Source Code Audit on nghttp3 and ngtcp2 for the Open Source Technology and Investment Fund

Final Report and Management Summary

2025-05-06

PUBLIC

X41 D-Sec GmbH
Krefelder Str. 123
D-52070 Aachen
Amtsgericht Aachen: HRB19989

https://x41-dsec.de/
info@x41-dsec.de

Organized by the Open Source Technology Improvement Fund

Revision	*Date*	*Change*	*Author(s)*
1	2025-03-20	Final Report and Management Summary	C. Mayr, D. Gstir, H. Mösl-Canaval, L. Kofler

Contents

1 Executive Summary **4**

2 Introduction **6**
2.1 Methodology 7
2.2 Findings Overview 8
2.3 Scope 8
2.4 Coverage 8
2.5 Recommended Further Tests 11

3 Rating Methodology **12**
3.1 CVSS 12
3.2 Severity Mapping 15
3.3 Common Weakness Enumeration 15

4 Results **16**
4.1 Findings 16
4.2 Informational Notes 17

5 About X41 D-Sec GmbH **22**

Dashboard

Target	
Customer	Open Source Technology and Investment Fund
Name	nghttp3, ngtcp2
Type	Source Code
Version	nghttp3 v1.7.0, ngtcp2 v1.10.0
Engagement	
Type	Source Code Security Audit
Consultants	4: Christian Mayr, David Gstir, Hannes Mösl-Canaval, Lorenz Kofler
Engagement Effort	25 person-days, 2025-02-17 to 2025-03-20
Total issues found	0

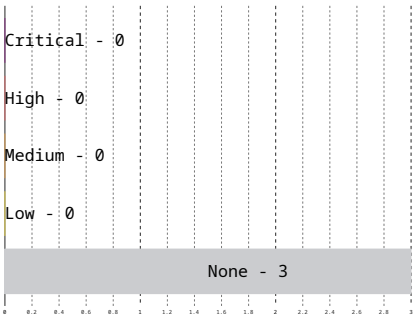

Figure 1: Issues Overview

1 Executive Summary

In February and March 2025, X41 D-Sec GmbH performed a Source Code Security Audit against the open-source repositories *nghttp3* and *ngtcp2* to identify vulnerabilities and weaknesses in the source code.

No directly exploitable issues were discovered during the audit. However, three issues without directly security impact were identified.

ngtcp2 implements QUIC, a network protocol aiming to improve the performance of connection-oriented web applications. On top of this, *nghttp3* implements HTTP/3, which aims to improve latency and loading times compared to HTTP/2 and earlier. Part of *nghttp3* is an implementation of QPACK, a compression format for HTTP fields that avoids head-of-line blocking by allowing correctness during out-of-order delivery.

X41 D-Sec GmbH conducted a comprehensive source code audit against the repositories in scope. The primary objective was to identify high-level logic bugs in HTTP over QUIC, as well as vulnerabilities that are archetypal of low-level languages like C/C++. The IETF Standard Track documents were extensively used as reference documentation and for a preliminary threat model assessment. For additional dynamic testing, auditors expanded fuzzing coverage to specific components and extensively tested them using AFL++.

In a source code audit, all information about the system is available to the team. The test was performed by four experienced security experts between 2025-02-17 and 2025-03-20.

One of the identified informational notes pertains to checking for stateless reset tokens. This is performed among already-retired connections, whereas the specification (RFC 9000) requires endpoints not to do this. The security impact is considered negligible because retired connections are maintained for only a few round-trips, but at minimum will allow fingerprinting the implementation.

The implementation also contains a potential timing side channel that leaks the length of packet numbers. The code is written quite carefully and avoids major timing discrepancies, but one instance was found where a loop is dependent on the packet number length. This is not currently known to have a significant security or privacy impact.

The lack of security-relevant findings highlights the strong security foundation of the audited repositories. It underscores the effectiveness of the development practices in place and the application's inherent robustness.

In conclusion, the systems appear to be on a good security level compared to systems of similar size and complexity, as well as other implementations of the here implemented standards.

2 Introduction

X41 conducted a review of the open-Source projects *nghttp3* and *ngtcp2*. *nghttp3* is an implementation of two key RFCs[1], namely:

- RFC 9114[2]: HTTP/3 mapping over QUIC[3]
- RFC 9204[4]: QPACK Header Compression for HTTP/3

The *ngtcp2* project implements the IETF[5] Standards Tracks defined by

- RFC 9000[6]: QUIC: A UDP[7]-based Multiplexed and Secure Transport
- RFC 9001[8]: Using TLS[9] to Secure QUIC
- RFC 9002[10]: QUIC Loss Detection and Congestion Control
- RFC 8999[11]: Version-Independent Properties of QUIC
- RFC 9369[12]: QUIC Version 2
- RFC 9221[13]: An Unreliable Datagram Extension to QUIC
- RFC 9287[14]: Greasing the QUIC Bit
- RFC 9368[15]: Compatible Version Negotiation for QUIC

Both projects are written in C.

[1] Request for Commentss
[2] `https://datatracker.ietf.org/doc/html/rfc9114`
[3] Quick UDP Internet Connections
[4] `https://datatracker.ietf.org/doc/html/rfc9204`
[5] Internet Engineering Task Force
[6] `https://datatracker.ietf.org/doc/html/rfc9000`
[7] User Datagram Protocol
[8] `https://datatracker.ietf.org/doc/html/rfc9001`
[9] Transport Layer Security
[10] `https://datatracker.ietf.org/doc/html/rfc9002`
[11] `https://datatracker.ietf.org/doc/html/rfc8999`
[12] `https://datatracker.ietf.org/doc/html/rfc9369`
[13] `https://datatracker.ietf.org/doc/html/rfc9221`
[14] `https://datatracker.ietf.org/doc/html/rfc9287`
[15] `https://datatracker.ietf.org/doc/html/rfc9368`

Reviewing HTTP[16] version 3 and QUIC implementations is crucial for several reasons. As HTTP/3, based on the QUIC transport protocol, becomes more widely adopted, ensuring that it is robust, secure, and efficiently implemented is essential for optimizing web performance, security, and scalability. Misconfigurations or vulnerabilities in HTTP/3 and QUIC can lead to serious security issues, including data leakage, denial of service, and traffic interception. Therefore, comprehensive reviews of these implementations help identify potential weaknesses and ensure the protocol's proper functioning in diverse network environments.

2.1 Methodology

X41 reviewed the source code in a security code audit.

A manual approach for penetration tests and for code reviews is used by X41. This process is supported by tools such as static code analyzers and industry standard web application security tools[17].

X41 adheres to established standards for source code reviewing and penetration testing. These are in particular the *CERT Secure Coding*[18] standards and the *Study - A Penetration Testing Model*[19] of the German Federal Office for Information Security.A security assessment attempts to find as many of the existing problems as possible, though it is practically never possible to rule out the likelihood of additional weaknesses being found in the future. Further, where applicable, suggestions of more resilient design patterns are given.

[16] HyperText Transfer Protocol
[17] `https://portswigger.net/burp`
[18] `https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards`
[19] `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1`

2.2 Findings Overview

DESCRIPTION	SEVERITY	ID	REF
Stateless Reset Token Is Matched Against Retired Connections	NONE	NGTCP2-CR-25-100	4.2.1
Hash Denial-of-Service Attack	NONE	NGTCP2-CR-25-101	4.2.2
Header Protection Timing Side Channels	NONE	NGTCP2-CR-25-102	4.2.3

Table 2.1: Security-Relevant Findings

2.3 Scope

The security review was performed on the following open-source repositories:

- `https://github.com/ngtcp2/nghttp3`
  - Commit: 86a72e9e64b81c770315636da8756d3ce38c3281, tagged v1.7.0
- `https://github.com/ngtcp2/ngtcp2`
  - Commit: 2372cdee861db68cabcb0e80c92adb535b0ad1b5, tagged v1.10.0

Both repositories primarily contain C and C++ code, with approximately 54 000 lines of C code and 22 000 lines of C++. The C++ code is mainly utilized for example implementations, such as HTTP/3 client and server executables.

2.4 Coverage

A security assessment attempts to find as many of the existing problems as possible, though it is practically never possible to rule out the likelihood of additional weaknesses being found in the future. Further, where applicable, suggestions of more resilient design patterns are given.

The time allocated to X41 for this assessment was sufficient to yield a good coverage of the given scope.

Due to the use of low-level languages like C and C++, the audit primarily focused on the following issues:

- Memory management problems, such as memory leaks
- Buffer overflows and out-of-bounds memory access

- Arithmetic overflows and underflows leading to unexpected behavior
- Insufficient error handling
- Improper validation of incoming packets

On a higher level, X41 also checked for:

- RFC-compliant implementations (only partially due to complexity and size of the RFCs)
- RFC-defined security considerations and their respective implementations and mitigations
- Cryptographic issues (Signature Bypass, flawed Transport Encryption)
- Data compression issues (excessive memory consumption due to "Zip Bomb" style attack)
- Handling of untrusted input (package parsing)
- Information disclosure (exposure of own or other users traffic)
- Denial-of-Service vectors (amplification attacks, excessive memory or CPU[20] consumption)
- Privacy issues (e.g. identify who is talking to whom)

The following paragraphs provide an overview of the security assessment processes as well as executed tests:

1. Static code analysis using sophisticated tools such as CodeQL[21] and semgrep[22]
2. Manual code review of the following key components:
   - QPACK header compression
   - QUIC and HTTP/3 connection handling
   - QUIC packet and frame parsing
   - QUIC amplification attack protection
   - QUIC address validation
   - QUIC stateless reset logic
   - QUIC path validation
   - Use of cryptographic primitives for key agreement and update
3. Dynamic code review was additionally performed with the QPACK binary.

The ngtcp2 audit focused on packet handling and frame parsing, as these components process untrusted input. Consequently, they were examined for logic flaws and potential memory corruption vulnerabilities. These code parts were thoroughly reviewed, along with selected aspects of frame parsing. Given the large number of frame types, only a subset was analyzed in detail. Key functions relevant to this analysis include:

[20] Central Processing Unit
[21] `https://codeql.github.com/`
[22] `https://semgrep.dev/`

- ***ngtcp2_accept()***
- ***ngtcp2_conn_read_pkt()***
- ***ngtcp2_pkt_decode_hd_long()***
- ***ngtcp2_pkt_decode_hd_short()***

Protection against amplification attacks is a important aspect of the QUIC protocol. The QUIC library must accurately track all bytes received prior to address validation and restrict the volume of data it sends, as specified in RFC 9000. This logic was reviewed and validated to ensure its compliance.

Address validation is employed to confirm that a peer can genuinely receive packets at the remote address it claims. The logic and RFC compliance for this mechanism were reviewed, specifically focusing on the Retry Packet logic, NEW_TOKEN generation and validation, and token integrity. These components were found to align with RFC 9000.

The stateless reset mechanism allows a peer to signal to another peer that it cannot maintain the current connection. Endpoints issue stateless reset tokens, and if a peer receives a valid token, the connection is terminated. It is necessary that the stateless reset token is properly validated in accordance with RFC 9000. During the review, we identified informational note 4.2.1.

As traffic security is an integral part of QUIC, the ngtcp2 code base was also reviewed for adherence to the QUIC TLS specification and proper use of cryptographic APIs. Here it is worth noting, that ngtcp2 does not directly depend on a specific cryptographic library like OpenSSL. Instead, it offers a generic API where the library user is responsible to wire ngtcp2 callbacks to the appropriate cryptographic library API[23] calls. As this puts the burden integration on the library user, ngtcp2 also provides existing implementations for BoringSSL, gnuTLS, Picotls, QuicTLS and wolfSSL.

Further, as a dynamic approach to the audit, AFL++ fuzzing harnesses were created for the following functions:

- ***nghttp3_qpack_huffman_decode()***
- ***ngtcp2_transport_params_decode()***
- ***ngtcp2_pkt_decode_hd_long()***
- ***ngtcp2_pkt_decode_hd_short()***

[23] Application Programming Interface

2.5 Recommended Further Tests

As previously mentioned, the IETF Standards Tracks defined code was not fully covered. However, certain components warrant special attention for future testing:

- the QPACK component regarding “Zip Bomb” like attacks

Furthermore, X41 recommends to apply the same testing strategy to any newly developed code and major changes.

3 Rating Methodology

Security vulnerabilities are given a purely technical rating by the testers when they are discovered during a test. Business factors and financial risks are beyond the scope of a penetration test, which focuses entirely on technical factors. However, technical results from a penetration test may be an integral part of a general risk assessment. A penetration test is based on a limited time frame and only covers vulnerabilities and security issues which have been found in the given time, there is no claim for full coverage.

The CVSS[1] is used to score all findings relevant to security. The resulting CVSS score is mapped to qualitative ratings as shown below.

3.1 CVSS

All findings relevant to security are rated by the testers using the CVSS industry standard version 3.1, revision 1.

Vulnerabilities scored with CVSS get a numeric value based on several metrics ranging from 0.0 (least worst) to 10.0 (worst).

The score captures different factors that express the impact and the ease of exploitation of a vulnerability among other factors. For a detailed description of how the scores are calculated, please see the CVSS version 3.1 specification.[2]

The metrics used to calculate the final score are grouped into three different categories.

[1] Common Vulnerability Scoring System
[2] `https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf`

The *Base Metric Group* represents the intrinsic and fundamental characteristics of a vulnerability that are constant over time and user environments. It captures the following metrics:

- Attack Vector (AV)
- Attack Complexity (AC)
- Privileges Required (PR)
- User Interaction (UI)
- Scope (S)
- Confidentiality Impact (C)
- Integrity Impact (I)
- Availability Impact (A)

The *Temporal Metric Group* represents the characteristics of a vulnerability that change over time but not among user environments. The following metrics are covered by it:

- Exploitability (E)
- Remediation Level (RL)
- Report Confidence (RC)

The *Environmental Metric Group* represents the characteristics of a vulnerability that are relevant and unique to a particular user's environment. It includes the following metrics:

- Attack Vector (MAV)
- Attack Complexity (MAC)
- Privileges Required (MPR)
- User Interaction (MUI)
- Confidentiality Requirement (MCR)
- Integrity Requirement (MIR)
- Availability Requirement (MAR)
- Scope (MS)
- Confidentiality Impact (MC)
- Integrity Impact (MI)
- Availability Impact (MA)

A CVSS vector defines a specific set of metrics and their values, and it can be used to reproduce and assess a given score. It is rendered as a string that exactly reproduces a score.

For example, the vector `CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:L/A:N` defines a base score metric with the following parameters:

- Attack Vector: Network
- Attack Complexity: High
- Privileges Required: Low
- User Interaction: Required
- Scope: Changed
- Confidentiality Impact: High
- Integrity Impact: Low
- Availability Impact: None

In this example, a network-based attacker performs a complex attack after gaining access to some privileges, by tricking a user into performing some actions. This allows the attacker to read confidential data and change some parts of that data.

The detailed scores are the following:

Metric	Score
CVSS Base Score	6.5
Impact Sub-Score	4.7
Exploitability Sub-Score	1.3
CVSS Temporal Score	Not Available
CVSS Environmental Score	Not Available
Modified Impact Sub-Score	Not Available
Overall CVSS Score	6.5

CVSS vectors can be automatically parsed to recreate the score, for example, with the CVSS calculator provided by FIRST, the organization behind CVSS: `https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:L/A:N`.

3.2 Severity Mapping

To help in understanding the results of a test, numeric CVSS scores are mapped to qualitative ratings as follows:

Severity Rating	CVSS Score
NONE	0.0
LOW	0.1–3.9
MEDIUM	4.0–6.9
HIGH	7.0–8.9
CRITICAL	9.0–10.0

3.3 Common Weakness Enumeration

The CWE[3] is a set of software weaknesses that allows vulnerabilities and weaknesses in software to be categorized. If applicable, X41 gives a CWE ID for each vulnerability that is discovered during a test.

CWE is a very powerful method for categorizing a vulnerability. It gives general descriptions and solution advice on recurring vulnerability types. CWE is developed by *MITRE*.[4] More information can be found on the CWE site at `https://cwe.mitre.org/`.

[3] Common Weakness Enumeration
[4] `https://www.mitre.org`

4 Results

This chapter describes the results of this test. The security-relevant findings are documented in Section 4.1. Additionally, findings without a direct security impact are documented in Section 4.2.

4.1 Findings

No findings with a direct security impact were identified during this audit.

4.2 Informational Notes

The following observations do not have a direct security impact, but are related to security hardening, affect functionality, or other topics that are not directly related to security. X41 recommends to mitigate these issues as well, because they often become exploitable in the future. Doing so will strengthen the security of the system and is recommended for defense in depth.

4.2.1 NGTCP2-CR-25-100: Stateless Reset Token Is Matched Against Retired Connections

Affected Component:	ngtcp2/lib/ngtcp2_conn.c:conn_on_stateless_reset()

4.2.1.1 Description

It was found that the stateless reset token validation in ngtcp2 does not fully conform to RFC 9000. Specifically, the implementation checks stateless reset tokens against retired connection IDs[1], which does not match logic described in the specification.

Section 10.3.1 (Detecting a Stateless Reset) from the RFC 9000, states:

> An endpoint **MUST NOT** check for any stateless reset tokens associated with connection IDs it has not used or for connection IDs that have been retired.

Listing 4.1 shows the non-compliant implementation.

```
1  static int conn_on_stateless_reset(ngtcp2_conn *conn, const ngtcp2_path *path,
2                                     const uint8_t *payload, size_t payloadlen) {
3    //...
4    if (!check_stateless_reset(&conn->dcid.current, path, &sr) &&
5        (!pv || (!check_stateless_reset(&pv->dcid, path, &sr) &&
6                 (!(pv->flags & NGTCP2_PV_FLAG_FALLBACK_ON_FAILURE) ||
7                  !check_stateless_reset(&pv->fallback_dcid, path, &sr))))) {
8      len = ngtcp2_ringbuf_len(&conn->dcid.retired.rb);
9      for (i = 0; i < len; ++i) {
10       dcid = ngtcp2_ringbuf_get(&conn->dcid.retired.rb, i);
11       if (check_stateless_reset(dcid, path, &sr)) {
12         break;
13       }
14     }
```

[1] Identifiers

```
15
16      //...
17  }
```

Listing 4.1: Stateless Reset Token Validation

As seen above, the code iterates over retired connection IDs in the `conn->dcid.retired.rb` ring buffer, checking their associated stateless reset tokens.

Checking a received reset token against the tokens of retired connections can have unwanted consequences and will at least allow fingerprinting of the specific implementation used by a remote endpoint.

4.2.1.2 Solution Advice

This issue was confirmed with the ngtcp2 maintainer. However, since retired connection IDs are retained only for a few round-trip times, the security impact is considered negligible. However, aligning the implementation with RFC 9000 by skipping checks against retired connection IDs is suggested to ensure full compliance.

4.2.2 NGTCP2-CR-25-101: Hash Denial-of-Service Attack

Affected Component:	ngtcp2/examples/server.cc

4.2.2.1 Description

It was found that the version of ngtcp2 in scope for this audit (1.10.0) is vulnerable to a Hash Denial-of-Service attack, as detailed in the technical advisory at `https://github.com/ncc-pbottine/QUIC-Hash-Dos-Advisory`.

4.2.2.2 Solution Advice

This issue has has already been resolved in the upstream repository and is fixed in all subsequent releases.

4.2.3 NGTCP2-CR-25-102: Header Protection Timing Side Channels

Affected Component: ngtcp2/lib/ngtcp2_conn.c

4.2.3.1 Description

Inspecting the header protection logic it was found that removing header protection is prone to leaking the packet number length and possibly other values due to subtle time differences in the removal logic.

Section 9.5 (Header Protection Timing Side Channels) of RFC 9001 states:

> An attacker could guess values for packet numbers or Key Phase and have an endpoint confirm guesses through timing side channels. Similarly, guesses for the packet number length can be tried and exposed. If the recipient of a packet discards packets with duplicate packet numbers without attempting to remove packet protection, they could reveal through timing side channels that the packet number matches a received packet. For authentication to be free from side channels, the entire process of header protection removal, packet number recovery, and packet protection removal **MUST** be applied together without timing and other side channels.

While the code of `conn_recv_pkt(...)` is written quite carefully to avoid major timing leaks, it was noticed it calls the function `decrypt_hp(...)` (see Listing 4.2) contains a loop which is bounded by the packet number length. An attacker capable of measuring subtle timing differences can use this to determine the packet number length.

```
1  static ngtcp2_ssize
2  decrypt_hp(ngtcp2_pkt_hd *hd, uint8_t *dest, const ngtcp2_crypto_cipher *hp,
3             const uint8_t *pkt, size_t pktlen, size_t pkt_num_offset,
4             const ngtcp2_crypto_cipher_ctx *hp_ctx, ngtcp2_hp_mask hp_mask) {
5
6    // ...
7
8    hd->pkt_numlen = (size_t)((dest[0] & NGTCP2_PKT_NUMLEN_MASK) + 1);
9
10   for (i = 0; i < hd->pkt_numlen; ++i) {
11     *p++ = *(pkt + pkt_num_offset + i) ^ mask[i + 1];
12   }
13
14   hd->pkt_num = ngtcp2_get_pkt_num(p - hd->pkt_numlen, hd->pkt_numlen);
15
```

```
16    return p - dest;
17  }
```

Listing 4.2: Header Protection Decryption

4.2.3.2 Solution Advice

Creating constant-time implementation for header protection removal, packet number recovery and packet protection removal together is complex and small timing leaks are currently not known to have a huge security or privacy impact. Nevertheless, X41 suggests to further harden the implementation by removing any obvious timing leaks like the one in `decrypt_hp(...)`.

5 About X41 D-Sec GmbH

X41 D-Sec GmbH is an expert provider for application security and penetration testing services. Having extensive industry experience and expertise in the area of information security, a strong core security team of world-class security experts enables X41 D-Sec GmbH to perform premium security services.

X41 has the following references that show their experience in the field:

- Source code audit of ISC BIND9 DNS server[1]
- Source code audit of the Git source code version control system[2]
- Review of the Mozilla Firefox updater[3]
- X41 Browser Security White Paper[4]
- Review of Cryptographic Protocols (Wire)[5]
- Identification of flaws in Fax Machines[6,7]
- Smartcard Stack Fuzzing[8]

The testers at X41 have extensive experience with penetration testing and red teaming exercises in complex environments. This includes enterprise environments with thousands of users and vendor infrastructures such as the Mozilla Firefox Updater (Balrog).

Fields of expertise in the area of application security encompass security-centered code reviews, binary reverse-engineering and vulnerability-discovery. Custom research and IT security consulting, as well as support services, are the core competencies of X41. The team has a strong technical background and performs security reviews of complex and high-profile applications such as Google Chrome and Microsoft Edge web browsers.

X41 D-Sec GmbH can be reached via `https://x41-dsec.de` or `mailto:info@x41-dsec.de`.

[1] `https://x41-dsec.de/news/security/research/source-code-audit/2024/02/13/bind9-security-audit/`
[2] `https://x41-dsec.de/security/research/news/2023/01/17/git-security-audit-ostif/`
[3] `https://blog.mozilla.org/security/2018/10/09/trusting-the-delivery-of-firefox-updates/`
[4] `https://browser-security.x41-dsec.de/X41-Browser-Security-White-Paper.pdf`
[5] `https://www.x41-dsec.de/reports/Kudelski-X41-Wire-Report-phase1-20170208.pdf`
[6] `https://www.x41-dsec.de/lab/blog/fax/`
[7] `https://2018.zeronights.ru/en/reports/zero-fax-given/`
[8] `https://www.x41-dsec.de/lab/blog/smartcards/`

Acronyms

API Application Programming Interface 10
CPU Central Processing Unit 9
CVSS Common Vulnerability Scoring System 12
CWE Common Weakness Enumeration 15
HTTP HyperText Transfer Protocol 7
ID Identifier 17
IETF Internet Engineering Task Force 6
QUIC Quick UDP Internet Connections 6
RFC Request for Comments 6
TLS Transport Layer Security 6
UDP User Datagram Protocol 6