is quite unlikely nowadays:

- NPN is a precursor to ALPN and was abandoned in 2012. It is very uncommon on internet servers now.
- Node.js 10 and later removed NPN support, and is well past end-of-life.
- Python 3.10 and later removed NPN support.

However, they *were* much more common in the lifetime of this bug. **You should review your historic usage** of `SSL_select_next_proto` and if you could have ever triggered this bug, I would

suggest rolling any secrets available to affected programs.

Android was affected up until 2014, with the heap leak behaviour observed independently by an okhttp developer.

Affected versions

- All versions of OpenSSL 1.0.x, 1.1.x, 3.x.
- BoringSSL was also affected, and have already landed their fix.
- LibreSSL retains the faulty `SSL_select_next_proto`, however NPN support was removed in 2017 so is not meaningfully broken.

Timeline

- **2011-11-13** - Bug introduced.
- **2014-04-10** - Work-around for bug added to Android after discovery by okhttp developer.
- **2024-04-23** - Discovery of `SSL_select_next_proto` memory unsafety while rewriting it in rust.
- **2024-05-02** - Discovery that Python 3.9 was practically affected
- **2024-05-02** - Report to OpenSSL project
- **2024-05-08** - Chase-up
- **2024-05-08** - Acknowledgement from OpenSSL project
- **2024-05-22** - Report to BoringSSL
- **2024-05-23** - Discovery that Node.js 9 was practically affected
- **2024-05-30** - Assigned CVE-2024-5535 and assessed as "Low" severity
- **2024-05-31** - Reported to security@python.org by David Benjamin
- **2024-06-04** - Python security confirm "Low" severity
- **2024-06-25** - In preparing this blog post, discovery that Android was affected until 2014.
- **2024-06-27** - Advisory released.
- **2024-11-08** - Google VRP rewards $500.

Conclusions

- The various audits, code reviews, static analysis, dynamic analysis, fuzzing, and popularity were not sufficient to remedy this bug earlier.

  Though valgrind immediately points it out once triggered, and a really basic static pointer bounds analysis should also find it (though likely with a huge false-positive rate in other code.)

- The known exposures to this bug (Python, Node, Android) are probably not all of them, but most likely will be historic.

Acknowledgements

- This discovery was made in the course of building rustls-libssl: this work was funded by ISRG Prossimo.
- Thanks to Daniel McCarney for pointing out the divergence in our `SSL_select_next_proto`.
- Thanks to Rich Salz @ Akamai for advice on communications with the OpenSSL project.
- Thanks to David Benjamin @ Google for advice and analysis, and taking care of the report to Python.

Report [1]

`SSL_select_next_proto` called with `client_len == 0` unconditionally reads `client[0]` and returns the pointer `client + 1`. Thereafter, `*out` points outside a valid address, and `*outlen` contains an undefined value.

Minimal reproducer:

```
#include <openssl/ssl.h>
#include <stdint.h>
#include <stdio.h>

int main() {
  uint8_t client_input[] = {};
  uint8_t *output = NULL;
  uint8_t output_len = 0;
  const uint8_t server_input[] = {2, 'h', '2'};

  int ret = SSL_select_next_proto(&output, &output_len, server_input,
                                  sizeof(server_input), client_input, 0);
  printf("ret = %d\n", ret);
  printf("output = %p\n", output);
  printf("output_len = %u\n", (unsigned)output_len);
  return 0;
}
```

Impact

`SSL_select_next_proto` is a normal API function so the parameters passed to it are ultimately up to the application. However it is almost always used inside an ALPN or NPN callback:

If `SSL_select_next_proto` is used straightforwardly in a server's ALPN callback, there is no impact because other code in openssl validates the client's extension as being non-empty.

If `SSL_select_next_proto` is used straightforwardly in a client's NPN callback, the client data and its length is typically a matter of local configuration and not under openssl's control. Most importantly, the impact of the wrong length and pointer being returned from this function is a catastrophic memory safety failure: the memory is copied, placed into the `selected_protocol` field of the `NextProtocolNegotiationEncryptedExtension` and sent to the peer!

As a realistic demonstration, in python3.9[2] the following code sees OpenSSL send a chunk of heap to the server:

```
import ssl, socket
assert ssl.HAS_NPN

ctx = ssl.create_default_context()
ctx.set_npn_protocols([])
ctx.load_verify_locations('root.crt')

sock = socket.create_connection(('127.0.0.1', 443))
sock = ctx.wrap_socket(sock, server_hostname='localhost')
sock.write('hello')
```

The peer receives (for example):

```
NextProtocolNegotiationEncryptedExtension {
  selected_protocol: c2d60ba35f0000000000000000000000000000000000310000000000000000
                     0000000000000060f3030d5e7f0000e0f5b9f1a65f00002006aaf1a65f0000
                     d04b0f0d5e7f000021000000000000000f6b9f1a65f000030f6b9f1a65f000
                     0d0ebb9f1a65f00003100000000000000,
  padding: 00000000000000000000000000000000,
}
```

1 with corrections

2 later versions of python have dropped NPN support.

Joe Birr-Pixton Mail: jbp@jbp.io GitHub: github.com/ctz Bluesky: bsky.app/profile/jbp.io